

Meta Multi-Task Learning for Sequence Modeling

Junkun Chen, Xipeng Qiu,* Pengfei Liu, Xuanjing Huang

Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{jkchen16, xpqiu, pfliu14, xjhuang}@fudan.edu.cn

Abstract

Semantic composition functions have been playing a pivotal role in neural representation learning of text sequences. In spite of their success, most existing models suffer from the underfitting problem: they use the same shared compositional function on all the positions in the sequence, thereby lacking expressive power due to incapacity to capture the richness of compositionality. Besides, the composition functions of different tasks are independent and learned from scratch. In this paper, we propose a new sharing scheme of composition function across multiple tasks. Specifically, we use a shared meta-network to capture the meta-knowledge of semantic composition and generate the parameters of the task-specific semantic composition models. We conduct extensive experiments on two types of tasks, text classification and sequence tagging, which demonstrate the benefits of our approach. Besides, we show that the shared meta-knowledge learned by our proposed model can be regarded as off-the-shelf knowledge and easily transferred to new tasks.

Introduction

Deep learning models have been widely used in many natural language processing (NLP) tasks. A major challenge is how to design and learn the semantic composition function while modeling a text sequence. The typical composition models involve sequential (Sutskever, Vinyals, and Le 2014; Chung et al. 2014), convolutional (Collobert et al. 2011; Kalchbrenner, Grefenstette, and Blunsom 2014; Kim 2014) and syntactic (Socher et al. 2013; Tai, Socher, and Manning 2015; Zhu, Sobihani, and Guo 2015) compositional models.

In spite of their success, these models have two major limitations. First, they usually use a shared composition function for all kinds of semantic compositions, even though the compositions have different characteristics in nature. For example, the composition of the adjective and the noun differs significantly from the composition of the verb and the noun. Second, different composition functions are learned from scratch in different tasks. However, given a certain natural language, its composition functions should be the same (on meta-knowledge level at least), even if the tasks are different.

*Corresponding Author

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

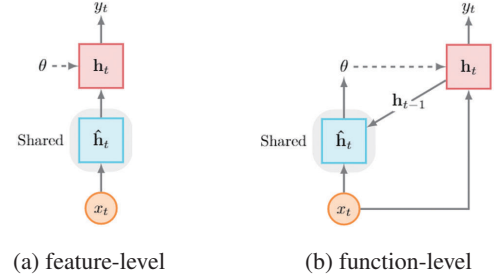


Figure 1: Two different sharing schemes. θ denotes the parameters of task-specific composition function. x_t , y_t , h_t and \hat{h}_t are the input, output, shared and private hidden states at step t . (a) The generic feature-level sharing scheme, in which the shared features will be taken as inputs for task-specific layers. (b) Our proposed function-level sharing scheme, in which a shared Meta-LSTM controls the parameters θ_t of task-specific composition function.

To address these problems, we need to design a dynamic composition function which can vary with different positions and contexts in a sequence, and share it across the different tasks. To share some meta-knowledge of composition function, we can adopt the multi-task learning (Caruana 1997). However, the sharing scheme of most neural multi-task learning methods is *feature-level sharing*, where a subspace of the feature space is shared across all the tasks. Although these sharing schemes are successfully used in various NLP tasks (Collobert and Weston 2008; Luong et al. 2015; Liu et al. 2015; Liu, Qiu, and Huang 2016; Hashimoto et al. 2017; Chen et al. 2017), they are not suitable to share the composition function.

In this paper, inspired by recent work on dynamic parameter generation (De Brabandere et al. 2016; Bertinetto et al. 2016; Ha, Dai, and Le 2016), we propose a *function-level sharing* scheme for multi-task learning, in which a shared meta-network is used to learn the meta-knowledge of semantic composition among the different tasks. The task-specific semantic composition function is generated by the meta-network. Then the task-specific composition function is used to obtain the task-specific representation of a text sequence.

The difference between two sharing schemes is shown in Figure 1. Specifically, we use two LSTMs as meta and basic (task-specific) network respectively. The meta LSTM is shared for all the tasks. The parameters of the basic LSTM are generated based on the current context by the meta LSTM, therefore the composition function is not only task-specific but also position-specific. The whole network is differentiable with respect to the model parameters and can be trained end-to-end.

We demonstrate the effectiveness of our architectures on two kinds of NLP tasks: text classification and sequence tagging. Experimental results show that jointly learning of multiple related tasks can improve the performance of each task relative to learning them independently.

Our contributions are of three-folds:

- We propose a new perspective of information sharing scheme for multi-task learning. Different from the feature-level sharing, we introduce *function-level* sharing scheme to extract the meta knowledge of semantic composition across the different tasks.
- The Meta-LSTMs not only improve the performance of multi-task learning, but also benefit the single-task learning since the parameters of the basic LSTM vary from position to position, in contrast to the same parameters used for all the positions in the standard LSTM. Thus, the of the task-specific LSTM vary from position to position, allowing for more sophisticated semantic compositions of text sequence.
- The Meta-LSTM can be regarded as a prior knowledge of semantic composition, while the basic LSTM is the posterior knowledge. Therefore, our learned Meta-LSTM also provides an efficient way of performing transfer learning (Pan and Yang 2010). Under this view, a new task can no longer be simply seen as an isolated task that starts accumulating knowledge afresh. As more tasks are observed, the learning mechanism is expected to benefit from previous experience.

