Deep Multi-Task Learning with Shared Memory

Pengfei Liu Xipeng Qiu* Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{pfliu14,xpqiu,xjhuang}@fudan.edu.cn

Abstract

Neural network based models have achieved impressive results on various specific tasks. However, in previous works, most models are learned separately based on single-task supervised objectives, which often suffer from insufficient training data. In this paper, we propose two deep architectures which can be trained jointly on multiple related tasks. More specifically, we augment neural model with an external memory, which is shared by several tasks. Experiments on two groups of text classification tasks show that our proposed architectures can improve the performance of a task with the help of other related tasks.

1 Introduction

Neural network based models have been shown to achieved impressive results on various NLP tasks rivaling or in some cases surpassing traditional models, such as text classification (Kalchbrenner et al., 2014; Socher et al., 2013; Liu et al., 2015a), semantic matching (Hu et al., 2014; Liu et al., 2016a), parser (Chen and Manning, 2014) and machine translation (Bahdanau et al., 2014).

Usually, due to the large number of parameters these neural models need a large-scale corpus. It is hard to train a deep neural model that generalizes well with size-limited data, while building the large scale resources for some NLP tasks is also a challenge. To overcome this problem, these models often involve an unsupervised pre-training phase. The final model is fine-tuned on specific task with respect to a supervised training criterion. However, most pre-training methods are based on unsupervised objectives (Collobert et al., 2011; Turian et al., 2010; Mikolov et al., 2013), which is effective to improve the final performance, but it does not directly optimize the desired task.

Multi-task learning is an approach to learn multiple related tasks simultaneously to significantly improve performance relative to learning each task independently. Inspired by the success of multi-task learning (Caruana, 1997), several neural network based models (Collobert and Weston, 2008; Liu et al., 2015b) are proposed for NLP tasks, which utilized multi-task learning to jointly learn several tasks with the aim of mutual benefit. The characteristic of these multi-task architectures is they share some lower layers to determine common features. After the shared layers, the remaining layers are split into multiple specific tasks.

In this paper, we propose two deep architectures of sharing information among several tasks in multi-task learning framework. All the related tasks are integrated into a single system which is trained jointly. More specifically, inspired by Neural Turing Machine (NTM) (Graves et al., 2014) and memory network (Sukhbaatar et al., 2015), we equip task-specific long short-term memory (LSTM) neural network (Hochreiter and Schmidhuber, 1997) with an external shared memory. The external memory has capability to store long term information and knowledge shared by several related tasks. Different with NTM, we use a deep fusion strategy to integrate the information from the external memory into task-specific LSTM, in which a fusion gate controls the

* Corresponding author.

information flowing flexibly and enables the model to selectively utilize the shared information.

We demonstrate the effectiveness of our architectures on two groups of text classification tasks. 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 proposed a generic multi-task framework, in which different tasks can share information by an external memory and communicate by a reading/writing mechanism. Two proposed models are complementary to prior multi-task neural networks.
- Different with Neural Turing Machine and memory network, we introduce a deep fusion mechanism between internal and external memories, which helps the LSTM units keep them interacting closely without being conflated.
- As a by-product, the fusion gate enables us to better understand how the external shared memory helps specific task.

2 Neural Memory Models for Specific Task

In this section, we briefly describe LSTM model, and then propose an external memory enhanced LSTM with deep fusion.

2.1 Long Short-term Memory

Long short-term memory network (LSTM) (Hochreiter and Schmidhuber, 1997) is a type of recurrent neural network (RNN) (Elman, 1990), and specifically addresses the issue of learning long-term dependencies. LSTM maintains an internal memory cell that updates and exposes its content only when deemed necessary.

Architecturally speaking, the memory state and output state are explicitly separated by activation gates (Wang and Cho, 2015). However, the limitation of LSTM is that it lacks a mechanism to index its memory while writing and reading (Danihelka et al., 2016).

While there are numerous LSTM variants, here we use the LSTM architecture used by (Jozefowicz et al., 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 $\mathbb{R}^d$: 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 precisely specified as follows.

