# Learning Multi-Task Communication with Message Passing for Sequence Learning

**Pengfei Liu,**[†‡] **Jie Fu,**[‡*] **Yue Dong,**[‡♯*] **Xipeng Qiu,**[†] **Jackie Chi Kit Cheung**[‡♯]
[†]School of Computer Science, Fudan University, Shanghai Insitute of Intelligent Electroics & Systems
[‡]MILA [♯]McGill University
{pfliu14,xpqiu}@fudan.edu.cn, jie.fu@polymtl.ca,yue.dong2@mail.mcgill.ca,jcheung@cs.mcgill.ca

## Abstract

We present two architectures for multi-task learning with neural sequence models. Our approach allows the relationships between different tasks to be learned dynamically, rather than using an ad-hoc pre-defined structure as in previous work. We adopt the idea from message-passing graph neural networks, and propose a general **graph multi-task learning** framework in which different tasks can communicate with each other in an effective and interpretable way. We conduct extensive experiments in text classification and sequence labelling to evaluate our approach on multi-task learning and transfer learning. The empirical results show that our models not only outperform competitive baselines, but also learn interpretable and transferable patterns across tasks.

## Introduction

Neural multi-task learning models have driven state-of-the-art results to new levels in a number of language processing tasks, ranging from part-of-speech (POS) tagging (Yang, Salakhutdinov, and Cohen 2016, Søgaard and Goldberg 2016), parsing (Peng, Thomson, and Smith 2017, Guo et al. 2016), text classification (Liu, Qiu, and Huang 2016, Liu, Qiu, and Huang 2017) to machine translation (Luong et al. 2015, Firat, Cho, and Bengio 2016).

Multi-task learning utilizes the correlation between related tasks to improve the performance of each task. In practice, existing work often models task relatedness by simply defining shared common parameters over some pre-defined task structures. Figure 1-(a,b) shows two typical pre-defined topology structures which have been popular. A flat structure (Collobert and Weston 2008) assumes that all tasks jointly share a hidden space, while a hierarchical structure (Søgaard and Goldberg 2016, Hashimoto et al. 2017) specifies a partial order of the direction of information flow between tasks.

There are two major limitations to the above approaches. First, static pre-defined structures represent a strong assumption about the nature of the interaction between tasks, restricting the model's capacity to make use of shared information. For example, the structure in 1(a) does not allow the model to explicitly learn the strength of relatedness between tasks. This restriction prevents the model from fully

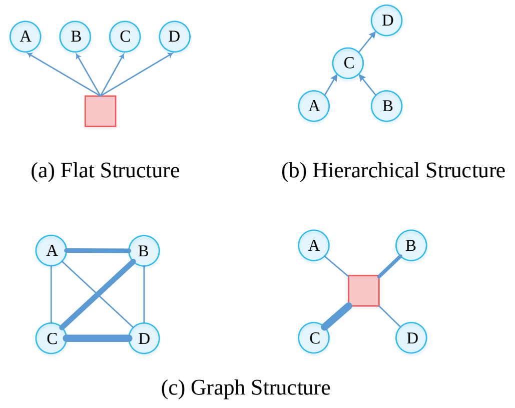


Figure 1: Different topology structures for multi-task learning. Each blue circle represents a task (A, B, C, D), while the red box denotes a virtual node, which stores shared information and facilitates communication. The boldness of the line indicates the strength of the relationships. The directed edges define the direction of information flow. For example, in sub-figure (b), task C receives information from task A, but not vice versa.

utilizing and handling the complexity of the data (Li and Tian 2015). Note that the strength of relatedness between tasks is itself not static but subject to change, depending on the data samples at hand. Second, these models are not interpretable to researchers and system developers, meaning that we learn little about what kinds of patterns have been shared besides the parameters themselves. Previous non-neural-network models (Bakker and Heskes 2003, Kim and Xing 2010, Chen et al. 2010) have demonstrated the importance of learning inter-task relationships for multi-task learning. However, there is little work giving an in-depth analysis in the neural setting.

