A Generalized Recurrent Neural Architecture for Text Classification with Multi-Task Learning

Honglun Zhang1, Liqiang Xiao1, Yongkun Wang2, Yaohui Jin1,2
1State Key Lab of Advanced Optical Communication System and Network
2Network and Information Center
Shanghai Jiao Tong University
{ykw}@sjtu.edu.cn

Abstract

Multi-task learning leverages potential correlations among related tasks to extract common features and yield performance gains. However, most previous works only consider simple or weak interactions, thereby failing to model complex correlations among three or more tasks. In this paper, we propose a multi-task learning architecture with four types of recurrent neural layers to fuse information across multiple related tasks. The architecture is structurally flexible and considers various interactions among tasks, which can be regarded as a generalized case of many previous works. Extensive experiments on five benchmark datasets for text classification show that our model can significantly improve performances of related tasks with additional information from others.

1 Introduction

Neural network based models have been widely exploited with the prosperities of Deep Learning [Bengio et al., 2013] and achieved inspiring performances on many NLP tasks, such as text classification [Chen et al., 2015; Liu et al., 2015a], semantic matching [Liu et al., 2016d; 2016a] and machine translation [Sutskever et al., 2014]. These models are robust at feature engineering and can represent words, sentences and documents as fix-length vectors, which contain rich semantic information and are ideal for subsequent NLP tasks.

One formidable constraint of deep neural networks (DNN) is their strong reliance on large amounts of annotated corpus due to substantial parameters to train. A DNN trained on limited data is prone to overfitting and incapable to generalize well. However, constructions of large-scale high-quality labeled datasets are extremely labor-intensive. To solve the problem, these models usually employ a pre-trained lookup table, also known as Word Embedding [Mikolov et al., 2013b], to map words into vectors with semantic implications. However, this method just introduces extra knowledge and does not directly optimize the targeted task. The problem of insufficient annotated resources is not solved either.

Multi-task learning leverages potential correlations among related tasks to extract common features, increase corpus size implicitly and yield classification improvements. Inspired by [Caruana, 1997], there are a large literature dedicated for multi-task learning with neural network based models [Collobert and Weston, 2008; Liu et al., 2015b; 2016b; 2016c]. These models basically share some lower layers to capture common features and further feed them to subsequent task-specific layers, which can be classified into three types:

- **Type-I** One dataset annotated with multiple labels and one input with multiple outputs.
- **Type-II** Multiple datasets with respective labels and one input with multiple outputs, where samples from different tasks are fed one by one into the models sequentially.
- **Type-III** Multiple datasets with respective labels and multiple inputs with multiple outputs, where samples from different tasks are jointly learned in parallel.

In this paper, we propose a generalized multi-task learning architecture with four types of recurrent neural layers for text classification. The architecture focuses on Type-III, which involves more complicated interactions but has not been researched yet. All the related tasks are jointly integrated into a single system and samples from different tasks are trained in parallel. In our model, every two tasks can directly interact with each other and selectively absorb useful information, or communicate indirectly via a shared intermediate layer. We also design a global memory storage to share common features and collect interactions among all tasks.

We conduct extensive experiments on five benchmark datasets for text classification. Compared to learning separately, jointly learning multiple relative tasks in our model demonstrate significant performance gains for each task.

Our contributions are three-folds:

- Our model is structurally flexible and considers various interactions, which can be concluded as a generalized case of many previous works with deliberate designs.
- Our model allows for interactions among three or more tasks simultaneously and samples from different tasks are trained in parallel with multiple inputs.
- We consider different scenarios of multi-task learning and demonstrate strong results on several benchmark classification datasets. Our model outperforms most of state-of-the-art baselines.
2 Problem Statements

2.1 Single-Task Learning

For a single supervised text classification task, the input is a word sequences denoted by \( x = \{x_1, x_2, ..., x_T \} \), and the output is the corresponding class label \( y \) or class distribution \( \hat{y} \). A lookup layer is used first to get the vector representation \( x_t \in \mathbb{R}^d \) of each word \( x_t \). A classification model \( f \) is trained to transform each \( x = \{x_1, x_2, ..., x_T \} \) into a predicted distribution \( \hat{y} \).

\[
f(x_1, x_2, ..., x_T) = \hat{y}
\]

and the training objective is to minimize the total cross-entropy of the predicted and true distributions over all samples.

\[
L = -\sum_{i=1}^{N} \sum_{j=1}^{C} y_{ij} \log \hat{y}_{ij} \tag{2}
\]

where \( N \) denotes the number of training samples and \( C \) is the class number.

