Shared-private LSTM for Multi-domain Text Classification

Haiming Wu[1], Yue Zhang[2], Xi Jin[3], Yun Xue[1](✉), and Ziwen Wang[1]

[1] School of Physics and Telecommunication Engineering,
South China Normal University, Guangzhou, China
xueyun@scnu.edu.cn
[2] School of Engineering, Westlake University, Hangzhou, China
[3] Faculty of Mathematics and Statistics, Hubei University, Wuhan, China

Abstract. Shared-private models can significantly improve the performance of cross-domain learning. These methods use a shared encoder for all domains and a private encoder for each domain. One issue is that domain-specific knowledge is separately learned, without interaction with each other. We consider tackling this problem through a shared-private LSTM (SP-LSTM), which allow domain-specific parameters to be updated on a three-dimensional recurrent neural network. The advantage of SP-LSTM is that it allows domain-private information to communicate with each other during the encoding process, and it is faster than LSTM due to the parallel mechanism. Results on text classification across 16 domains indicate that SP-LSTM outperforms state-of-the-art shared-private architecture.

Keywords: Multi-task learning · Shared-private LSTM · Text classification.

1 Introduction

When faced with multiple domains datasets, multi-task learning, as an effective approach to transfer knowledge from one text domain to another [1,2,3,4,5,6,7], which can improve the performance of a single task [8], has been paid much attention by researchers. In recent years, with the rise of deep learning, the neural-based model for multi-task learning has been widely applied as a common technique in many tasks of natural language processing. Such as, sequence tagging [9,10], syntactic parsing [6,11,12,13,14], named entity recognition [7,15], text classification [8,16,17], etc. Compared with discrete representations, the neural-based model can allow efficient knowledge sharing across more than two domains.

It has been shown that, however, the vanilla multi-task learning does not always yield benefit between different NLP tasks [21,22]. And even sometimes, the optimization of one task can reduce the performance of other tasks. There has been an investigation into which tasks are more "compatible" with each other in a multi-task learning environment, therefore, the transfer learning can only be conducted between selected tasks [23,24]. Another different line of work tries to solve the problem from a modeling perspective, which is to learn common knowledge to share between tasks while keeping task-specific knowledge private [12,25,26,27].

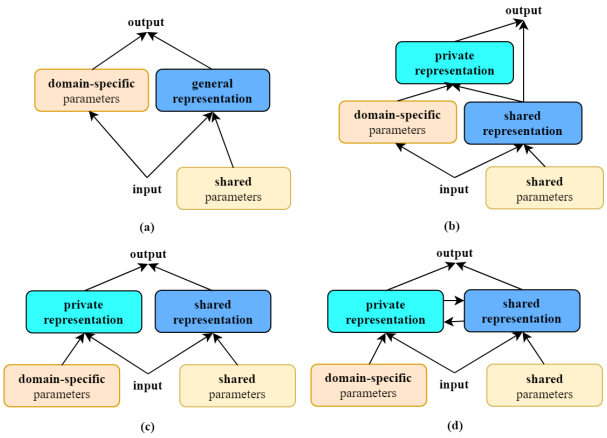

Fig. 1: Shared-private models.

As shown in Fig. 1(a), multi-domains training model [18,19] is introduced. A set of shared parameters are used as a general extractor, followed by an output layer of each domain. Given an input, a shared representation is generated, which is passed a private output layer with domain-specific parameters. And in the work of [20] (Fig. 1 (b)), a self-attention of each domain is used to learn domain descriptor vectors based on the general representation. Connecting domain descriptor vectors and the shared representation as the final sentence representation for each sentence. These models take a set of parameters for a general extractor, and a set of parameters of each domain is used to distinguish private information. However, a limitation is that they only use an output layer for each domain, and sentence vectors are from the same domain-agnostic representation, resulting in weak utilization of domain knowledge. Therefore, under the circumstances, it is hard to ensure that the shared extractor can fully extract domain knowledge, the final outputs thus are difficult to complete the task.

One state-of-the-art system for multi-task learning, shown in Fig. 1 (c), is a standard shared-private network [8] with common parameters shared across domains, as well as a set of private parameters for each domain. Given an input, both the shared and the private parameters are used for predicting the output. The set of shared parameters are regularized to remove domain-dependent information. To this end, adversarial training is typically used, by maximizing the cross-entropy in predicting the input domain with the shared parameters.

This method separates domain-specific information from domain-common information by adding model parameters. The approach, however, transfers knowledge across domains mainly through shared parameters that represent common knowledge across all domains but do not represent more fine-grained similarities across certain subsets of domains. Furthermore, multiple sets of LSTM are used, which not only has a number of model parameters that scale with the number of domains but also its computation is non-parallel because of the inherent sequential nature endows of LSTM. Accordingly, the method may be less useful when the number of domains is very large. Besides that,

these models do not dynamically model domain-specific and domain-common information inside the encoder.

As Fig. 1 (d) draws, we consider a new solution for addressing these issues, that is, by using a shared-private LSTM (SP-LSTM) to extract both the domain-specific and the shared representation simultaneously, which allows domain-private and domain-common representations dynamically interact with each other during the encoding process. Given an input, SP-LSTM can internally compute both shared and private representations owing to the shared-private paradigm [8]. The main idea is to model hidden states of all words with the domain knowledge at each recurrent step, and the state is able to get the knowledge from domain-private and domain-common. At each step, each sentence has a single state, which is composed of word-level states and two sentence-level states. And each state is updated recurrently by exchanging information between respective contexts for learning knowledge of local and non-local contexts. Therefore, all states can be computed in parallel as drawn Fig. 2.