The above issues motivate the following research questions: 1) How can we explicitly model complex relationships between different tasks? 2) Can we design models that learn interpretable shared structures?

To address these questions, we propose to model the relationships between language processing tasks using a graph structure, in which each task is regarded as a node. We take

[*]These two authors contributed equally

inspiration from the idea of message passing (Berendsen, van der Spoel, and van Drunen 1995, Serlet, Boynton, and Tevanian 1996, Gilmer et al. 2017, Kipf and Welling 2016), designing two methods for communication between tasks, in which messages can be passed between any two nodes in a direct or an indirect way (Figure 1-(c)). Importantly, the strength of the relatedness is learned dynamically, rather than being pre-specified, which allows tasks to selectively share information when needed.

We evaluate our proposed models on two types of sequence learning tasks, text classification and sequence tagging, both well-studied NLP tasks (Li and Zong 2008, Liu, Qiu, and Huang 2017, Yang, Salakhutdinov, and Cohen 2016). Moreover, we conduct experiments in both the multi-task setting and in the transfer learning setting to demonstrate that the shared knowledge learned by our models can be useful for new tasks. Our experimental results not only show the effectiveness of our methods in terms of reduced error rates, but also provide good interpretablility of the shared knowledge.

The contributions of this paper can be summarized as follows:

1. We explore the problem of learning the relationship between multiple tasks and formulate this problem as message passing over a graph neural network.
2. We present a state-of-the-art approach that allows multiple tasks to communicate dynamically rather than following a pre-defined structure.
3. Different from traditional black-box learned models, this paper makes a step towards learning **transferable** and **interpretable** representations, which enables us to know what types of patterns are shared.

## Message Passing Framework for Multi-task Communication

We propose to use graph neural networks with message passing to deal with the problem of multi-task sequence learning. Two well-studied sequence learning tasks, text classification and sequence tagging, are used in our experiments. We denote the text sequence as $X = \{x_1, x_2, \dots, x_T\}$ and the output as $Y$. In text classification, $Y$ is a single label; whereas in sequence labelling, $Y = \{y_1, y_2, \dots, y_T\}$ is a sequence.

Assuming that there are $K$ related tasks, we refer to $D_k$ as a dataset with $N_k$ samples for task $k$. Specifically,

$$D_k = \{(X_i^{(k)}, Y_i^{(k)})\}_{i=1}^{N_k}, \tag{1}$$

where $X_i^{(k)}$ and $Y_i^{(k)}$ denote a sentence and a corresponding label sequence for task $k$, respectively. The goal is to learn a neural network to estimate the conditional probability $P(Y|X)$.

Generally, when combining multi-task learning with sequence learning, two kinds of interactions should be modelled: the first is the interactions between different words within a sentence, and the other is the interactions across different tasks.

For the first type of interaction (**interaction of words within a sentence**), many models have been proposed by applying a composition function in order to obtain representation of the sentence. Typical choices for defining the composition function include recurrent neural networks (Hochreiter and Schmidhuber 1997), convolutional neural networks (Kalchbrenner, Grefenstette, and Blunsom 2014), and tree-structured neural networks (Tai, Socher, and Manning 2015). In this paper, we adopt the LSTM architecture to learn the dependencies within a sentence, due to their impressive performance on many NLP tasks (Cheng, Dong, and Lapata 2016). Formally, we refer to $\mathbf{h}_t$ as the hidden state of the word at time $t$, $w_t$. Then, $\mathbf{h}_t$ can be computed as:

$$\mathbf{h}_t = \mathbf{LSTM}(\mathbf{x}_t, \mathbf{h}_{t-1}, \theta). \tag{2}$$

Here, the $\theta$ represents all the parameters of LSTM.

For the second type of interaction (**interaction across different tasks**), we propose to conceptualize tasks and their interactions as a graph, and utilize message passing mechanisms to allow them to communicate. Our framework is inspired by the idea of message passing, which is used ubiquitously in modern computer software (Berendsen, van der Spoel, and van Drunen 1995) and programming languages (Serlet, Boynton, and Tevanian 1996). The general idea of this framework is that we provide a graph network that allows different tasks to cooperate and interact with one another. Below, we describe our conceptualization of the graph construction process, then we describe the message passing mechanism used for inter-task communication.