Generic Neural Architecture of Multi-Task Learning for Sequence Modeling

In this section, we briefly describe generic neural architecture of multi-task learning.

Task Definition

The task of Sequence Modeling is to assign a label sequence $Y = \{y_1, y_2, \dots, y_T\}$ to a text sequence $X = \{x_1, x_2, \dots, x_T\}$. In classification task, Y is a single label. Assuming that there are K related tasks, we refer \mathcal{D}_k as the corpus of the k -th task with N_k samples:

$$\mathcal{D}_k = \{(X_i^{(k)}, Y_i^{(k)})\}_{i=1}^{N_k}, \quad (1)$$

where X_i^k and Y_i^k denote the i -th sample and its label respectively in the k -th task.

Multi-task learning (Caruana 1997) is an approach to learn multiple related tasks simultaneously to significantly improve performance relative to learning each task independently. The main challenge of multi-task learning is how to

design the sharing scheme. For the shallow classifier with discrete features, it is relatively difficult to design the shared feature spaces, usually resulting in a complex model. Fortunately, deep neural models provide a convenient way to share information among multiple tasks.

Generic Neural Architecture of Multi-Task Learning for Sequence Modeling

The generic neural architecture of multi-task learning is to share some lower layers to determine common features. After the shared layers, the remaining higher layers are parallel and independent respective to each specific task. Figure 2 illustrates the generic architecture of multi-task learning. (Collobert and Weston 2008; Liu et al. 2015; Liu, Qiu, and Huang 2016)

Sequence Modeling with LSTM There are many neural sentence models, which can be used for sequence modeling, including recurrent neural networks (Sutskever, Vinyals, and Le 2014; Chung et al. 2014), convolutional neural networks (Collobert et al. 2011; Kalchbrenner, Grefenstette, and Blunsom 2014), and recursive neural networks (Socher et al. 2013). Here we adopt recurrent neural network with long short-term memory (LSTM) due to their superior performance in various NLP tasks.

LSTM (Hochreiter and Schmidhuber 1997) is a type of recurrent neural network (RNN), and specifically addresses the issue of learning long-term dependencies. While there are numerous LSTM variants, here we use the LSTM architecture used by (Jozefowicz, Zaremba, and Sutskever 2015), which is similar to the architecture of (Graves 2013) but without peep-hole connections.

We define the LSTM *units* at each time step t to be a collection of vectors in \mathbb{R}^h : an *input gate* \mathbf{i}_t , a *forget gate* \mathbf{f}_t , an *output gate* \mathbf{o}_t , a *memory cell* \mathbf{c}_t and a *hidden state* \mathbf{h}_t . d is the number of the LSTM units. The elements of the gating vectors \mathbf{i}_t , \mathbf{f}_t and \mathbf{o}_t are in $[0, 1]$.

The LSTM is compactly specified as follows.

$$\begin{bmatrix} \mathbf{g}_t \\ \mathbf{o}_t \\ \mathbf{i}_t \\ \mathbf{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\mathbf{W} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix} + \mathbf{b} \right), \quad (2)$$

$$\mathbf{c}_t = \mathbf{g}_t \odot \mathbf{i}_t + \mathbf{c}_{t-1} \odot \mathbf{f}_t, \quad (3)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \quad (4)$$

where $\mathbf{x}_t \in \mathbb{R}^d$ is the input at the current time step; $\mathbf{W} \in \mathbb{R}^{4h \times (h+d)}$ and $\mathbf{b} \in \mathbb{R}^{4h}$ are parameters of affine transformation; σ denotes the logistic sigmoid function and \odot denotes elementwise multiplication.

The update of each LSTM unit can be written precisely as follows:

$$\mathbf{h}_t = \mathbf{LSTM}(\mathbf{h}_{t-1}, \mathbf{x}_t, \theta). \quad (5)$$

Here, the function $\mathbf{LSTM}(\cdot, \cdot, \cdot)$ is a shorthand for Eq. (2-4), and θ represents all the parameters of LSTM.

Given a text sequence $X = \{x_1, x_2, \dots, x_T\}$, we first use a lookup layer to get the vector representation (embeddings) \mathbf{x}_t of each word x_t . The output at the last moment \mathbf{h}_T can be regarded as the representation of the whole sequence.

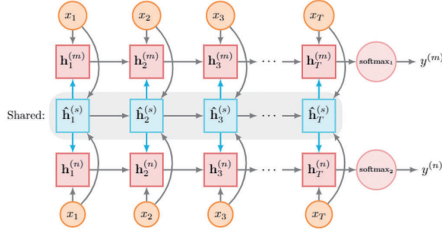


Figure 2: Generic Architecture of Multi-Task Learning. The blue modules whose output will be taken as the input of private layers are shared between different tasks.