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

$$\mathbf{c}_t = \tilde{\mathbf{c}}_t \odot \mathbf{i}_t + \mathbf{c}_{t-1} \odot \mathbf{f}_t, \quad (2)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \quad (3)$$

where $\mathbf{x}_t \in \mathbb{R}^m$ is the input at the current time step; $\mathbf{W} \in \mathbb{R}^{4h \times (d+m)}$ and $\mathbf{b}_p \in \mathbb{R}^{4h}$ 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:

$$(\mathbf{h}_t, \mathbf{c}_t) = \mathbf{LSTM}(\mathbf{h}_{t-1}, \mathbf{c}_{t-1}, \mathbf{x}_t, \theta_p). \quad (4)$$

Here, the function $\mathbf{LSTM}(\cdot, \cdot, \cdot, \cdot)$ is a shorthand for Eq. (1-3), and θ_p represents all the parameters of LSTM.

2.2 Memory Enhanced LSTM

LSTM has an internal memory to keep useful information for specific task, some of which may be beneficial to other tasks. However, it is non-trivial to share information stored in internal memory.

Recently, there are some works to augment LSTM with an external memory, such as neural Turing machine (Graves et al., 2014) and memory network (Sukhbaatar et al., 2015), called memory enhanced LSTM (ME-LSTM). These models enhance the low-capacity internal memory to have a capability of modelling long pieces of text (Andrychowicz and Kurach, 2016).

Inspired by these models, we introduce an external memory to share information among several tasks. To better control shared information and understand how it is utilized from external memory, we propose a deep fusion strategy for ME-LSTM.

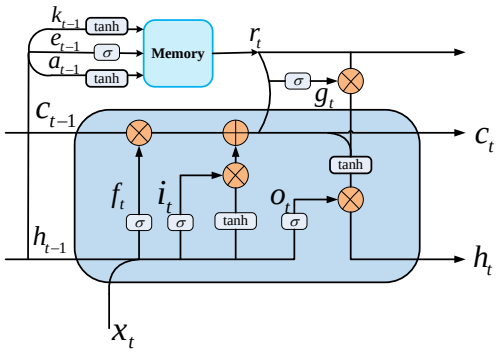


Figure 1: Graphical illustration of the proposed ME-LSTM unit with deep fusion of internal and external memories.

As shown in Figure 1, ME-LSTM consists the original LSTM and an external memory which is maintained by reading and writing operations. The LSTM not only interacts with the input and output information but accesses the external memory using selective read and write operations.

The external memory and corresponding operations will be discussed in detail below.

External Memory The form of external memory is defined as a matrix $\mathbf{M} \in \mathbb{R}^{K \times M}$, where K is the number of memory segments, and M is the size of each segment. Besides, K and M are generally instance-independent and pre-defined as hyper-parameters.

At each step t, LSTM emits output $\mathbf{h}_t$ and three key vectors $\mathbf{k}_t$, $\mathbf{e}_t$ and $\mathbf{a}_t$ simultaneously. $\mathbf{k}_t$, $\mathbf{e}_t$ and $\mathbf{a}_t$ can be computed as

$$\begin{bmatrix} \mathbf{k}_t \\ \mathbf{e}_t \\ \mathbf{a}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \tanh \end{bmatrix} (\mathbf{W}_m \mathbf{h}_t + \mathbf{b}_m) \tag{5}$$

where $\mathbf{W}_m$ and $\mathbf{b}_m$ are parameters of affine transformation.

Reading The read operation is to read information $\mathbf{r}_t \in \mathbb{R}^M$ from memory $\mathbf{M}_{t-1}$.

$$\mathbf{r}_t = \alpha_t \mathbf{M}_{t-1}, \tag{6}$$

where $\mathbf{r}_t$ denotes the reading vector and $\alpha_t \in \mathbb{R}^K$ represents a distribution over the set of segments of memory $\mathbf{M}_{t-1}$, which controls the amount of information to be read from and written to the memory.