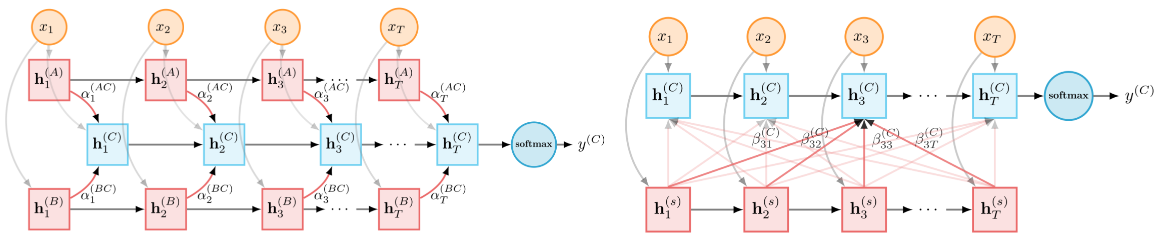
Formally, a graph $G$ can be defined as an ordered pair $G = (V, E)$, where $V$ is a set of nodes $\{V_1, \dots, V_m\}$ and $E$ is a set of edges. In this work, we use directed graphs to model the communication flows between tasks and an edge is therefore defined as an ordered set of two nodes $(V_i, V_j), i \neq j$.

In our models, we represent each task as a node. In addition, we allow virtual nodes to be introduced. These virtual nodes do not correspond to a task. Rather, their purpose is to facilitate communication among different tasks. Intuitively, the virtual node functions as a mailbox, storing messages from other nodes and distributing them as needed.

Tasks and virtual nodes are connected by weighted edges, which represent communication between different nodes. Previous flat and hierarchical architectures for multi-task learning can be considered as graphs with fixed edge connections. Our models dynamically learn the weight of each edge, which allows the models to adjust the strength of the communication signals.

### Message Passing

In our graph structures, we use directed edges to model the communication between tasks. In other words, nodes communicate with each other by sending and receiving messages over edges. Given a sentence with a sequence of words $(w_1^k, ..., w_T^k)$ from task $k$, we use $\mathbf{r}_t^k$ to represent the **aggregated messages** that the word of task $k$ at time step $t$ can get, and we use $\mathbf{h}_t^k$ to denote the task-dependent hidden representation of the word $w_t^k$.



(a) DC-MTL (b) IC-MTL

Figure 2: Two frameworks for multi-task communication. (a) shows how task A and B send information to task C. Here, we only present the information flow relating to task C and omit some edges for easy understanding. (b) shows how the mailbox (shared layer) sends information to task C. The $\alpha$ and $\beta$ values correspond to the edge weights. $\alpha$ controls the strength of the association from a word in a source task to a word in the target task while $\beta$ controls the amount of information we should take from the shared layer. For example, $\beta_{32}^{C}$ can be understood as: to compute the hidden state $\mathbf{h}_3$ of task C, the amount of information should be taken from the second hidden state of shared layers is $\beta_{32}^{C}$.

Below, we propose two basic communication architectures for message passing, which differ according to whether they allow direct communication between any pair of tasks (DC-MTL), or whether the communication is mediated by an intermediate virtual node (IC-MTL).

**1. Direct Communication for Multi-Task Learning (DC-MTL):** in this model, each node can directly send (or receive) messages to (or from) any other nodes. Specifically, as shown in Fig.2-(a), we first assign each task a task-dependent LSTM layer. Each sentence in task $k$ can be passed to all the other task-dependent LSTMs[1] to get corresponding representations $\mathbf{h}_t^{(i)}$, $i = 1 \ldots K$, $i \neq k$. Then, these messages will be aggregated as:

$$\mathbf{r}_t^{(k)} = \sum_{i=1 \ldots K s.t. i \neq k} \alpha_t^{(i \to k)} \mathbf{h}_t^{(i)} \tag{3}$$