2.2 Multi-Task Learning

Given \( K \) supervised text classification tasks, \( T_1, T_2, ..., T_K \), a jointly learning model \( F \) is trained to transform multiple inputs into a combination of predicted distributions in parallel.

\[
F(x^{(1)}, x^{(2)}, ..., x^{(K)}) = (\hat{y}^{(1)}, \hat{y}^{(2)}, ..., \hat{y}^{(K)}) \tag{3}
\]

where \( x^{(k)} \) are sequences from each tasks and \( \hat{y}^{(k)} \) are the corresponding predictions.

The overall training objective of \( F \) is to minimize the weighted linear combination of costs for all tasks.

\[
L = -\sum_{i=1}^{N} \sum_{k=1}^{K} \lambda_k \sum_{j=1}^{C} y_{ij}^{(k)} \log \hat{y}_{ij}^{(k)} \tag{4}
\]

where \( N \) denotes the number of sample collections, \( C_k \) and \( \lambda_k \) are class numbers and weights for each task \( T_k \) respectively.

2.3 Three Perspectives of Multi-Task Learning

Different tasks may differ in characteristics of the word sequences \( x \) or the labels \( y \). We compare lots of benchmark tasks for text classification and conclude three different perspectives of multi-task learning.

- **Multi-Cardinality** Tasks are similar except for cardinality parameters, for example, movie review datasets with different average sequence lengths and class numbers.

- **Multi-Domain** Tasks involve contents of different domains, for example, product review datasets on books, DVDs, electronics and kitchen appliances.

- **Multi-Objective** Tasks are designed for different objectives, for example, sentiment analysis, topics classification and question type judgment.

The simplest multi-task learning scenario is that all tasks share the same cardinality, domain and objective, while come from different sources, so it is intuitive that they can obtain useful information from each other. However, in the most complex scenario, tasks may vary in cardinality, domain and even objective, where the interactions among different tasks can be quite complicated and implicit. We will evaluate our model on different scenarios in the Experiment section.

3 Methodology

Recently neural network based models have obtained substantial interests in many natural language processing tasks for their capabilities to represent variable-length text sequences as fix-length vectors, for example, Neural Bag-of-Words (NBOW), Recurrent Neural Networks (RNN), Recursive Neural Networks (RecNN) and Convolutional Neural Network (CNN). Most of them first map sequences of words, n-grams or other semantic units into embedding representations with a pre-trained lookup table, then fuse these vectors with different architectures of neural networks, and finally utilize a softmax layer to predict categorical distribution for specific classification tasks. For recurrent neural network, input vectors are absorbed one by one in a recurrent way, which makes RNN particularly suitable for natural language processing tasks.

3.1 Recurrent Neural Network

A recurrent neural network maintains an internal hidden state vector \( h_t \) that is recurrently updated by a transition function \( f \). At each time step \( t \), the hidden state \( h_t \) is updated according to the current input vector \( x_t \) and the previous hidden state \( h_{t-1} \).

\[
h_t = \begin{cases} 0 & t = 0 \\ f(h_{t-1}, x_t) & \text{otherwise} \end{cases}
\]

where \( f \) is usually a composition of an element-wise non-linearity with an affine transformation of both \( x_t \) and \( h_{t-1} \).

In this way, recurrent neural networks can comprehend a sequence of arbitrary length into a fix-length vector and feed it to a softmax layer for text classification or other NLP tasks. However, gradient vector of \( f \) can grow or decay exponentially over long sequences during training, also known as the gradient exploding or vanishing problems, which makes it difficult to learn long-term dependencies and correlations for RNNs.

[Hochreiter and Schmidhuber, 1997] proposed Long Short-Term Memory Network (LSTM) to tackle the above problems. Apart from the internal hidden state \( h_t \), LSTM also maintains a internal hidden memory cell and three gating mechanisms. While there are numerous variants of the standard LSTM, here we follow the implementation of [Graves, 2013]. At each time step \( t \), states of the LSTM can be fully represented by five vectors in \( \mathbb{R}^n \), an input gate \( i_t \), a forget gate \( f_t \), an output gate \( o_t \), the hidden state \( h_t \) and the memory cell \( c_t \), which adhere to the following transition functions.