Each scalar $\alpha_{t,k}$ in attention distribution α_t can be obtained as:

$$\alpha_{t,k} = \text{softmax}(g(\mathbf{M}_{t-1,k}, \mathbf{k}_{t-1})) \tag{7}$$

where $\mathbf{M}_{t-1,k}$ represents the k-th row memory vector, and $\mathbf{k}_{t-1}$ is a key vector emitted by LSTM.

Here $g(\mathbf{x}, \mathbf{y})$ $(\mathbf{x} \in \mathbb{R}^M, \mathbf{y} \in \mathbb{R}^M)$ is a align function for which we consider two different alternatives:

$$g(\mathbf{x}, \mathbf{y}) = \begin{cases} \mathbf{v}^T \tanh(\mathbf{W}_a[\mathbf{x}; \mathbf{y}]) \\ \mathbf{cosine}(x, y) \end{cases} \tag{8}$$

where $\mathbf{v} \in \mathbb{R}^M$ is a parameter vector.

In our current implementation, the similarity measure is cosine similarity.

Writing The memory can be written by two operations: erase and add.

$$\mathbf{M}_t = \mathbf{M}_{t-1}(\mathbf{1} - \alpha_t \mathbf{e_t^T}) + \alpha_t \mathbf{a_t^T}, \tag{9}$$

where $\mathbf{e}_t, \mathbf{a}_t \in \mathbb{R}^M$ represent erase and add vectors respectively.

To facilitate the following statements, we re-write the writing equation as:

$$\mathbf{M}_t = \mathbf{f}_{write}(\mathbf{M}_{t-1}, \alpha_t, \mathbf{h}_t). \tag{10}$$

Deep Fusion between External and Internal Memories After we obtain the information from external memory, we need a strategy to comprehensively utilize information from both external and internal memory.

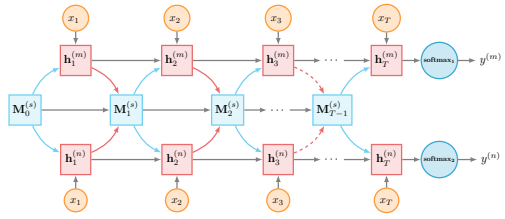
To better control signals flowing from external memory, inspired by (Wang and Cho, 2015), we propose a deep fusion strategy to keep internal and external memories interacting closely without being conflated.

In detail, the state $\mathbf{h}_t$ of LSTM at step t depends on both the read vector $\mathbf{r}_t$ from external memory, and internal memory c_t, which is computed by

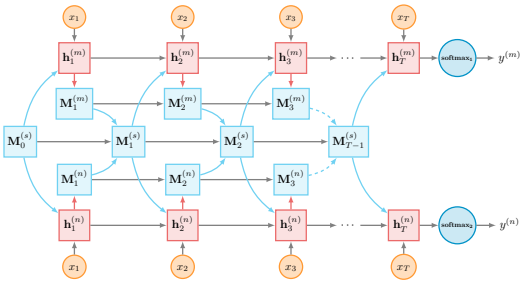
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t + \mathbf{g}_t \odot (\mathbf{W}_f \mathbf{r}_t)), \tag{11}$$

where $\mathbf{W}_f$ is parameter matrix, and $\mathbf{g}_t$ is a fusion gate to select information from external memory, which is computed by

$$\mathbf{g}_t = \sigma(\mathbf{W}_r \mathbf{r}_t + \mathbf{W}_c \mathbf{c}_t), \tag{12}$$



(a) Global Memory Architecture



(b) Local-Global Hybrid Memory Architecture

Figure 2: Two architectures for modelling text with multi-task learning.

where $\mathbf{W}_r$ and $\mathbf{W}_c$ are parameter matrices.