The results on a 16-domain classification task show that our method is superior to the standard shared-private model, and gives the best accuracy compared with other methods in the literature. Our code is released at `https://github.com/haiming-wu/SP-LSTM`.

2 Related Work

Multi-domain Learning The work of jointly learning multiple domains can improve generalization proposed by [18,19]. A general encoder is used for all domain, and multiple output layers for prediction. Another work [20] adopts a general network with shared sentence representation, and multiple sets of parameters of attention to better capture domain characteristics. Our work is similar to theirs, but we have a set of shared word-level parameters for computing each word-level state, and multiple sets of domain-specific parameters are used to exchange sentence-level states. And domain-specific sentence state is used to update word-level states within a sentence.

Adversarial training Adversarial network is first used for generative model [28], and in the work [8], they take adversarial training to separate domain-common features from separate domain-specific features. A task discriminator is used to map the shared representation of sentence to a probability distribution, and estimate the task label of the sentence. Nevertheless, the shared representation from the shared encoder does not want to be identified. Formally, it is transformed into a minimax problem. Finally, the shared domain information is stored in the shared parameters that do not contain any domain-specific knowledge. In similar spirits, we adopt adversarial training to ensure the shared sentence-level parameters do not store domain-specific information.

Sentence-state LSTM Sentence-state LSTM [31](SLSTM) is an alternative recurrent neural network structure. SLSTM views the whole sentence as a single state, which consists of sub-states for individual words and a whole sentence-level state. And states can be updated recurrently by exchanging information between their contexts. SP-LSTM adopts the same calculation mechanism as SLSTM, and captures the shared and domain-specific information by setting two sentence-level states.

3 Setting

Task Input a sentence $\boldsymbol{s} = \{w_1, w_2, ..., w_n\}$, where w_i is the i-th word, n is the number of words and all word comes from a vocabulary $\boldsymbol{V}$. The goal is to build a model F, which can predict the sentiment label $y \in \{0, 1\}$ as a binary classification task.

Text Classifier Firstly, We map word w_i to an embedding x_i by a embedding matrix $\boldsymbol{R}^{|\boldsymbol{V}| \times h}$, where $|\boldsymbol{V}|$ is the size of vocabulary $\boldsymbol{V}$, h is the hidden size, the sentence is thus mapped into $\boldsymbol{x} = \{x_1, x_2, ..., x_n\}$. Then, encoder is used to represent the sequence $\boldsymbol{x}$ as a representation, which can be expressed as follows:

$$\boldsymbol{h} = \text{Encoder}(\boldsymbol{x}, \boldsymbol{\theta}) \tag{1}$$

where $\boldsymbol{h}$ denotes the representation of output, $\boldsymbol{\theta}$ denotes parameters.

Secondly, the representation $\boldsymbol{h}$ is passed into a $softmax$ classification layer:

$$\hat{y} = \text{softmax}(\boldsymbol{W}_c \boldsymbol{h} + \boldsymbol{b}_c) \tag{2}$$

where $\hat{y}$ is the prediction probability distribution, $\boldsymbol{W}_c, \boldsymbol{b}_c$ are weight and bias term, correspondingly.

Training We train the model on a specified training corpus $\boldsymbol{D}^{tra}$ by minimizing the cross-entropy between the predicted distribution $\hat{y}$ and the true distribution y:

$$L_{loss} = -\sum_{i=1}^{|\boldsymbol{D}^{tra}|} y_i \log(\hat{y}_i) \tag{3}$$

where $|\boldsymbol{D}^{tra}|$ is the size of $\boldsymbol{D}^{tra}$, y_i and $\hat{y}_i$ are distributions about i-th sample.

Multi-domain Learning Setting: Given datasets with multiple domains, multi-tasking learning attempts to investigate the correlation between these related domains to improve the performance of each dataset. Suppose there are m domains $\{\boldsymbol{D}_i\}_{i=1}^{m}$, and each of them comes from a domain i. $\boldsymbol{D}_k$ contains $|\boldsymbol{D}_k|$ data points $(\boldsymbol{s}_j^k, d_k, y_j^k)$, where $j \in \{1, 2, ..., |\boldsymbol{D}_k|\}$, $\boldsymbol{s}_j^k$ is a sequence includes $\left|\boldsymbol{s}_j^i\right|$ words $\{w_1, w_2, ..., w_{\left|\boldsymbol{s}_j^i\right|}\}$, d_k is a domain label (since we use 1 to m to number each domain, $d_k = k$) and y_j^k indicates the sentiment label (e.g. $y_j^k \in \{0, 1\}$ for binary sentiment classification). The task is to learn a classification function F trained on multiple source domain which will be used for each domain.

4 Method

In this paper, we propose a novel recurrent neural network for multi-task learning, namely, shared-private LSTM (SP-LSTM). As the name suggests that shared parameters are learned through all domains, while private parameters for their own domain. The architecture of our proposed model is illustrated in Fig. 2, which shows that the model mainly has two parts: encoding network of SP-LSTM, regularizer by adversarial training. In the following sections, we will describe each part in detail.