\[
i_t = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1} + b_i) \tag{6}
\]

\[
f_t = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1} + b_f) \tag{7}
\]

\[
o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_{t-1} + b_o) \tag{8}
\]

\[
c_t = \tanh(W_c x_t + U_c h_{t-1}) \tag{9}
\]

\[
c_t = f_t \odot c_{t-1} + i_t \odot c_t \tag{10}
\]

\[
h_t = o_t \odot \tanh(c_t) \tag{11}
\]
where $x_t$ is the current input, $\sigma$ denotes logistic sigmoid function and $\odot$ denotes element-wise multiplication. By selectively controlling portions of the memory cell $c_t$ to update, erase and forget at each time step, LSTM can better comprehend long-term dependencies with respect to labels of the whole sequences.

### 3.2 A Generalized Architecture

Based on the LSTM implementation of [Graves, 2013], we propose a generalized multi-task learning architecture for text classification with four types of recurrent neural layers to convey information inside and among tasks. Figure 1 illustrates the structure design and information flows of our model, where three tasks are jointly learned in parallel.

As Figure 1a shows, each task owns a LSTM-based Single Layer for intra-task learning. Pair-wise Coupling Layer and Local Fusion Layer are designed for direct and indirect inter-task interactions. And we further utilize a Global Fusion Layer to maintain a global memory for information shared among all tasks.

**Single Layer**
Each task owns a LSTM-based Single Layer with a collection of parameters $\Phi^{(k)}$, taking Eqs.(9) for example.

$$c_t^{(k)} = \tanh(W_c^{(k)}x_t^{(k)} + U_c^{(k)}h_{t-1}^{(k)})$$  \hspace{1cm} (12)

Input sequences of each task are transformed into vector representations ($x^{(1)}, x^{(2)}, ..., x^{(K)}$), which are later recurrently fed into the corresponding Single Layers. The hidden states at the last time step $h_T^{(k)}$ of each Single Layer can be regarded as fix-length representations of the whole sequences, which are followed by a fully connected layer and a softmax non-linear layer to produce class distributions.

$$\hat{y}^{(k)} = \text{softmax}(W^{(k)}h_T^{(k)} + b^{(k)})$$ \hspace{1cm} (13)

where $\hat{y}^{(k)}$ is the predicted class distribution for $x^{(k)}$.

**Coupling Layer**
Besides Single Layers, we design Coupling Layers to model direct pair-wise interactions between tasks. For each pair of tasks, hidden states and memory cells of the Single Layers can obtain extra information directly from each other, as shown in Figure 1b.

We re-define Eqs.(12) and utilize a gating mechanism to control the portion of information flows from one task to another. The memory content $c_t^{(k)}$ of each Single Layer is updated on the leverage of pair-wise couplings.

$$c_t^{(k)} = \tanh(W_c^{(k)}x_t^{(k)} + \sum_{j=1}^{K} g^{(j-k)}U_c^{(j)}h_{t-1}^{(j)})$$ \hspace{1cm} (14)

$$g^{(j-k)} = \sigma(W_c^{(j-k)}x_t^{(k)} + U_c^{(j)}h_{t-1}^{(j)})$$ \hspace{1cm} (15)

where $g^{(j-k)}$ controls the portion of information flow from $T_j$ to $T_k$, based on the correlation strength between $x_t^{(k)}$ and $h_{t-1}^{(j)}$ at the current time step.

In this way, the hidden states and memory cells of each Single Layer can obtain extra information from other tasks and stronger relevance results in higher chances of reception.

**Local Fusion Layer**
Different from Coupling Layers, Local Fusion Layers introduce a shared bi-directional LSTM Layer to model indirect pair-wise interactions between tasks. For each pair of tasks, we feed the Local Fusion Layer with the concatenation of both inputs, $x_t^{(j,k)} = x_t^{(j)} \odot x_t^{(k)}$, as shown in Figure 1c. We denote the output of the Local Fusion Layer as

![Figure 1: A generalized recurrent neural architecture for text with multi-task learning](image-url)
h_t^{j,k} = h_t^{j,k} \oplus h_t^{j,k}, \text{ a concatenation of hidden states from the forward and backward LSTM at each time step.}

Similar to Coupling Layers, hidden states and memory cells of the Single Layers can selectively decide how much information to accept from the pair-wise Local Fusion Layers. We re-define Eqs.(14) by considering the interactions between the memory content c_t^{j,k} and outputs of the Local Fusion Layers as follows.