Finally, the update of external memory enhanced LSTM unit can be written precisely as

$$(\mathbf{h}_t, \mathbf{M}_t, \mathbf{c}_t) = \textbf{ME-LSTM}(\mathbf{h}_{t-1}, \mathbf{M}_{t-1}, \mathbf{c}_{t-1}, \mathbf{x}_t, \theta_p, \theta_q), \quad (13)$$

where θ_p represents all the parameters of LSTM internal structure and θ_q represents all the parameters to maintain the external memory.

Overall, the external memory enables ME-LSTM to have larger capability to store more information, thereby increasing the ability of ME-LSTM. The read and write operations allow ME-LSTM to capture complex sentence patterns.

3 Deep Architectures with Shared Memory for Multi-task Learning

Most existing neural network methods are based on supervised training objectives on a single task (Collobert et al., 2011; Socher et al., 2013; Kalchbrenner et al., 2014). These methods often suffer from the limited amounts of training data. To deal with this problem, these models often involve an unsupervised pre-training phase. This unsupervised pre-training is effective to improve the final performance, but it does not directly optimize the desired task.

Motivated by the success of multi-task learning (Caruana, 1997), we propose two deep architectures with shared external memory to leverage supervised data from many related tasks. Deep neural model is well suited for multi-task learning since the features learned from a task may be useful for other tasks. Figure 2 gives an illustration of our proposed architectures.

ARC-I: Global Shared Memory In ARC-I, the input is modelled by a task-specific LSTM and external shared memory. More formally, given an input text x, the task-specific output $\mathbf{h}_t^{(m)}$ of task m at step t is defined as

$$(\mathbf{h}_t^{(m)}, \mathbf{M}_t^{(s)}, \mathbf{c}_t^{(m)}) = \textbf{ME-LSTM}(\mathbf{h}_{t-1}^{(m)}, \mathbf{M}_{t-1}^{(s)}, \mathbf{c}_{t-1}^{(m)}, \mathbf{x}_t, \theta_p^{(m)}, \theta_q^{(s)}), \quad (14)$$

where $\mathbf{x}_t$ represents word embeddings of word x_t; the superscript s represents the parameters are shared across different tasks; the superscript m represents that the parameters or variables are task-specific for task m.

Here all tasks share single global memory $\mathbf{M}^{(s)}$, meaning that all tasks can read information from it and have the duty to write their shared or task-specific information into the memory.

$$\mathbf{M}_t^{(s)} = \mathbf{f}_{write}(\mathbf{M}_{t-1}^{(s)}, \alpha_t^{(s)}, \mathbf{h}_t^{(m)}) \quad (15)$$

After calculating the task-specific representation of text $\mathbf{h}_T^{(m)}$ for task m, we can predict the probability distribution over classes.

ARC-II: Local-Global Hybrid Memory In ARC-I, all tasks share a global memory, but can also record task-specific information besides shared information. To address this, we allocate each task a local task-specific external memory, which can further write shared information to a global memory for all tasks.

More generally, for task m, we assign each task-specific LSTM with a local memory $\mathbf{M}^{(m)}$, followed by a global memory $\mathbf{M}^{(s)}$, which is shared across different tasks.

The read and write operations of the local and global memory are defined as

$$\mathbf{r}_t^{(m)} = \alpha_t^{(m)} \mathbf{M}_t^{(m)}, \quad (16)$$

	Movie Reviews	Product Reviews
Embedding dimension	100	100
Hidden layer size	100	100
External memory size	(50,20)	(50,20)
Initial learning rate	0.01	0.1
Regularization	0	$1E{-}5$

Table 2: Hyper-parameters of our models.

$$\mathbf{M}_t^{(m)} = \mathbf{f}_{write}(\mathbf{M}_{t-1}^{(m)}, \alpha_t^{(m)}, \mathbf{h}_t^{(m)}), \quad (17)$$

$$\mathbf{r}_t^{(s)} = \alpha_{t-1}^{(s)} \mathbf{M}_{t-1}^{(s)}, \quad (18)$$

$$\mathbf{M}_t^{(s)} = \mathbf{f}_{write}(\mathbf{M}_{t-1}^{(s)}, \alpha_t^{(s)}, \mathbf{r}_t^{(m)}), \quad (19)$$

where the superscript s represents the parameters are shared across different tasks; the superscript m represents that the parameters or variables are task-specific for task m.