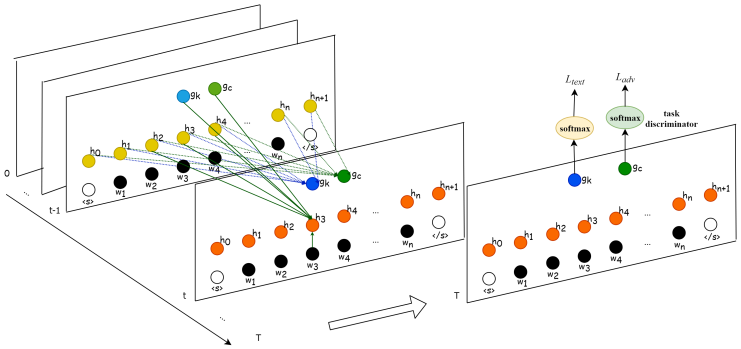

Fig. 2: Shared-private LSTM.

4.1 Shared-private LSTM

Shared-private LSTM (SP-LSTM) is designed for multi-task learning, which has three parts at each recurrent step t: hidden vectors of word-level, domain-specific and the shared sentence states. When calculating each state, word hidden state $\boldsymbol{h}_i$ extracts syntactic and semantic information for word w_i under the sentential context, private state $\boldsymbol{g}_k$ extracts knowledge for the whole sentence from domain k and shared state $\boldsymbol{g}_c$ extracts the common knowledge for all the whole sentence. Based on the previous work, we set $< s >, < /s >$ at both ends of each sentence. The hidden state at time step t thus can be denoted by:

$$\boldsymbol{H}^t = \langle \boldsymbol{h}_0^t, \boldsymbol{h}_1^t, ..., \boldsymbol{h}_{n+1}^t, \boldsymbol{g}_k^t, \boldsymbol{g}_c^t \rangle \tag{4}$$

where $\boldsymbol{h}_i^t$ denotes the hidden state for each word w_i and $\boldsymbol{g}_k^t, \boldsymbol{g}_c^t$ are sentence hidden states, the former denotes domain-specific for domain k, the latter is common representation.

SP-LSTM updates all states with a recurrent state transition process, which enriches the state representations incrementally. Then we set $\boldsymbol{h}_i^0 = x_i$, and $\boldsymbol{g}_k^0, \boldsymbol{g}_c^0$ all are the average of $\boldsymbol{h}^0$. At the recurrent step from time $t-1$ to t, the state transitions of word-level from $\boldsymbol{h}_i^{t-1}$ to $\boldsymbol{h}_i^t$, and the exchange of the sentence-level from $\boldsymbol{g}_k^{t-1}$ to $\boldsymbol{g}_k^t$ and $\boldsymbol{g}_c^{t-1}$ to $\boldsymbol{g}_c^t$. Besides that, current cell $\boldsymbol{c}_i^t, \boldsymbol{c}_k^t, \boldsymbol{c}_c^t$ are used to complete this transition of $w_i, \boldsymbol{g}^k, \boldsymbol{g}^c$ respectively as simple as LSTM.

Word-level hidden As shown in Fig. 2, the solid green lines identify that each hidden vector $\boldsymbol{h}_i^t$ is computed based on the vector of x_i, the hidden vectors $\boldsymbol{h}_{i-1}^{t-1}, \boldsymbol{h}_i^{t-1}, \boldsymbol{h}_{i+1}^{t-1}$ under the sentential context and sentence hidden vectors $\boldsymbol{g}_k^{t-1}, \boldsymbol{g}_c^{t-1}$. At the same time, seven gates are utilized to balance the knowledge: $\boldsymbol{i}_i^t$ is a input gate to restrict the original information of w_i, $\boldsymbol{l}_i^t, \boldsymbol{f}_i^t, \boldsymbol{r}_i^t$ are gates that control information flow from hidden vectors $\boldsymbol{h}_{i-1}^{t-1}, \boldsymbol{h}_i^{t-1}, \boldsymbol{h}_{i+1}^{t-1}$. $\boldsymbol{k}_i^t$ is a gate controlling information flow from sentence hidden vector $\boldsymbol{g}_k^{t-1}$ of domain k. $\boldsymbol{s}_i^t$ is a gate that controls information flow from $\boldsymbol{g}_c^{t-1}$, which denotes the influence of common information on $\boldsymbol{h}_i^t$. $\boldsymbol{o}_i^t$ is an output gate from the cell state $\boldsymbol{c}_i^t$ to hidden state $\boldsymbol{h}_i^t$.

The process can be formally expressed as follows:

$$
\begin{aligned}
\boldsymbol{\xi}_i^t &= [\boldsymbol{h}_{i-1}^{t-1}, \boldsymbol{h}_i^{t-1}, \boldsymbol{h}_{i+1}^{t-1}] \\
\begin{bmatrix} \hat{\boldsymbol{i}}_i^t \\ \hat{\boldsymbol{l}}_i^t \\ \hat{\boldsymbol{f}}_i^t \\ \hat{\boldsymbol{r}}_i^t \\ \hat{\boldsymbol{k}}_i^t \\ \hat{\boldsymbol{s}}_i^t \\ \hat{\boldsymbol{o}}_i^t \end{bmatrix} &= \sigma \left(\begin{bmatrix} \boldsymbol{W}_i \\ \boldsymbol{W}_l \\ \boldsymbol{W}_f \\ \boldsymbol{W}_r \\ \boldsymbol{W}_k \\ \boldsymbol{W}_s \\ \boldsymbol{W}_o \end{bmatrix} \boldsymbol{\xi}_i^t + \begin{bmatrix} \boldsymbol{O}_i \\ \boldsymbol{O}_l \\ \boldsymbol{O}_f \\ \boldsymbol{O}_r \\ \boldsymbol{O}_k \\ \boldsymbol{O}_s \\ \boldsymbol{O}_o \end{bmatrix} \boldsymbol{x}_i + \begin{bmatrix} \boldsymbol{U}_i \\ \boldsymbol{U}_l \\ \boldsymbol{U}_f \\ \boldsymbol{U}_r \\ \boldsymbol{U}_k \\ \boldsymbol{U}_s \\ \boldsymbol{U}_o \end{bmatrix} \boldsymbol{g}_k^{t-1} + \begin{bmatrix} \boldsymbol{V}_i \\ \boldsymbol{V}_l \\ \boldsymbol{V}_f \\ \boldsymbol{V}_r \\ \boldsymbol{V}_k \\ \boldsymbol{V}_s \\ \boldsymbol{V}_o \end{bmatrix} \boldsymbol{g}_c^{t-1} + \begin{bmatrix} \boldsymbol{b}_i \\ \boldsymbol{b}_l \\ \boldsymbol{b}_f \\ \boldsymbol{b}_r \\ \boldsymbol{b}_k \\ \boldsymbol{b}_s \\ \boldsymbol{b}_o \end{bmatrix} \right) \\
\boldsymbol{u}_i^t &= tanh(\boldsymbol{W}_u \boldsymbol{\xi}_i^t + \boldsymbol{O}_u \boldsymbol{x}_i + \boldsymbol{U}_u \boldsymbol{g}_k^{t-1} + \boldsymbol{V}_u \boldsymbol{g}_c^{t-1} + \boldsymbol{b}_u) \\
\boldsymbol{i}_i^t, \boldsymbol{l}_i^t, \boldsymbol{f}_i^t, \boldsymbol{r}_i^t, \boldsymbol{k}_i^t, \boldsymbol{s}_i^t &= softmax(\hat{\boldsymbol{i}}_i^t, \hat{\boldsymbol{l}}_i^t, \hat{\boldsymbol{f}}_i^t, \hat{\boldsymbol{r}}_i^t, \hat{\boldsymbol{k}}_i^t, \hat{\boldsymbol{s}}_i^t) \\
\boldsymbol{c}_i^t &= \boldsymbol{l}_i^t \odot \boldsymbol{c}_{i-1}^{t-1} + \boldsymbol{f}_i^t \odot \boldsymbol{c}_i^{t-1} + \boldsymbol{r}_i^t \odot \boldsymbol{c}_{i+1}^{t-1} + \boldsymbol{k}_i^t \odot \boldsymbol{c}_k^{t-1} \\
&\quad + \boldsymbol{s}_i^t \odot \boldsymbol{c}_c^{t-1} + \boldsymbol{i}_i^t \odot \boldsymbol{u}_i^{t-1} \\
\boldsymbol{h}_i^t &= \boldsymbol{o}_i^t \odot tanh(c_i^t)
\end{aligned}
\tag{5}
$$

where $\boldsymbol{\xi}_i^t$ is the concatenation of hidden vectors of a context window. The values of $\boldsymbol{i}_i^t, \boldsymbol{l}_i^t, \boldsymbol{f}_i^t, \boldsymbol{r}_i^t, \boldsymbol{k}_i^t$ and $\boldsymbol{c}_i^t$ are normalized so that the sum of them is 1. $\boldsymbol{\theta}_w = \{\boldsymbol{W}_x, \boldsymbol{U}_x, \boldsymbol{V}_x, \boldsymbol{b}_x\}(x \in \{i, l, f, r, k, s, u, o\})$ are model parameters that will be shared by all domains. σ is the sigmoid function.

Note that the representation $\boldsymbol{g}_k^{t-1}$ of private is used to update the state of each word, so that domain-specific knowledge is allowed to be captured during encoding. Furthermore, all the parameters $\boldsymbol{\theta}_w$ are shared across all domains.

Domain-specific Sentence State Thereafter, $\boldsymbol{g}_k^t, \boldsymbol{g}_c^t$ are domain-private and shared hidden representation. As Fig. 2 draws, blue dashed lines denote the calculation process from $\boldsymbol{g}_k^{t-1}$ to $\boldsymbol{g}_c^{t-1}$. Besides that, the values of $\boldsymbol{g}_k^t, \boldsymbol{g}_c^t$ both are computed based on the values of $\boldsymbol{h}_i^{t-1}$ for all $i \in [0, n+1]$. The formulas are formally presented as follows:

$$
\begin{aligned}
\bar{\boldsymbol{h}} &= avg(\boldsymbol{h}_0^{t-1}, \boldsymbol{h}_1^{t-1}, ..., \boldsymbol{h}_{n+1}^{t-1}) \\
\hat{\boldsymbol{f}}_{ki}^t &= \sigma(\boldsymbol{W}_{fk} \boldsymbol{g}_k^{t-1} + \boldsymbol{U}_{fk} \boldsymbol{h}_i^{t-1} + \boldsymbol{b}_{fk}) \\
\left[\hat{\boldsymbol{i}}_k^t, \hat{\boldsymbol{o}}_k^t\right]^\mathsf{T} &= \sigma(\left[\boldsymbol{W}_{ik}, \boldsymbol{W}_{ok}\right]^\mathsf{T} \boldsymbol{g}_k^{t-1} + \left[\boldsymbol{U}_{ik}, \boldsymbol{U}_{ok}\right]^\mathsf{T} \bar{\boldsymbol{h}} + \left[\boldsymbol{b}_{ik}, \boldsymbol{b}_{ok}\right]^\mathsf{T}) \\
\boldsymbol{f}_{k0}^t, ..., \boldsymbol{f}_{k(n+1)}^t, \boldsymbol{i}_k^t &= \text{softmax}(\hat{\boldsymbol{f}}_{k0}^t, ..., \hat{\boldsymbol{f}}_{k(n+1)}^t, \hat{\boldsymbol{i}}_k^t) \\
\boldsymbol{c}_k^t &= \boldsymbol{i}_k^t \odot \boldsymbol{c}_k^{t-1} + \sum_{i=0}^{n+1} \boldsymbol{f}_{ki}^t \odot \boldsymbol{c}_i^{t-1} \\
\boldsymbol{g}_k^t &= \boldsymbol{o}_k^t \odot \tanh(\boldsymbol{c}_k^t)
\end{aligned}
\tag{6}
$$

where $\boldsymbol{f}_{k0}^t, \boldsymbol{f}_{k1}^t, ..., \boldsymbol{f}_{k(n+1)}^t$ and $\boldsymbol{i}_k^t$ are gates that control the information from $\boldsymbol{c}_0^{t-1}, \boldsymbol{c}_1^{t-1}$, ..., $\boldsymbol{c}_{n+1}^{t-1}$ and $\boldsymbol{c}_k^{t-1}$, respectively, and they are normalized. $\boldsymbol{o}_k^t$ is an output gate of $\boldsymbol{g}_k^t$. The parameters $\boldsymbol{\theta}_k = \{W_{xk}, U_{xk}, b_{xk}, x \in \{i, f\}\}$ are private of the domain k.

Shared Sentence State Subsequent to that, $\boldsymbol{g}_c^t$ is calculated with the calculation process is similar to that of $\boldsymbol{g}_k^t$, which denoted by green dashed lines of Fig.2.

$$\begin{aligned}
\hat{\boldsymbol{f}}_{ci}^t &= \sigma(\boldsymbol{W}_{fc}\boldsymbol{g}_c^{t-1} + \boldsymbol{U}_{fc}\boldsymbol{h}_i^{t-1} + \boldsymbol{b}_{fc}) \\
\left[\hat{\boldsymbol{i}}_c^t, \hat{\boldsymbol{o}}_c^t\right]^\top &= \sigma(\left[\boldsymbol{W}_{ic}, \boldsymbol{W}_{oc}\right]^\top \boldsymbol{g}_c^{t-1} + \left[\boldsymbol{U}_{ic}, \boldsymbol{U}_{oc}\right]^\top \bar{\boldsymbol{h}} + \left[\boldsymbol{b}_{ic}, \boldsymbol{b}_{oc}\right]^\top) \\
\boldsymbol{f}_{c0}^t, ..., \boldsymbol{f}_{c(n+1)}^t, \boldsymbol{i}_c^t &= \text{softmax}(\hat{\boldsymbol{f}}_{c0}^t, ..., \hat{\boldsymbol{f}}_{c(n+1)}^t, \hat{\boldsymbol{i}}_c^t) \\
\boldsymbol{c}_c^t &= \boldsymbol{i}_c^t \odot \boldsymbol{c}_c^{t-1} + \sum_{i=0}^{n+1} \boldsymbol{f}_{ci}^t \odot \boldsymbol{c}_i^{t-1} \\
\boldsymbol{g}_c^t &= \boldsymbol{o}_c^t \odot \tanh(\boldsymbol{c}_c^t)
\end{aligned} \tag{7}$$

where $\boldsymbol{f}_{c0}^t, ..., \boldsymbol{f}_{c(n+1)}^t$ and $\boldsymbol{i}_c^t$ are gates, and they are normalized. $\boldsymbol{o}_c^t$ is the output gate for $\boldsymbol{g}_c^t$. $\boldsymbol{\theta}_c = \{W_{xc}, U_{xc}, b_{xc}, x \in \{i, f\}\}$ are model parameters shared with all domains.

At last, we take the tuple $(\boldsymbol{g}_k^T, \boldsymbol{g}_c^T)$ as the output of SP-LSTM at the last time step T. Then, we adopt SP-LSTM to encode a sequence $\boldsymbol{x}_j^k$, which is described as follows:

$$(\boldsymbol{h}_j^k, \boldsymbol{s}_j^k) = (\boldsymbol{g}_k^T, \boldsymbol{g}_c^T)_j = \text{SP-LSTM}(\boldsymbol{x}_j^k, \boldsymbol{\theta}_w, \boldsymbol{\theta}_k, \boldsymbol{\theta}_c) \tag{8}$$

where $\boldsymbol{h}_j^k$ and $\boldsymbol{s}_j^k$ are the final representations of domain-specific and the domain-common.