Shared-Private Sharing Scheme To exploit the shared information between these different tasks, the general deep multi-task architecture consists of a private (task-specific) layer and a shared (task-invariant) layer. The shared layer captures the shared information for all the tasks.

The shared layer and private layer is arranged in stacked manner. The private layer takes the output of the shared layer as input. For task k , the hidden states of shared layer and private layer are:

$$\mathbf{h}_t^{(s)} = \text{LSTM}(\mathbf{x}_t, \mathbf{h}_{t-1}^{(s)}, \theta_s), \quad (6)$$

$$\mathbf{h}_t^{(k)} = \text{LSTM}\left(\begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_t^{(s)} \end{bmatrix}, \mathbf{h}_{t-1}^{(k)}, \theta_k\right) \quad (7)$$

where $\mathbf{h}_t^{(s)}$ and $\mathbf{h}_t^{(k)}$ are hidden states of the shared layer and the k -th task-specific layer respectively; θ_s and θ_k denote their parameters.

Task-specific Output Layer The task-specific representations $\mathbf{h}^{(k)}$, which is emitted by the multi-task architecture, are ultimately fed into different task-specific output layers.

Here, we use two kinds of tasks: text classification and sequence tagging.

Text Classification For task k in , the label predictor is defined as

$$\hat{\mathbf{y}}^{(k)} = \text{softmax}(\mathbf{W}^{(k)}\mathbf{h}^{(k)} + \mathbf{b}^{(k)}), \quad (8)$$

where $\hat{\mathbf{y}}^{(k)}$ is prediction probabilities for task k , $\mathbf{W}^{(k)}$ is the weight matrix which needs to be learned, and $\mathbf{b}^{(k)}$ is a bias term.

Sequence Tagging Following the idea of (Huang, Xu, and Yu 2015; Ma and Hovy 2016), we use a conditional random field (CRF) (Lafferty, McCallum, and Pereira 2001) as output layer.

Training

The parameters of the network are trained to minimise the cross-entropy of the predicted and true distributions for all tasks.

$$\mathcal{L}(\Theta) = - \sum_{k=1}^K \lambda_k \sum_{i=1}^{N_k} \mathbf{y}_i^{(k)} \log(\hat{\mathbf{y}}_i^{(k)}), \quad (9)$$

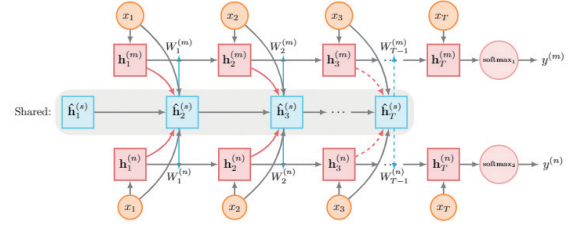


Figure 3: Architecture of Meta Multi-task Learning. The blue modules are shared between different tasks, which control the parameters of private layers.

where λ_k is the weights for each task k respectively; $\mathbf{y}_i^{(k)}$ is the one-hot vector of the ground-truth label of the sample $X_i^{(k)}$; $\hat{\mathbf{y}}_i^{(k)}$ is its prediction probabilities.

It is worth noticing that labeled data for training each task can come from completely different datasets. Following (Collobert and Weston 2008), the training is achieved in a stochastic manner by looping over the tasks:

1. Select a random task.
2. Select a mini-batch of examples from this task.
3. Update the parameters for this task by taking a gradient step with respect to this mini-batch.
4. Go to 1.

After the joint learning phase, we can use a fine tuning strategy to further optimize the performance for each task.

Meta Multi-Task Learning

In this paper, we take a very different multi-task architecture from meta-learning perspective (Brazdil et al. 2008). One goal of meta-learning is to find efficient mechanisms to transfer knowledge across domains or tasks (Lemke, Budka, and Gabrys 2015).

Different from the generic architecture with the *representational sharing* (feature sharing) scheme, our proposed architecture uses a *functional sharing* scheme, which consists of two kinds of networks. As shown in Figure 3, for each task, a basic network is used for task-specific prediction, whose parameters are controlled by a shared meta network across all the tasks.

We firstly introduce our architecture on single task, then apply it for multi-task learning.

Meta-LSTMs for Single Task

Inspired by recent work on dynamic parameter prediction (De Brabandere et al. 2016; Bertinetto et al. 2016; Ha, Dai, and Le 2016), we also use a meta network to generate the parameters of the task network (basic network). Specific to text classification, we use LSTM for both the networks in this paper, but other options are possible.

There are two networks for each single task: a basic LSTM and a meta LSTM.