Here, $\alpha_t^{(ki)}$ is a scalar, which controls the relatedness between two tasks $k$ and $i$, and can be dynamically computed as:

$$\mathbf{s}_t^{(i \to k)} = f(\mathbf{x}_t^{(k)}, \mathbf{h}_{t-1}^{(k)}, \mathbf{h}_t^{(i)}) \tag{4}$$

$$= \mathbf{u}^{(s)} \tanh(\mathbf{W}^{(s)}[\mathbf{x}_t, \mathbf{h}_{t-1}^{(k)}, \mathbf{h}_t^{(i)}]) \tag{5}$$

where $\mathbf{u}^{(s)}$ and $\mathbf{W}^{(s)}$ are learnable parameters. And the relatedness scores will be normalized into a probability distribution:

$$\boldsymbol{\alpha}_t = \text{softmax}(\mathbf{s}_t) \tag{6}$$

**2. Indirect Communciation for Multi-Task Learning (IC-MTL):** the potential limitation of our proposed DC-MTL model lies in its computational cost, because the number of pairwise interactions grows quadratically with the number of tasks. Inspired by the mailbox idea used in the traditional message passing paradigm (Netzer and Miller 1995), we introduce an extra virtual node into our graph to address this problem. In this setting, messages are not sent directly from one node to another, but are bridged by the virtual node. Intuitively, the virtual node stores the shared messages across all the tasks; different tasks can put messages into this global space, then other tasks can take out the useful messages for themselves from the same space. Fig.2-(b) shows how one task collects information from the mailbox (shared layer).

[1]At training time, the loss is only calculated and used to compute the gradient for the task from which the sentence is drawn.

In details, we introduce an extra LSTM to act as the virtual node, whose parameters are shared across tasks. Given a sentence from task $k$, its information can be written into the shared LSTM by the following operation:

$$\mathbf{h}_t^{(s)} = \mathbf{LSTM}(x_t^{(k)}, \mathbf{h}_{t-1}^{(s)}, \theta^{(s)}), \tag{7}$$

where $\theta^{(s)}$ denotes the parameters are shared across all the tasks.

Then, the aggregated messages at time $t$ can be read from the shared LSTM:

$$\mathbf{r}_t^{(k)} = \sum_{i=1}^{T} \beta_{t \to i}^{(k)} \mathbf{h}_i^{(s)}, \tag{8}$$

where $T$ denotes the length of the sentence and $\beta_{t \to i}^{(k)}$ is used to retrieval useful messages from the shared LSTM, which can be computed in a similar fashion to Equations 5 and 6.

Once the graphs and message passing between the nodes are defined, the next question to ask is how to update the task-dependent representation $\mathbf{h}_t^{(k)}$ for node $k$ using the current input information $\mathbf{x}_t$ and the aggregated messages $\mathbf{r}_t^{(k)}$. We employ a gating unit that allows the model to decide how many aggregated messages should be used for the target tasks, which avoids unnecessary information redundancy. Formally, the $\mathbf{h}_t^{(k)}$ can be computed as:

$$\mathbf{h}_t^{(k)} = \mathbf{LSTM}^{\dagger}(\mathbf{x}_t, \mathbf{h}_{t-1}^{(k)}, \mathbf{r}_t^{(k)}, \theta^{(k)}, \theta^{(s)}). \tag{9}$$

The function $\mathbf{LSTM}^{\dagger}$ is the same as Eq.2 except that we replace the memory update step of the inner function in Eq.2 with the following equation:

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t + \mathbf{g}_t \odot (\mathbf{W}_f^{(s)} \mathbf{r}_t)), \tag{10}$$

where $\mathbf{W}_f^{(s)}$ is a parameter matrix, and $\mathbf{g}_t$ is a fusion gate that selects the aggregated messages. $\mathbf{g}_t$ is computed as follows:

$$\mathbf{g}_t = \sigma(\mathbf{W}_r^{(s)}\mathbf{r}_t^{(k)} + \mathbf{W}_c^{(s)}\mathbf{c}_t), \tag{11}$$

where $\mathbf{W}_r^{(s)}$ and $\mathbf{W}_c^{(s)}$ are parameter matrices.