The difference between SP-LSTM and other RNNs (LSTM, Bi-LSTM, GRU) is due to their different recurrent states. The traditional model RNNs only uses one state to represent the sequence from the beginning to a certain word, While SP-LSTM uses a structural state to represent the full sentence, which consists of two sentence-level states and $n + 2$ word-level states, simultaneously. Different from RNNs, $\boldsymbol{h}_i^t$ is used to represent w_i at each step t. As t increases from 0, each word-level and sentence-level states are enriched with increasingly deeper context information.

From the perspective of multi-task learning, RNNs transfers knowledge from one end of the sentence to the other. Accordingly, on the one hand, the number of time steps scales with the size of the input, and on the other hand, RNNs can not capture shared and private knowledge simultaneously. In contrast, SP-LSTM we proposed not only allows bi-directional information flow at each word simultaneously, but also allows it between sentence-level state and every word-level state. At each step, each h_i captures an increasingly larger n-gram context, and they will be used for communication between g_k and g_c. So that SP-LSTM is able to dynamically capture the shared and private knowledge. Then, the number of recurrent steps of SP-LSTM is decided by the end-task performance rather than the sentence size, and it is more suitable for the representation of multi-domain.

4.2 Regularizer: ADV

Adversarial Training Inspired by the generative adversarial networks (GAN) [28], we incorporate adversarial networks as a regularizer to maintain the common information which is stored by common parameters $\boldsymbol{\theta}_c$.

In theory, GAN is used to simulate a distribution $P_{g(x)}$ that is similar to the real data distribution $P_{data(x)}$, GAN thus builds a generative network G and a discriminative model D that check whether the input is real data, and then, the generative network G will be trained to generate better samples that are similar to the real data, which indicates that the data distribution $P_{g(x)}$ generated by the generative network tends to be consistent with the distribution $P_{data(x)}$ of the real data. It is formally expressed as:

$$L_{Adv} = \min_{\theta_G} \max_{\theta_D} (E_{x\ P_{data}}[\log D(x)] + E_{z\ p(z)}[\log(1 - D(G(z))]) \tag{9}$$

where θ_G, θ_D are parameters of generative network and discriminative model respectively. From the optimization formula, the negative cross-entropy is used as a loss function. And the aim of G is generating a sample that D do not know whether it is real data, so the output from the maximum optimizer is minimized.

Therefore, adversarial training can be applied in our model to optimize parameters $\boldsymbol{\theta}_c$ and $\boldsymbol{\theta}_k$. So a $softmax$ function is used as task discriminator to estimate which domain the input data comes from. Formally:

$$D(\boldsymbol{s}_j^k, \boldsymbol{\theta}_D) = \text{softmax}(\boldsymbol{W}_D \boldsymbol{s}_j^k + \boldsymbol{b}_D) \tag{10}$$

where $D(\boldsymbol{s}_j^k, \theta_D)$ is the prediction probability distribution of domain label, $\boldsymbol{\theta}_D = \{\boldsymbol{w}_D, \boldsymbol{b}_D\}$ are model parameters.

Then, we define an adversarial loss L_{adv} with minimax optimization function:

$$L_{adv} = -\max_{\theta_c} (\min_{\theta_D} \sum_{k=1}^{m} \sum_{j=1}^{|D_k^{tra}|} d_k \log[D(\boldsymbol{s}_j^k, \theta_D)]) \tag{11}$$

where L_{adv} is minimized the cross-entropy of domain label to fine-tune discriminative model, but maximized for $\boldsymbol{\theta}_c$ of the common model parameters. In our implementation, we use the control gradient method to complete the above calculation.

The Loss Function of Model Considering that knowledge of each sentence is distributed between the shared and private representations, so the private vectors are joined by the common one as the final representation. Then, we take a $softmax$ function to predict the probability distribution $\hat{y}_j^k$ of input $\boldsymbol{s}_j^k$ likes Equation (2):

$$\hat{y}_j^k = \text{softmax}(\boldsymbol{W}_{tk}[\boldsymbol{h}_j^k, \boldsymbol{s}_j^k] + \boldsymbol{b}_{tk}) \tag{12}$$

The loss L_{text} of text classification can be computed as:

$$L_{text} = -\sum_{k=1}^{m} \sum_{j=1}^{|\boldsymbol{D}_k^{tra}|} y_j^k \log(\hat{y}_j^k) \tag{13}$$

where, y_j^k denotes the real probability distribution.

To sum up, the training of L_{text} is straightforward, its goal is to perform better within its own domain. The adversarial loss L_{adv} has two targets: not only helps the

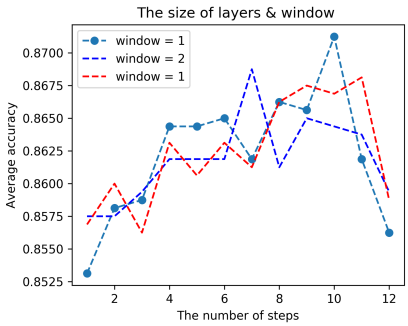

Fig. 3: Average accuracies with various window sizes and steps on four review development datas

main task achieve higher accuracy, but also keeps the common features away from across private features of each domain. Thus, the total loss function as:

$$L = L_{text} + \lambda L_{adv} \tag{14}$$

where λ is the hyper-parameter of the regularizer L_{adv}.

5 Experiments

5.1 Datasets and Settings

Datasets. In this experiment, we compare the SP-LSTM with state-of-the-art multi-task methods of text classification on FDU-MTL [8]. FDU-MTL is composed of reviews from different domains, in which there are 14 Amazon domains and two movies review domains from the IMDB and the MR datasets. And we use the original split [8]. All experiments are conducted using a GeForce GTX 1080ti GPU with 11GB memory.