Basic-LSTM For each specific task, we use a basic LSTM to encode the text sequence. Different from the standard LSTM, the parameters of the basic LSTM is controlled by a meta vector \mathbf{z}_t , generated by the meta LSTM. The new equations of the basic LSTM are

$$\begin{bmatrix} \mathbf{g}_t \\ \mathbf{o}_t \\ \mathbf{i}_t \\ \mathbf{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\mathbf{W}(\mathbf{z}_t) \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix} + \mathbf{b}(\mathbf{z}_t) \right), \quad (10)$$

$$\mathbf{c}_t = \mathbf{g}_t \odot \mathbf{i}_t + \mathbf{c}_{t-1} \odot \mathbf{f}_t, \quad (11)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \quad (12)$$

where $\mathbf{W}(\mathbf{z}_t) : \mathbb{R}^z \rightarrow \mathbb{R}^{4h \times (h+d)}$ and $\mathbf{b}(\mathbf{z}_t) : \mathbb{R}^z \rightarrow \mathbb{R}^{4h}$ are dynamic parameters controlled by the meta network.

Since the output space of the dynamic parameters $\mathbf{W}(\mathbf{z}_t)$ is very large, its computation is slow without considering matrix optimization algorithms. Moreover, the large parameters makes the model suffer from the risk of overfitting. To remedy this, we define $\mathbf{W}(\mathbf{z}_t)$ with a low-rank factorized representation of the weights, analogous to the Singular Value Decomposition.

The parameters $\mathbf{W}(\mathbf{z}_t)$ and $\mathbf{b}(\mathbf{z}_t)$ of the basic LSTM are computed by

$$\mathbf{W}(\mathbf{z}_t) = \begin{bmatrix} P_c \mathbf{D}(\mathbf{z}_t) Q_c \\ P_o \mathbf{D}(\mathbf{z}_t) Q_o \\ P_i \mathbf{D}(\mathbf{z}_t) Q_i \\ P_f \mathbf{D}(\mathbf{z}_t) Q_f \end{bmatrix} \quad (13)$$

$$\mathbf{b}(\mathbf{z}_t) = \begin{bmatrix} B_c \mathbf{z}_t \\ B_o \mathbf{z}_t \\ B_i \mathbf{z}_t \\ B_f \mathbf{z}_t \end{bmatrix} \quad (14)$$

where $P_* \in \mathbb{R}^{h \times z}$, $Q_* \in \mathbb{R}^{z \times d}$ and $B_* \in \mathbb{R}^{h \times z}$ are parameters for $* \in \{c, o, i, f\}$.

Thus, our basic LSTM needs $(8hz + 4dz)$ parameters, while the standard LSTM has $(4h^2 + 4hd + 4h)$ parameters. With a small z , the basic LSTM needs less parameters than the standard LSTM. For example, if we set $d = h = 100$ and $z = 20$, our basic LSTM just needs 24,000 parameter while the standard LSTM needs 80,400 parameters.

Meta-LSTM The Meta-LSTM is usually a smaller network, which depends on the input \mathbf{x}_t and the previous hidden state \mathbf{h}_{t-1} of the basic LSTM.

The Meta-LSTM cell is given by:

$$\begin{bmatrix} \hat{\mathbf{g}}_t \\ \hat{\mathbf{o}}_t \\ \hat{\mathbf{i}}_t \\ \hat{\mathbf{f}}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\mathbf{W}_m \begin{bmatrix} \mathbf{x}_t \\ \hat{\mathbf{h}}_{t-1} \\ \mathbf{h}_{t-1} \end{bmatrix} + \mathbf{b}_m \right), \quad (15)$$

$$\hat{\mathbf{c}}_t = \hat{\mathbf{g}}_t \odot \hat{\mathbf{i}}_t + \hat{\mathbf{c}}_{t-1} \odot \hat{\mathbf{f}}_t, \quad (16)$$

$$\hat{\mathbf{h}}_t = \hat{\mathbf{o}}_t \odot \tanh(\hat{\mathbf{c}}_t), \quad (17)$$

$$\mathbf{z}_t = \mathbf{W}_z \hat{\mathbf{h}}_t, \quad (18)$$

where $\mathbf{W}_m \in \mathbb{R}^{4m \times (d+h+m)}$ and $\mathbf{b}_m \in \mathbb{R}^{4m}$ are parameters of Meta-LSTM; $\mathbf{W}_z \in \mathbb{R}^{z \times m}$ is a transformation matrix.

Thus, the Meta-LSTM needs $(4m(d+h+m+1) + mz)$ parameters. When $d = h = 100$ and $z = m = 20$, its parameter number is 18,080. The total parameter number of the whole networks is 42,080, nearly half of the standard LSTM.

We precisely describe the update of the units of the Meta-LSTMs as follows:

$$[\hat{\mathbf{h}}_t, \mathbf{z}_t] = \text{Meta-LSTM}(\mathbf{x}_t, \hat{\mathbf{h}}_{t-1}, \mathbf{h}_{t-1}; \theta_m), \quad (19)$$

$$\mathbf{h}_t = \text{Basic-LSTM}(\mathbf{x}_t, \mathbf{h}_{t-1}; \mathbf{z}_t, \theta_b) \quad (20)$$

where θ_m and θ_b denote the parameters of the Meta-LSTM and Basic-LSTM respectively.