\[
\begin{align*}
\hat{c}_t^{(k)} &= \tanh(W_c^{(k)}x_t^{(k)} + C_t^{(k)} + LF_t^{(k)}) \\
LF_t^{(k)} &= \sum_{j=1, j\neq k}^K g^{(j,k)}U^{(j,k)}h_t^{(j,k)} \\
g^{(j,k)} &= \sigma(W_{gf}^{(j,k)}x_t^{(j,k)} + U_{gf}^{(j,k)}h_t^{(j,k)})
\end{align*}
\]

where C_t^{(k)} denotes the coupling term in Eqs.(14) and LF_t^{(k)} represents the local fusion term. Again, we employ a gating mechanism g^{(j,k)} to control the portion of information flow from the Local Coupling Layers to T_k.

Global Fusion Layer

Indirect interactions between Single Layers can be pair-wise or global, so we further propose the Global Fusion Layer as a shared memory storage among all tasks. The Global Fusion Layer consists of a bi-directional LSTM layer with the inputs x_t^{(g)} = x_t^{(1)} \oplus x_t^{(2)} \oplus \cdots \oplus x_t^{(K)} and the outputs h_t^{(g)} = h_t^{(g)} \oplus h_t^{(g)}.

We denote the global fusion term as GF_t^{(k)} and the memory content c_t^{(k)} is calculated as follows.

\[
\begin{align*}
\hat{c}_t^{(k)} &= \tanh(W_c^{(k)}x_t^{(k)} + C_t^{(k)} + LF_t^{(k)} + GF_t^{(k)}) \\
GF_t^{(k)} &= \sigma(W_6^{(k)}x_t^{(k)} + U_6^{(k)}h_t^{(g)} + U_6^{(g)}h_t^{(g)})
\end{align*}
\]

As a result, our architecture covers complicated interactions among different tasks. It is capable of mapping a collection of input sequences from different tasks into a combination of predicted class distributions in parallel, as shown in Eqs.(3).

3.3 Sampling & Training

Most previous multi-task learning models [Collobert and Weston, 2008; Liu et al., 2015b; 2016b; 2016c] belongs to Type-I or Type-II. The total number of input samples is N = \sum_{k=1}^K N_k, where N_k are the sample numbers of each task.

However, our model focuses on Type-III and requires a 4-D tensor N \times K \times T \times d as inputs, where N, K, T, d are total number of input collections, task number, sequence length and embedding size respectively. Samples from different tasks are jointly learned in parallel so the total number of all possible input collections is N_{max} = \prod_{k=1}^K N_k. We propose a Task Oriented Sampling algorithm to generate sample collections for improvements of a specific task T_k.

Given the generated sequence collections X and label combinations Y, the overall loss function can be calculated based on Eqs.(4) and (13). The training process is conducted in a stochastic manner until convergence. For each loop, we randomly select a collection from the N candidates and update the parameters by taking a gradient step.

4 Experiment

In this section, we design three different scenarios of multi-task learning based on five benchmark datasets for text classification, we investigate the empirical performances of our model and compare it to existing state-of-the-art models.

4.1 Datasets

As Table 1 shows, we select five benchmark datasets for text classification and design three experiment scenarios to evaluate the performances of our model.

- Multi-Cardinality Movie review datasets with different average lengths and class numbers, including SST-1 [Socher et al., 2013], SST-2 and IMDB [Maas et al., 2011].
- Multi-Domain Product review datasets on different domains from Multi-Domain Sentiment Dataset [Blitzer et al., 2007], including Books, DVDs, Electronics and Kitchen.
- Multi-Objective Classification datasets with different objectives, including IMDB, RN [Apté et al., 1994] and QC [Li and Roth, 2002].