In ARC-II, the local memories enhance the capacity of memorizing, while global memory enables the information flowing from different tasks to interact sufficiently.

4 Training

The task-specific representation $\mathbf{h}^{(m)}$, emitted by the deep muti-task architectures, is ultimately fed into the corresponding task-specific output layers.

$$\hat{\mathbf{y}}^{(m)} = \mathrm{softmax}(\mathbf{W}^{(m)}\mathbf{h}^{(m)} + \mathbf{b}^{(m)}), \quad (20)$$

where $\hat{\mathbf{y}}^{(m)}$ is prediction probabilities for task m.

Given M related tasks, our global cost function is the linear combination of cost function for all tasks.

$$\phi = \sum_{m=1}^{M} \lambda_m L(\hat{y}^{(m)}, y^{(m)}) \quad (21)$$

where λ_m is the weights for each task m respectively.

Computational Cost Compared with vanilla LSTM, our proposed two models do not cause much extra computational cost while converge faster. In our experiment, the most complicated ARC-II, costs 2 times as long as vanilla LSTM.

5 Experiment

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

5.1 Datasets

The used multi-task datasets are briefly described as follows. The detailed statistics are listed in Table 1.

Movie Reviews The movie reviews dataset consists of four sub-datasets about movie reviews.

- **SST-1** The movie reviews with five classes in the Stanford Sentiment Treebank[1] (Socher et al., 2013).
- **SST-2** The movie reviews with binary classes. It is also from the Stanford Sentiment Treebank.
- **SUBJ** The movie reviews with labels of subjective or objective (Pang and Lee, 2004).
- **IMDB** The IMDB dataset[2] consists of 100,000 movie reviews with binary classes (Maas et al., 2011). One key aspect of this dataset is that each movie review has several sentences.

Product Reviews This dataset[3], constructed by Blitzer et al. (2007), contains Amazon product reviews from four different domains: Books, DVDs, Electronics and Kitchen appliances. The goal in each domain is to classify a product review as either positive or negative. The datasets in each domain are partitioned randomly into training data, development data and testing data with the proportion of 70%, 20% and 10% respectively.

5.2 Competitor Methods for Multi-task Learning

The multi-task frameworks proposed by previous works are various while not all can be applied to the tasks we focused. Nevertheless, we chose two most related neural models for multi-task learning and implement them as strong competitor methods .

- MT-CNN: This model is proposed by Collobert and Weston (2008) with convolutional layer, in which lookup-tables are shared partially while other layers are task-specific.
- MT-DNN: The model is proposed by Liu et al. (2015b) with bag-of-words input and multi-

[1] http://nlp.stanford.edu/sentiment.

[2] http://ai.stanford.edu/~amaas/data/sentiment/

[3] https://www.cs.jhu.edu/~mdredze/datasets/sentiment/

layer perceptrons, in which a hidden layer is shared.

5.3 Hyperparameters and Training

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

The word embeddings for all of the models are initialized with the 100d GloVe vectors (840B token version, (Pennington et al., 2014)) and fine-tuned during training to improve the performance. The other parameters are initialized by randomly sampling from uniform distribution in $[-0.1, 0.1]$. The mini-batch size is set to 16.

For each task, we take the hyperparameters which achieve the best performance on the development set via an small grid search over combinations of the initial learning rate $[0.1, 0.01]$, l_2 regularization $[0.0, 5E{-}5, 1E{-}5]$. For datasets without development set, we use 10-fold cross-validation (CV) instead. The final hyper-parameters are set as Table 2.