Compared to the standard LSTM, the Meta-LSTMs have two advantages. One is the parameters of the Basic-LSTM is dynamically generated conditioned on the input at the position, while the parameters of the standard LSTM are the same for all the positions, even though different positions have very different characteristics. Another is that the Meta-LSTMs usually have less parameters than the standard LSTM.

Meta-LSTMs for Multi-Task Learning

For multi-task learning, we can assign a basic network to each task, while sharing a meta network among tasks. The meta network captures the meta (shared) knowledge of different tasks. The meta network can learn at the meta-level of predicting parameters for the basic task-specific network.

For task k , the hidden states of the shared layer and the private layer are:

$$[\hat{\mathbf{h}}_t^{(s)}, \mathbf{z}_t^{(s)}] = \text{Meta-LSTM}(\mathbf{x}_t, \hat{\mathbf{h}}_{t-1}^{(s)}, \mathbf{h}_{t-1}^{(k)}; \theta_m^{(s)}), \quad (21)$$

$$\mathbf{h}_t^{(k)} = \text{Basic-LSTM}(\mathbf{x}_t, \mathbf{h}_{t-1}^{(k)}; \mathbf{z}_t^{(s)}, \theta_b^{(k)}) \quad (22)$$

where $\hat{\mathbf{h}}_t^{(s)}$ and $\mathbf{h}_t^{(k)}$ are the hidden states of the shared meta LSTM and the k -th task-specific basic LSTM respectively; $\theta_m^{(s)}$ and $\theta_b^{(k)}$ denote their parameters. The superscript (s) indicates the parameters or variables are shared across the different tasks.

Experiment

In this section, we investigate the empirical performances of our proposed model on two multi-task datasets. Each dataset contains several related tasks.

Exp-I: Multi-task Learning of text classification

We first conduct our experiment on classification tasks.

Datasets For classification task, we test our model on 16 classification datasets, the first 14 datasets are product reviews that collected based on the dataset¹, constructed by Blitzer et al. (2007), contains Amazon product reviews from different domains: Books, DVDs, Electronics and Kitchen and so on. The goal in each domain is to classify a product review as either positive or negative. The datasets in each

¹<https://www.cs.jhu.edu/~mdredze/datasets/sentiment/>

Datasets	Train Size	Dev. Size	Test Size	Class	Avg. Length	Voc. Size
Books	1400	200	400	2	159	62K
Elec	1398	200	400	2	101	30K
DVD	1400	200	400	2	173	69K
Kitchen	1400	200	400	2	89	28K
Apparel	1400	200	400	2	57	21K
Camera	1397	200	400	2	130	26K
Health	1400	200	400	2	81	26K
Music	1400	200	400	2	136	60K
Toys	1400	200	400	2	90	28K
Video	1400	200	400	2	156	57K
Baby	1300	200	400	2	104	26K
Mag	1370	200	400	2	117	30K
Soft	1315	200	400	2	129	26K
Sports	1400	200	400	2	94	30K
IMDB	1400	200	400	2	269	44K
MR	1400	200	400	2	21	12K

Table 1: Statistics of sixteen multi-task datasets for text classification.

domain are partitioned randomly into training data, development data and testing data with the proportion of 70%, 10% and 20% respectively. The detailed statistics are listed in Table 1.

The remaining two datasets are two sub-datasets about movie reviews.

- **IMDB** The movie reviews² with labels of subjective or objective (Maas et al. 2011).
- **MR** The movie reviews³ with two classes (Pang and Lee 2005).

Competitor Models For single-task learning, we compare our Meta-LSTMs with three models.

- **LSTM**: the standard LSTM with one hidden layer;
- **HyperLSTMs**: a similar model which also uses a small network to generate the weights for a larger network (Ha, Dai, and Le 2016).

For multi-task learning, we compare our Meta-LSTMs with the generic shared-private sharing scheme.

- **ASP-MTL**: Proposed by (Liu, Qiu, and Huang 2017), using adversarial training method on PSP-MTL.
- **PSP-MTL**: Parallel shared-private sharing scheme, using a fully-shared LSTM to extract features for all tasks and concatenate with the outputs from task-specific LSTM.
- **SSP-MTL**: Stacked shared-private sharing scheme, introduced in Section 2.

Hyperparameters and Training The networks are trained with backpropagation and the gradient-based optimization is performed using the Adagrad update rule (Duchi, Hazan, and Singer 2011).

²<https://www.cs.jhu.edu/~mdredze/datasets/sentiment/unprocessed.tar.gz>

³<https://www.cs.cornell.edu/people/pabo/movie-review-data/>.

Hyper-parameters	classification
Embedding dimension: d	200
Size of \mathbf{h} in Basic-LSTM: h	100
Size of $\hat{\mathbf{h}}$ in Meta-LSTM: m	40
Size of meta vector \mathbf{z} : z	40
Initial learning rate	0.1
Regularization	$1E-5$

Table 2: Hyper-parameters of our models.