4.2 Hyperparameters and Training

The whole network is trained through back propagation with stochastic gradient descent [Amari, 1993]. We obtain a pre-trained lookup table by applying Word2Vec [Mikolov et al., 2013a] on the Google News corpus, which contains more than 100B words with a vocabulary size of about 3M. All involved parameters are randomly initialized from a truncated normal distribution with zero mean and standard deviation.
Table 1: Five benchmark classification datasets: SST, IMDB, MDSD, RN, QC.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Description</th>
<th>Type</th>
<th>Length</th>
<th>Class</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>SST</td>
<td>Movie reviews in Stanford Sentiment Treebank including SST-1 and SST-2</td>
<td>Sentence</td>
<td>19 / 19</td>
<td>5 / 2</td>
<td>Sentiment</td>
</tr>
<tr>
<td>IMDB</td>
<td>Internet Movie Database</td>
<td>Document</td>
<td>279</td>
<td>2</td>
<td>Sentiment</td>
</tr>
<tr>
<td>MDSD</td>
<td>Product reviews on books, DVDs, electronics and kitchen appliances</td>
<td>Document</td>
<td>176 / 189 / 115 / 97</td>
<td>2</td>
<td>Sentiment</td>
</tr>
<tr>
<td>RN</td>
<td>Reuters Newswire topics classification</td>
<td>Document</td>
<td>146</td>
<td>46</td>
<td>Topics</td>
</tr>
<tr>
<td>QC</td>
<td>Question Classification</td>
<td>Sentence</td>
<td>10</td>
<td>6</td>
<td>Question Types</td>
</tr>
</tbody>
</table>

For each task $T_k$, we conduct TOS with $n_0 = 2$ to improve its performance. After training our model on the generated sample collections, we evaluate the performance of task $T_k$ by comparing $y^{(k)}$ and $y^{(k)}$ on the test set. We apply 10-fold cross-validation and different combinations of hyperparameters are investigated, of which the best one, as shown in Table 2, is reserved for comparisons with state-of-the-art models.

Table 2: Hyperparameter settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedding size</td>
<td>$d = 300$</td>
</tr>
<tr>
<td>Hidden layer size of LSTM</td>
<td>$n = 100$</td>
</tr>
<tr>
<td>Initial learning rate</td>
<td>$\eta = 0.1$</td>
</tr>
<tr>
<td>Regularization weight</td>
<td>$\lambda = 10^{-5}$</td>
</tr>
</tbody>
</table>

4.3 Results

We compare performances of our model with the implementation of [Graves, 2013] and the results are shown in Table 3. Our model obtains better performances in Multi-Domain scenario with an average improvement of 4.5%, where datasets are product reviews on different domains with similar sequence lengths and the same class number, thus producing stronger correlations. Multi-Cardinality scenario also achieves significant improvements of 2.77% on average, where datasets are movie reviews with different cardinalities.

However, Multi-Objective scenario benefits less from multi-task learning due to lacks of salient correlation among sentiment, topic and question type. The QC dataset aims to classify each question into six categories and its performance. After training our model on the generated sample collections, we evaluate the performance of task $T_k$ by comparing $y^{(k)}$ and $y^{(k)}$ on the test set. We apply 10-fold cross-validation and different combinations of hyperparameters are investigated, of which the best one, as shown in Table 2, is reserved for comparisons with state-of-the-art models.

Influences of $n_0$ in TOS

We further explore the influence of $n_0$ in TOS on our model, which can be any positive integer. A higher value means larger and more various samples combinations, while requires higher computation costs.

Figure 2 shows the performances of datasets in Multi-Domain scenario with different $n_0$. Compared to $n_0 = 1$, our model can achieve considerable improvements when $n_0 = 2$ as more samples combinations are available. However, there are no more salient gains as $n_0$ gets larger and potential noises from other tasks may lead to performance degradation. For a trade-off between efficiency and effectiveness, we determine $n_0 = 2$ as the optimal value for our experiments.

Figure 2: Influences of $n_0$ in TOS on different datasets

Pair-wise Performance Gain

In order to measure the correlation strength between two task $T_i$ and $T_j$, we learn them jointly with our model and define Pair-wise Performance Gain as $PPG_{ij} = \sqrt{P_{ij}P_{ij}'}$, where $P_i, P_j, P_i', P_j'$ are the performances of tasks $T_i$ and $T_j$ when learned individually and jointly.

We calculate PPGs for every two tasks in Table 1 and illustrate the results in Figure 3, where darkness of colors indicate strength of correlation. It is intuitive that datasets of Multi-Domain scenario obtain relatively higher PPGs with each other as they share similar cardinalities and abundant low-level linguistic characteristics. Sentences of QC dataset are much shorter and convey unique characteristics from other tasks, thus resulting in quite lower PPGs.