5.4 Multi-task Learning of Movie Reviews

We first compare our proposed models with the baseline system for single task classification. Table 3 shows the classification accuracies on the movie reviews dataset. The row of "Single Task" shows the results of LSTM and ME-LSTM for each individual task. With the help of multi-task learning, the performances of these four tasks are improved by 1.8% (ARC-I) and 2.9% (ARC-II) on average relative to LSTM. We can find that the architecture of local-global hybrid external memory has better performances. The reason is that the global memory in ARC-I could store some task-specific information besides shared information, which maybe noisy to other tasks. Moreover, both of our proposed models outperform MT-CNN and MT-DNN, which indicates the effectiveness of our proposed shared mechanism. To give an intuitive evaluation of these results, we also list the following state-of-the-art neural models. With the help of utilizing the shared information of several related tasks, our results outperform most of state-of-the-art models. Although Tree-LSTM outperforms our method on SST-1, it needs an external parser to get the sentence topological structure. It is worth noticing that our models are generic and compatible with the other LSTM based models. For example, we can easily extend our models to incorporate the Tree-LSTM model.

5.5 Multi-task Learning of Product Reviews

Table 4 shows the classification accuracies on the tasks of product reviews. The row of "Single Task" shows the results of the baseline for each individual task. With the help of global shared memory (ARC-I), the performances of these four tasks are improved by an average of 2.9%(2.6%) compared with LSTM(ME-LSTM). ARC-II achieves best performances on three sub-tasks, and its average improvement is 3.7%(3.5%). Compared with MT-CNN and MT-DNN, our models achieve a better performance. We think the reason is that our models can not only share lexical information but share complicated patterns of sentences by reading/writing operations of external memory. Furthermore, these results on product reviews are consistent with that on movie reviews, which shows our architectures are robust.

5.6 Case Study

To get an intuitive understanding of what is happening when we use shared memory to predict the class of text, we design an experiment to compare and analyze the difference between our models and vanilla LSTM, thereby demonstrating the effectiveness of our proposed architectures.

We sample two sentences from the SST-2 validation dataset, and the changes of the predicted sentiment score at different time steps are shown in Figure 3, which are obtained by vanilla LSTM and ARC-I respectively. Additionally, both models are bidirectional for better visualization. To get more insights into how the shared external memory influences the specific task, we plot and observe the evolving activation of fusion gates through time, which controls signals flowing from a shared external memory to task-specific output, to understand the behaviour of neurons.

For the sentence "*It is a cookie-cutter movie, a cut-and-paste job.*", which has a negative sentiment, while the standard LSTM gives a wrong prediction due to not understanding the informative words "*cookie-cutter*" and "*cut-and-paste*".

In contrast, our model makes a correct prediction and the reason can be inferred from the activation of