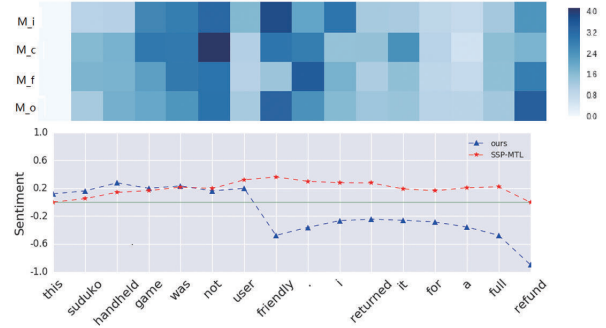


Figure 4: The lower figure presents sentiment prediction for each time step. Y-axis represents the predicted score, greater than zero for positive, less than zero for negative. The other one presents changes of the matrix generated by Eq.(13). We split the matrix into four pieces, which represent the matrix to compute input gate, new input, forget gate, and output gate respectively. And we calculate the changes of matrices each time step.

The word embeddings for all of the models are initialized with the 200d GloVe vectors (6B token version, (Pennington, Socher, and Manning 2014)) and fine-tuned during training to improve the performance. The mini-batch size is set to 16. The final hyper-parameters are set as Table 2.

Experiment result Table 3 shows the classification accuracies on the tasks of product reviews.

The row of “Single Task” shows the results for single-task learning. With the help of Meta-LSTMs, the performances of the 16 subtasks are improved by an average of 3.2%, compared to the standard LSTM. However, the number of parameters is a little more than standard LSTM and much less than the HyperLSTMs.

For multi-task Learning, our model also achieves a better performance than our competitor models, with an average improvement of 5.1% to average accuracy of single task and 2.2% to best competitor Multi-task model. The main reason is that our models can capture more abstractive shared information. With a meta LSTM to generate the matrices, the layer will become more flexible.

With the meta network, our model can use quite a few parameters to achieve the state-of-the-art performances.

We have experimented various z size in our multi-task model, where $z \in [20, 30, \dots, 60]$, and the difference of the average accuracies of sixteen datasets is less than 0.8%,

Task	Single Task				Multiple Tasks				Transfer
	LSTM	HyperLSTM	MetaLSTM	Avg.	ASP-MTL*	PSP-MTL	SSP-MTL	Meta-MTL(ours)	Meta-MTL(ours)
Books	79.5	78.3	83.0	80.2	87.0	84.3	85.3	87.5	86.3
Electronics	80.5	80.7	82.3	81.2	89.0	85.7	87.5	89.5	86.0
DVD	81.7	80.3	82.3	81.4	87.4	83.0	86.5	88.0	86.5
Kitchen	78.0	80.0	83.3	80.4	87.2	84.5	86.5	91.3	86.3
Apparel	83.2	85.8	86.5	85.2	88.7	83.7	86.0	87.0	86.0
Camera	85.2	88.3	88.3	87.2	91.3	86.5	87.5	89.7	87.0
Health	84.5	84.0	86.3	84.9	88.1	86.5	87.5	90.3	88.7
Music	76.7	78.5	80.0	78.4	82.6	81.3	85.8	86.3	85.7
Toys	83.2	83.7	84.3	83.7	88.8	83.5	87.0	88.5	85.3
Video	81.5	83.7	84.3	83.1	85.5	83.3	85.5	88.3	85.5
Baby	84.7	85.5	84.0	84.7	89.8	86.5	87.0	88.0	86.0
Magazines	89.2	91.3	92.3	90.9	92.4	88.3	88.0	91.0	90.3
Software	84.7	86.5	88.3	86.5	87.3	84.0	86.0	88.5	86.5
Sports	81.7	82.0	82.5	82.1	86.7	82.0	85.0	86.7	85.7
IMDB	81.7	77.0	83.5	80.7	85.8	82.0	84.5	88.0	87.3
MR	72.7	73.0	74.3	73.3	77.3	74.5	75.8	77.0	75.5
AVG	81.8	82.4	84.0	82.8	87.2 _(+4.4)	83.7 _(+0.9)	85.7 _(+2.9)	87.9_(+5.1)	85.9 _(+3.1)
Parameters	120K	321K	134K		5490k	2056K	1411K	1339K	1339K

Table 3: Accuracies of our models on 16 datasets against typical baselines. The numbers in brackets represent the improvements relative to the average performance (Avg.) of three single task baselines. *is from (Liu, Qiu, and Huang 2017)

which indicates that the meta network with less parameters can also generate a basic network with a considerable good performance.

Visualization To illustrates the insight of our model, we randomly sample a sequence from the development set of Toys task. In Figure 4 we predict the sentiment scores each time step. Moreover, to describe how our model works, we visualize the changes of matrices generated by Meta-LSTM, the changes **diff** are calculate by Eq.23.