4.4 Comparisons with State-of-the-art Models

We apply the optimal hyperparameter settings and compare our model against the following state-of-the-art models:

- **NBOV** Neural Bag-of-Words that simply sums up embedding vectors of all words.
- **PV** Paragraph Vectors followed by logistic regression [Le and Mikolov, 2014].
- **MT-RNN** Multi-Task learning with Recurrent Neural Networks by a shared-layer architecture [Liu et al., 2016c].
Table 3: Results of our model on different scenarios

<table>
<thead>
<tr>
<th>Model</th>
<th>Multi-Cardinality</th>
<th>Multi-Domain</th>
<th>Multi-Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SST-1</td>
<td>SST-2</td>
<td>IMDB</td>
</tr>
<tr>
<td>Single Task</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our Model</td>
<td>49.2</td>
<td>87.7</td>
<td>91.6</td>
</tr>
</tbody>
</table>

Table 4: Comparisons with state-of-the-art models

<table>
<thead>
<tr>
<th>Model</th>
<th>SST-1</th>
<th>SST-2</th>
<th>IMDB</th>
<th>Books</th>
<th>DVDs</th>
<th>Electronics</th>
<th>Kitchen</th>
<th>QC</th>
</tr>
</thead>
<tbody>
<tr>
<td>NBOB</td>
<td>42.4</td>
<td>80.5</td>
<td>83.62</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PV</td>
<td>44.6</td>
<td>82.7</td>
<td>91.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MT-RNN</td>
<td>49.6</td>
<td>87.9</td>
<td>91.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MT-CNN</td>
<td>-</td>
<td>-</td>
<td>80.2</td>
<td>81.0</td>
<td>83.4</td>
<td>83.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MT-DNN</td>
<td>-</td>
<td>-</td>
<td>79.7</td>
<td>80.5</td>
<td>82.5</td>
<td>82.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GRNN</td>
<td>47.5</td>
<td>85.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>93.8</td>
<td></td>
</tr>
<tr>
<td>Our Model</td>
<td>49.2</td>
<td>87.7</td>
<td>91.6</td>
<td>83.5</td>
<td>84.0</td>
<td>86.2</td>
<td>84.8</td>
<td>92.3</td>
</tr>
</tbody>
</table>

Figure 3: Visualization of Pair-wise Performance Gains

- **MT-CNN** Multi-Task learning with Convolutional Neural Networks [Collobert and Weston, 2008] where lookup tables are partially shared.
- **MT-DNN** Multi-Task learning with Deep Neural Networks [Liu et al., 2015b] that utilizes bag-of-word representations and a hidden shared layer.
- **GRNN** Gated Recursive Neural Network for sentence modeling [Chen et al., 2015].

As Table 4 shows, our model obtains competitive or better performances on all tasks except for the QC dataset, as it contains poor correlations with other tasks. MT-RNN slightly outperforms our model on SST, as sentences from this dataset are much shorter than those from IMDB and MDSD, and another possible reason may be that our model are more complex and requires larger data for training. Our model proposes the designs of various interactions including coupling, local and global fusion, which can be further implemented by other state-of-the-art models and produce better performances.

5 Related Work

There are a large body of literatures related to multi-task learning with neural networks in NLP [Collobert and Weston, 2008; Liu et al., 2015b; 2016b; 2016c].

[Collobert and Weston, 2008] belongs to **Type-I** and utilizes shared lookup tables for common features, followed by task-specific neural layers for several traditional NLP tasks such as part-of-speech tagging and semantic parsing. They use a fix-size window to solve the problem of variable-length texts, which can be better handled by recurrent neural networks.

[Liu et al., 2015b; 2016b; 2016c] all belong to **Type-II** where samples from different tasks are learned sequentially. [Liu et al., 2015b] applies bag-of-word representation and information of word orders are lost. [Liu et al., 2016b] introduces an external memory for information sharing with a reading/writing mechanism for communicating, and [Liu et al., 2016c] proposes three different models for multi-task learning with recurrent neural networks. However, models of these two papers only involve pair-wise interactions, which can be regarded as specific implementations of Coupling Layer and Fusion Layer in our model.

Different from the above models, our model focuses on **Type-III** and utilize recurrent neural networks to comprehensively capture various interactions among tasks, both direct and indirect, local and global. Three or more tasks are learned simultaneously and samples from different tasks are trained in parallel benefiting from each other, thus obtaining better sentence representations.

6 Conclusion and Future Work

In this paper, we propose a multi-task learning architecture for text classification with four types of recurrent neural layers. The architecture is structurally flexible and can be regarded as a generalized case of many previous works with deliberate designs. We explore three different scenarios of multi-task learning and our model can improve performances of most tasks with additional related information from others in all scenarios.

In future work, we would like to investigate further implementations of couplings and fusions, and conclude more multi-task learning perspectives.
References