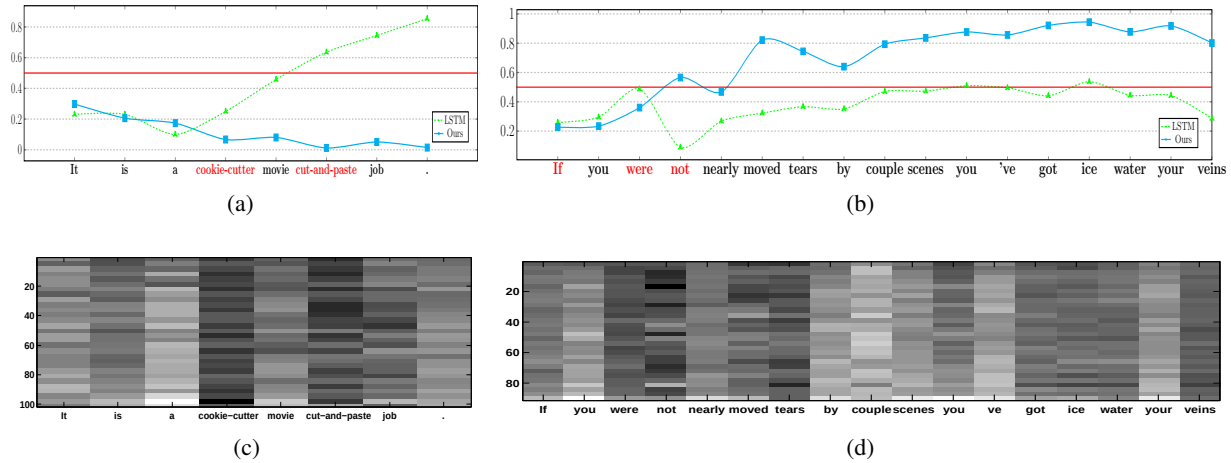


Figure 3: (a)(b) The change of the predicted sentiment score at different time steps. Y-axis represents the sentiment score, while X-axis represents the input words in chronological order. The red horizontal line gives a border between the positive and negative sentiments. (c)(d) Visualization of the fusion gate's activation.

fusion gates. As shown in Figure 3-(c), we can see clearly the neurons are activated much when they take input as "*cookie-cutter*" and "*cut-and-paste*", which indicates much information in shared memory has be passed into LSTM, therefore enabling the model to give a correct prediction.

Another case "*If you were not nearly moved to tears by a couple of scenes , you 've got ice water in your veins*", a subjunctive clause introduced by "*if*", has a positive sentiment.

As shown in Figure 3-(b,d), vanilla LSTM failed to capture the implicit meaning behind the sentence, while our model is sensitive to the pattern "*If ... were not ...*" and has an accurate understanding of the sentence, which indicates the shared memory mechanism can not only enrich the meaning of certain words, but teach some information of sentence structure to specific task.

6 Related Work

Neural networks based multi-task learning has been proven effective in many NLP problems (Collobert and Weston, 2008; Glorot et al., 2011; Liu et al., 2015b; Liu et al., 2016b). In most of these models, the lower layers are shared across all tasks, while top layers are task-specific.

Collobert and Weston (2008) used a shared representation for input words and solved different traditional NLP tasks within one framework. However, only one lookup table is shared, and the other lookup tables and layers are task-specific.

Liu et al. (2015b) developed a multi-task DNN for learning representations across multiple tasks. Their multi-task DNN approach combines tasks of query classification and ranking for web search. But the input of the model is bag-of-word representation, which loses the information of word order.

More recently, several multi-task encoder-decoder networks were also proposed for neural machine translation (Dong et al., 2015; Luong et al., 2015; Firat et al., 2016), which can make use of cross-lingual information.

Unlike these works, in this paper we design two neural architectures with shared memory for multi-task learning, which can store useful information across the tasks. Our architectures are relatively loosely coupled, and therefore more flexible to expand. With the help of shared memory, we can obtain better task-specific sentence representation by utilizing the knowledge obtained by other related tasks.

7 Conclusion and Future Work

In this paper, we introduce two deep architectures for multi-task learning. The difference with the previous models is the mechanisms of sharing information among several tasks. We design an external memory to store the knowledge shared by several re-

lated tasks. Experimental results show that our models can improve the performances of several related tasks by exploring common features.

In addition, we also propose a deep fusion strategy to integrate the information from the external memory into task-specific LSTM with a fusion gate.

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

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011 and 61672162), the National High Technology Research and Development Program of China (No. 2015AA015408).

References

Marcin Andrychowicz and Karol Kurach. 2016. Learning efficient algorithms with hierarchical attentive memory. *arXiv preprint arXiv:1602.03218*.

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *ArXiv e-prints*, September.

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Ivo Danihelka, Greg Wayne, Benigno Uria, Nal Kalchbrenner, and Alex Graves. 2016. Associative long short-term memory. *CoRR*, abs/1602.03032.

Misha Denil, Alban Demiraj, Nal Kalchbrenner, Phil Blunsom, and Nando de Freitas. 2014. Modelling, visualising and summarising documents with a single convolutional neural network. *arXiv preprint arXiv:1406.3830*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the ACL*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv:1601.01073*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 513–520.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of The 32nd International Conference on Machine Learning*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

PengFei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015a. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the Conference on EMNLP*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015b. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *NAACL*.