As we see it, the matrices change obviously facing the emotional vocabulary like "friendly", "refund", and slowly change to a normal state. They can also capture words that affect sentiments like "not". For this case, SSP-MTL give a wrong answer, it captures the emotion word "refund", but it makes an error on pattern "not user friendly", we consider that it's because fixed matrices don't have satisfactorily ability to capture long patterns' emotions and information. Dynamic matrices generated by Meta-LSTM will make the layer more flexible.

$$\text{diff}^{(k)} = \text{mean}\left(\frac{\text{abs}(\mathbf{W}^{(k)} - \mathbf{W}^{(k-1)})}{\text{abs}(\mathbf{W}^{(k-1)})}\right), \quad (23)$$

Convergence speed during shared training Figure 5 shows the learning curves of various multi-task model on the 16 classification datasets.

Because it's inappropriate to evaluate different tasks every training step during shared parameters training since mini-batch of which tasks are selected randomly, so we use the average loss after every epoch. We can find that our proposed model is more efficient to fit the train datasets than our competitor models, and get better performance on the dev

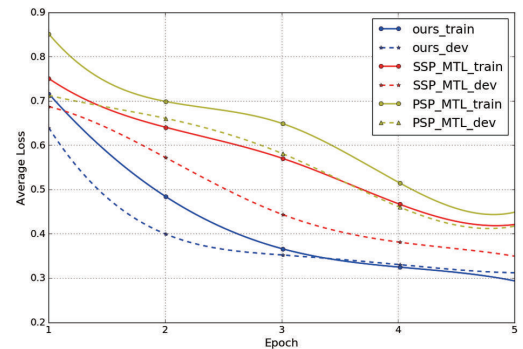


Figure 5: The train loss and dev loss of various multitask model decaying during the share training epochs.

datasets. Therefore, we can consider that our model could learn shareable knowledge more effectively.

Meta knowledge transfer Since our Meta-LSTM captures some meta knowledge of semantic composition, which should have an ability of being transferred to a new task. Under this view, a new task can no longer be simply seen as an isolated task that starts accumulating knowledge afresh. As more tasks are observed, the learning mechanism is expected to benefit from previous experience.

The meta network can be considered as off-the-shelf knowledge and then be used for unseen new tasks.

To test the transferability of our learned Meta-LSTM, we also design an experiment, in which we take turns choosing 15 tasks to train our model with multi-task learning, then the learned Meta-LSTM are transferred to the remaining one task. The parameters of transferred Meta-LSTM, $\theta_m^{(s)}$

Tagging Dataset	Task	Training	Dev	Test
WSJ	POS Tagging	912344	131768	129654
CoNLL 2000	Chunking	211727	-	47377
CoNLL 2003	NER	204567	51578	46666

Table 4: Statistics of four multi-task datasets for sequence tagging.

	CoNLL2000 [†]	CoNLL2003 [‡]	WSJ [‡]
Single Task Model:			
LSTM+CRF [◆]	93.67	89.91	97.25
Meta-LSTM+CRF	93.71	90.08	97.30
Collobert et al. (2011)	94.32	89.59	97.29
Multi-Task Model:			
LSTM-SSP-MTL+CRF	94.32	90.38	97.23
Meta-LSTM-MTL+CRF	95.11	90.72	97.45

Table 5: Accuracy rates of our models on three tasks for sequence tagging. [†] means evaluated by F1 score(%), [‡] means evaluated by accuracy(%). [◆] is the model implemented in (Huang, Xu, and Yu 2015).

in Eq.(21), are fixed and cannot be updated on the new task.

The results are also shown in the last column of Table 3. With the help of meta knowledge, we observe an average improvement of 3.1% over the average accuracy of single models, and even better than other competitor multi-task models. This observation indicates that we can save the meta knowledge into a meta network, which is quite useful for a new task.

Exp-II: Multi-task Learning of Sequence Tagging

In this section, we conduct experiment for sequence tagging. Similar to (Huang, Xu, and Yu 2015; Ma and Hovy 2016), we use the bi-directional Meta-LSTM layers to encode the sequence and a conditional random field (CRF) (Lafferty, McCallum, and Pereira 2001) as output layer. The hyperparameters settings are same to Exp-I, but with 100d embedding size and 30d Meta-LSTM size.

Datasets For sequence tagging task, we use the Wall Street Journal(WSJ) portion of Penn Treebank (PTB) (Marcus, Marcinkiewicz, and Santorini 1993), CoNLL 2000 chunking, and CoNLL 2003 English NER datasets. The statistics of these datasets are described in Table 4.

Experiment result Table 5 shows the accuracies or F1 scores on the sequence tagging datasets of our models, compared to some state-of-the-art results. As shown, our proposed Meta-LSTM performs better than our competitor models whether it is single or multi-task learning.

Result Analysis

From the above two experiments, we have empirically observed that our model is consistently better than the competitor models, which shows our model is very robust. Explicit to multi-task learning, our model outperforms SSP-MTL and PSP-MTL by a large margin with fewer param-

eters, which indicates the effectiveness of our proposed functional sharing mechanism.