Hyperparameters Glove 200-dimensional embeddings are adopted to initialize the word embeddings for all of the models, and embeddings are fine-tuned during model training for all tasks. Dropout [29] is applied to embedding hidden states and the output of all encoder model, with a rate of 0.7. The Adam optimizer [30] is selected for all models during the training process, with an initial learning rate of 0.0005. The batch size is set to 4. The hyper-parameters λ and β are 0.05.

5.2 Development Experiments

We choose four development data (MR, books, camera, magazines) to investigate the effect with different configurations of SP-LSTMs on average accuracies. Fig. 3 draws average accuracies of SP-LSTMs with three kinds of window sizes against the number of recurrent steps. As can be seen from Fig. 3, when the number of steps increases from 1 to 12, accuracies are subject to certain fluctuations, and generally increase before

Table 1: Results on the FDU-MTL dataset: the training corpus contains all domains

Model	LSTM +DO	LSTM +ADV	LSTM+ ADV+Diff	DSAM	SP-LSTM+ ADV+Diff	SP-LSTM	SP-SLSTM +ADV
Apparel	83.50	86.00	87.25	85.00	88.00	**89.50**	88.75
Baby	86.75	88.00	87.00	90.00	89.75	89.75	**90.75**
Books	85.75	84.50	86.25	84.50	88.25	88.50	**89.00**
Camera	89.25	89.00	88.25	89.50	91.00	91.00	**91.50**
Electronics	84.75	85.75	85.75	86.25	88.25	87.75	**89.75**
DVD	85.25	85.75	86.00	87.25	87.50	88.50	**88.75**
Health	88.75	88.00	87.50	89.25	87.25	**90.75**	90.00
IMDB	85.25	83.00	86.25	86.75	**87.00**	85.75	85.75
Kitchen	84.25	85.50	87.25	90.00	**90.75**	88.50	88.50
Magazines	90.50	92.25	93.25	93.50	**94.25**	94.00	94.00
MR	74.75	74.75	74.75	**77.25**	76.50	74.00	76.25
Music	82.75	84.25	83.25	82.25	85.25	84.75	**85.25**
Software	86.50	87.50	83.75	86.75	90.00	89.75	**90.75**
Sports	86.50	85.50	86.00	89.25	**90.25**	89.50	90.00
Toys	86.00	85.75	89.75	89.25	89.00	**91.00**	88.75
Video	84.00	85.50	83.75	87.00	**89.50**	88.75	88.75
Avg_Acc.	85.28	85.69	86.00	87.11	88.28	88.23	**88.53**

reaching a maximum value, and then shows a downward trend. It indicates that the effectiveness of recurrent information exchange in SP-LSTM state transition.

Besides, there are no significant differences in the peak accuracies given by different window sizes. Considering efficiency, we choose a window size of 1 and set the number of recurrent steps to 10 according to Fig. 3.

5.3 Results & Analysis

Table 1 shows the final results using different multi-task learning strategies. LSTM+DO denotes the model adopts a general encoder with LSTM, and multiple output layers for prediction. We reproduce it with an average accuracy of 85.28%.

LSTM+ADV denotes the adversarial shared-private model [8]. they use multiple LSTM to encode sentence from different domains, adversarial training (ADV) is adopted to model parameters. And LSTM-ADV-Diff is their model by adding orthogonality constraints (Diff) that maximizes the difference between shared and private parameters. Our implementation of LSTM-ADV+Diff gives an averaged accuracy of 86.00%, which is comparable to 86.10% reported by themselves.

MSAM denotes the method [20] of learning domain descriptor vectors for multi-domain training. Which learns a general sentence-level vector by Bi-LSTM, and adopts self-attention to learn a domain-specific descriptor vectors. Then, the connecting of the general and domain-specific vectors is passed domain-specific output layers. We rework it with our dataset, and an average accuracy of 87.11% is given.

SP-LSTM+ADV+Diff represents the model of [8], but instead of multiple LSTM, it encodes the sentence only with an SP-LSTM we proposed, which adds orthogonality constraints as regularizer. As can be seen from Table 1, the average accuracy of 88.28% is given, which significantly outperforms LSTM+ADV+Diff, indicating that SP-LSTM is better at encoding in multi-task learning environment. Therefore, We take this method as our baseline.

SP-LSTM denotes our method, which is utilized to represent sentence and give the shared and domain-specific vectors. It gives an average accuracy of 88.23%, which

significantly outperforms existing methods. This shows that SP-LSTM is suitable for the encoding of multi-domain.

By further adding the adversarial training regularizer, the performance of SP-LSTM+ ADV increases to 88.53%, which shows the effectiveness of adversarial training under our framework. However, the orthogonality constraints are not still useful since the average accuracy of SP-LSTM+ADV+Diff model is 88.28%, which indicates that SP-LSTM and orthogonality constrains are incompatible, and Diff can not help improve the performance of SP-LSTM. Meanwhile, SP-LSTM+ADV gives the best reported results in the literature.

In addition, the advantage of our method is that it runs faster. Under the same model parameters as LSTM+ADV, the test time of our model was 8.30 seconds, while that of LSTM+ADV was 9.62 seconds.