Pengfei Liu, Xipeng Qiu, Jifan Chen, and Xuanjing Huang. 2016a. Deep fusion LSTMs for text semantic matching. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016b. Recurrent neural network for text classification with multi-task learning. In *Proceedings of International Joint Conference on Artificial Intelligence*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the ACL*, pages 142–150.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, pages 1201–1211.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL*.

Tian Wang and Kyunghyun Cho. 2015. Larger-context language modelling. *arXiv preprint arXiv:1511.03729*.

Dataset		Type	Train Size	Dev. Size	Test Size	Class	Avg. Length	Vocabulary Size
Movie	SST-1	Sen.	8544	1101	2210	5	19	18K
	SST-2	Sen.	6920	872	1821	2	18	15K
	SUBJ	Sen.	9000	-	1000	2	21	21K
	IMDB	Doc.	25,000	-	25,000	2	294	392K
Product	Books	Doc.	1400	200	400	2	181	27K
	DVDs	Doc.	1400	200	400	2	197	29K
	Electronics	Doc.	1400	200	400	2	117	14K
	Kitchen	Doc.	1400	200	400	2	98	12K

Table 1: Statistics of two multi-task datasets. Each dataset consists of four related tasks.

Model		SST-1	SST-2	SUBJ	IMDB	AvgΔ
Single Task	LSTM	45.9	85.8	91.6	88.5	-
	ME-LSTM	46.4	85.5	91.0	88.7	-
Multi-task	ARC-I	48.6	87.0	93.8	89.8	+(1.8/1.9)
	ARC-II	**49.5**	**87.8**	**95.0**	**91.2**	+(2.9/3.0)
	MT-CNN	46.7	86.1	92.2	88.4	-
	MT-DNN	44.5	84.0	90.1	85.6	-
NBOW		42.4	80.5	91.3	83.6	-
RAE (Socher et al., 2011)		43.2	82.4	-	-	-
MV-RNN (Socher et al., 2012)		44.4	82.9	-	-	-
RNTN (Socher et al., 2013)		45.7	85.4	-	-	-
DCNN (Kalchbrenner et al., 2014)		48.5	86.8	-	89.3	-
CNN-multichannel (Kim, 2014)		47.4	**88.1**	93.2	-	-
Tree-LSTM (Tai et al., 2015)		**50.6**	86.9	-	-	-

Table 3: Accuracies of our models on movie reviews tasks against state-of-the-art neural models. The last column gives the improvements relative to LSTM and ME-LSTM respectively. **NBOW**: Sums up the word vectors and applies a non-linearity followed by a softmax classification layer. **RAE**: Recursive Autoencoders with pre-trained word vectors from Wikipedia (Socher et al., 2011). **MV-RNN**: Matrix-Vector Recursive Neural Network with parse trees (Socher et al., 2012). **RNTN**: Recursive Neural Tensor Network with tensor-based feature function and parse trees (Socher et al., 2013). **DCNN**: Dynamic Convolutional Neural Network with dynamic k-max pooling (Kalchbrenner et al., 2014; Denil et al., 2014). **CNN-multichannel**: Convolutional Neural Network (Kim, 2014). **Tree-LSTM**: A generalization of LSTMs to tree-structured network topologies (Tai et al., 2015).

Model		Books	DVDs	Electronics	Kitchen	AvgΔ
Single Task	LSTM	78.0	79.5	81.2	81.8	-
	ME-LSTM	77.5	80.2	81.5	82.1	-
Multi-task	ARC-I	81.2	82.0	84.5	**84.3**	+(2.9/2.6)
	ARC-II	**82.8**	**83.0**	**85.5**	84.0	+(3.7/3.5)
	MT-CNN	80.2	81.0	83.4	83.0	-
	MT-DNN	79.7	80.5	82.5	82.8	-

Table 4: Accuracies of our models on product reviews dataset. The last column gives the improvement relative to LSTM and ME-LSTM respectively.