Related Work

One thread of related work is neural networks based multi-task learning, which has been proven effective in many NLP problems (Collobert and Weston 2008; Glorot, Bordes, and Bengio 2011; Liu et al. 2015; Liu, Qiu, and Huang 2016). In most of these models, the lower layers are shared across all tasks, while top layers are task-specific. This kind of sharing scheme divide the feature space into two parts: the shared part and the private part. The shared information is representation-level, whose capacity grows linearly as the size of shared layers increases.

Different from these models, our model captures the function-level sharing information, in which a meta-network captures the meta-knowledge across tasks and controls the parameters of task-specific networks.

Another thread of related work is the idea of using one network to predict the parameters of another network. De Brabandere et al. (2016) used a filter-generating network to generate the parameters of another dynamic filter network, which implicitly learn a variety of filtering operations. Bertinetto et al. (2016) introduced a learnnet for one-shot learning, which can predicts the parameters of a second network given a single exemplar. Ha, Dai, and Le (2016) proposed the model hypernetwork, which uses a small network to generate the weights for a larger network. In particular, their proposed hyperLSTMs is same with our Meta-LSTMs except for the computational formulation of the dynamic parameters. Besides, we also use a *low-rank approximation* to generate the parameter matrix, which can reduce greatly the model complexity, while keeping the model ability.

Conclusion and Future Work

In this paper, we introduce a novel knowledge sharing scheme for multi-task learning. The difference from the previous models is the mechanisms of sharing information among several tasks. We design a meta network to store the knowledge shared by several related tasks. With the help of the meta network, we can obtain better task-specific sentence representation by utilizing the knowledge obtained by other related tasks. Experimental results show that our model can improve the performances of several related tasks by exploring common features and outperforms the representational sharing scheme. The knowledge captured by the meta network can be transferred across other new tasks.

In future work, we would like to investigate other functional sharing mechanisms of neural network based multi-task learning.

Acknowledgement

We would like to thank the anonymous reviewers for their valuable comments. The research work is supported by the National Key Research and Development Program of China (No. 2017YFB1002104), Shanghai Municipal Science and

References

- Bertinetto, L.; Henriques, J. F.; Valmadre, J.; Torr, P.; and Vedaldi, A. 2016. Learning feed-forward one-shot learners. In *NIPS*, 523–531.
- Blitzer, J.; Dredze, M.; Pereira, F.; et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, 440–447.
- Brazdil, P.; Carrier, C. G.; Soares, C.; and Vilalta, R. 2008. *Metalearning: Applications to data mining*. Springer Science & Business Media.
- Caruana, R. 1997. Multitask learning. *Machine learning* 28(1):41–75.
- Chen, X.; Shi, Z.; Qiu, X.; and Huang, X. 2017. Adversarial multi-criteria learning for chinese word segmentation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1193–1203. Vancouver, Canada: Association for Computational Linguistics.
- Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.
- Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.
- Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *JMLR* 12:2493–2537.
- De Brabandere, B.; Jia, X.; Tuytelaars, T.; and Van Gool, L. 2016. Dynamic filter networks. In *Neural Information Processing Systems (NIPS)*.
- Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR* 12:2121–2159.
- Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML-11*, 513–520.
- Graves, A. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.
- Ha, D.; Dai, A.; and Le, Q. V. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106*.
- Hashimoto, K.; Xiong, C.; Tsuruoka, Y.; and Socher, R. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. *Proceedings of EMNLP*.
- Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.
- Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.
- Jozefowicz, R.; Zaremba, W.; and Sutskever, I. 2015. An empirical exploration of recurrent network architectures. In *ICML-15*.
- Kalchbrenner, N.; Grefenstette, E.; and Blunsom, P. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*.
- Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.
- Lafferty, J. D.; McCallum, A.; and Pereira, F. C. N. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML-2001*.
- Lemke, C.; Budka, M.; and Gabrys, B. 2015. Metalearning: a survey of trends and technologies. *Artificial Intelligence Review* 44(1):117–130.
- Liu, X.; Gao, J.; He, X.; Deng, L.; Duh, K.; and Wang, Y.-Y. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *NAACL*.
- Liu, P.; Qiu, X.; and Huang, X. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of IJCAI*, 2873–2879.
- Liu, P.; Qiu, X.; and Huang, X. 2017. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*.
- Luong, M.-T.; Le, Q. V.; Sutskever, I.; Vinyals, O.; and Kaiser, L. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.
- Ma, X., and Hovy, E. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.
- Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the ACL*, 142–150.
- Marcus, M. P.; Marcinkiewicz, M. A.; and Santorini, B. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2):313–330.
- Pan, S. J., and Yang, Q. 2010. A survey on transfer learning. *TKDE* 22(10):1345–1359.
- Pang, B., and Lee, L. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, 115–124.
- Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. *EMNLP 2014* 12:1532–1543.
- Socher, R.; Perelygin, A.; Wu, J. Y.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.
- Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 3104–3112.
- Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.
- Zhu, X.; Sobihani, P.; and Guo, H. 2015. Long short-term memory over recursive structures. In *International Conference on Machine Learning*, 1604–1612.