6 Conclusion

We investigate SP-LSTM for multi-domain text classification, which separates the domain-specific and domain-common knowledge by using a set of shared word-level parameters to encode word hidden states with domain knowledge, and using a set of shared sentence-level parameters to extract the common information, using a set of domain-specific parameters for storing private knowledge. Results on 16 domains show that our method significantly outperforms traditional shared-private architectures for transfer learning.

7 Acknowledgments

The work described in this paper was initiated and partly done when Haiming Wu was visiting Westlake University. It is supported by the National Statistical Science Research Project of China under Grant No. 2016LY98, the National Natural Science Foundation of China under Grant No. 61876205, the Science and Technology Department of Guangdong Province in China under Grant Nos. 2016A010101020, 2016A010101021 and 2016A010101022, the Science and Technology Plan Project of Guangzhou under Grant Nos. 201802010033 and 201804010433.

References

1. Daume III H, Marcu D. Domain adaptation for statistical classifiers[J]. Journal of artificial Intelligence research, 2006, 26: 101-126.
2. Daumé III H. Frustratingly easy domain adaptation[J]. arXiv preprint arXiv:0907.1815, 2009.
3. Blitzer J, McDonald R, Pereira F. Domain adaptation with structural correspondence learning[C]//In Proc.EMNLP, 2006: 120-128.
4. Chen M, Weinberger K Q, Blitzer J. Co-training for domain adaptation[C]//Advances in neural information processing systems. 2011: 2456-2464.
5. Blitzer J, Foster D P, Kakade S M. Domain adaptation with coupled subspaces[J]. 2011.
6. McDonald R, Petrov S, Hall K. Multi-source transfer of delexicalized dependency parsers[C]//In Proc.EMNLP, 2011: 62-72.

7. Lin B Y, Lu W. Neural adaptation layers for cross-domain named entity recognition[J]. arXiv preprint arXiv:1810.06368, 2018.
8. Liu P, Qiu X, Huang X. Adversarial multi-task learning for text classification[J]. arXiv preprint arXiv:1704.05742, 2017.
9. Alonso H M, Plank B. When is multitask learning effective? Semantic sequence prediction under varying data conditions[J]. arXiv preprint arXiv:1612.02251, 2016.
10. Ruder12 S, Bingel J, Augenstein I, et al. Sluice networks: Learning what to share between loosely related tasks[J]. stat, 2017, 1050: 23.
11. McDonald R, Petrov S, Hall K. Multi-source transfer of delexicalized dependency parsers[C]//In Proc.EMNLP, 2011: 62-72.
12. Braud C, Plank B, Søgaard A. Multi-view and multi-task training of RST discourse parsers[C]//In Proc.COLING 2016, 2016: 1903-1913.
13. Rasooli M S, Tetreault J. Yara parser: A fast and accurate dependency parser[J]. arXiv preprint arXiv:1503.06733, 2015.
14. Goodman J, Vlachos A, Naradowsky J. Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing[C]//In Proc.ACL, 2016, 1: 1-11.
15. Cao P, Chen Y, Liu K, et al. Adversarial Transfer Learning for Chinese Named Entity Recognition with Self-Attention Mechanism[C]//In Proc.EMNLP, 2018: 182-192.
16. Li S, Zong C. Multi-domain sentiment classification[C]//In Proc.ACL, 2008: 257-260.
17. Chen X, Cardie C. Multinomial adversarial networks for multi-domain text classification[J]. arXiv preprint arXiv:1802.05694, 2018.
18. Liu P, Qiu X, Huang X. Recurrent neural network for text classification with multi-task learning[J]. arXiv preprint arXiv:1605.05101, 2016.
19. Nam H, Han B. Learning multi-domain convolutional neural networks for visual tracking[C]//In Proc.CVPR. 2016: 4293-4302.
20. Liu Q, Zhang Y, Liu J. Learning domain representation for multi-domain sentiment classification[C]//In Proc.ACL, Volume 1 (Long Papers). 2018: 541-550.
21. Mou L, Meng Z, Yan R, et al. How transferable are neural networks in nlp applications?[J]. arXiv preprint arXiv:1603.06111, 2016.
22. Bingel J, Søgaard A. Identifying beneficial task relations for multi-task learning in deep neural networks[J]. arXiv preprint arXiv:1702.08303, 2017.
23. Bollmann M, Søgaard A, Bingel J. Multi-task learning for historical text normalization: Size matters[C]//Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP. 2018: 19-24.
24. Augenstein I, Ruder S, Søgaard A. Multi-task learning of pairwise sequence classification tasks over disparate label spaces[J]. arXiv preprint arXiv:1802.09913, 2018.
25. Shi P, Teng Z, Zhang Y. Exploiting mutual benefits between syntax and semantic roles using neural network[C]//In Proc.EMNLP. 2016: 968-974.
26. Søgaard A, Goldberg Y. Deep multi-task learning with low level tasks supervised at lower layers[C]//In Proc.ACL. 2016, 2: 231-235.
27. Zhang M, Zhang Y, Fu G. End-to-end neural relation extraction with global optimization[C]//In Proc.EMNLP, 2017: 1730-1740.
28. Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]//Advances in neural information processing systems. 2014: 2672-2680.
29. Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. The Journal of Machine Learning Research, 2014, 15(1): 1929-1958.
30. Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.
31. Zhang Y, Liu Q, Song L. Sentence-state lstm for text representation[J]. arXiv preprint arXiv:1805.02474, 2018.