## Task-dependent Layers

Given a sentence $X$ from task $k$ with its label $Y$ (note $Y$ is either a classification label or sequential labels) and its feature vector $\mathbf{h}^{(k)}$ emitted by the two communication methods above, we can adapt our models to different tasks by using different task-specific layers. We call the task-specific layer as the OUTPUT-LAYER. For text classification tasks, the commonly used OUTPUT-LAYER is a softmax layer, while for sequence labelling tasks, it can be a conditional random field (CRF) layer. Finally, the output probability $P(Y|X)$ can be computed as:

$$P(Y|X) = \text{OUTPUT-LAYER}(X, \mathbf{h}_T^{(k)}, \theta^{(k)}). \tag{12}$$

Then, we can maximize the above probability to optimize the parameters of each task:

$$\mathcal{L}_k(X, Y, \theta^{(k)}, \theta^{(s)}) = -P(Y|X). \tag{13}$$

Once the loss function for a specific task is defined, we could compute the overall loss for our multi-task models as the following:

$$\mathcal{L} = \sum_{k=1}^{K} \lambda_k \mathcal{L}_k(X, Y, \theta^{(k)}, \theta^{(s)}) \tag{14}$$

where $\lambda_k$ is the weight for task $k$. The parameters of the whole network are trained over all datasets and the overall training procedure is presented in Algorithm 1.

## Experiments and Results

In this section, we describe our hyperparameter settings and present the empirical performance of our proposed models on two types of multi-task learning datasets, first on text classification, then on sequence tagging. Each dataset contains several related tasks.

## Hyperparameters

The word embeddings for all of the models are initialized with the 200-dimensional GloVe vectors (840B token version (Pennington, Socher, and Manning 2014)). The other parameters are initialized by randomly sampling from the uniform distribution of $[-0.1, 0.1]$. The mini-batch size is set to 8.

For each task, we take the hyperparameters which achieve the best performance on the development set via a grid search over combinations of the hidden size $[100, 200, 300]$ and $l_2$ regularization $[0.0, 5E-5, 1E-5]$. Additionally, for text classification tasks, we set an equal lambda for each task; while for tagging tasks, we run a grid search of lambda in the range of $[1, 0.8, 0.5]$ and take the hyperparameters which achieve the best performance on the development set.

**Algorithm 1** Training Process for Multi-task Learning over Graph Structures

```
Require: A set of training tasks $\{\mathcal{D}^{\mathcal{T}_i}\}_{i=1}^{K}$, where $\mathcal{T}_i$ is drawn from $p(\mathcal{T})$
 1: Initialize $\Theta := \{\theta^{(s)}, \theta^{(k)}\}$
 2: while not done do
 3:     for $\mathcal{T}_k \sim p(\mathcal{T})$ do
 4:         Sample a batch of samples $\mathcal{B}^{\mathcal{T}_k} = \{X, Y\} \in \mathcal{D}^{\mathcal{T}_k}$
 5:         // Message Passing
 6:         if Direct-Communication then
 7:             for $\mathcal{T}_j \sim p(\mathcal{T})$ do                ▷ $\mathcal{T}_k \neq \mathcal{T}_j$
 8:                 $\mathbf{h}_t^{(j\rightarrow k)} = \mathbf{LSTM}(\mathcal{B}^{\mathcal{T}_k}, \mathbf{h}_{t-1}, \theta^{(k)})$
 9:             end for
10:             $\mathbf{r}_t^{(k)} = \sum_j \alpha_t^{(j\rightarrow k)} \mathbf{h}_t^{(j)}$                ▷ Aggregating
11:         else if Indirect-Communication then
12:             $\mathbf{h}_t^{(s)} = \mathbf{LSTM}(\mathcal{B}^{\mathcal{T}_k}, \mathbf{h}_{t-1}^{(s)}, \theta^{(s)})$
13:             $\mathbf{r}_t^{(k)} = \sum_{i=1}^{T} \beta_{i\rightarrow t}^{(k)} \mathbf{h}_i^{(s)}$                ▷ Aggregating
14:         end if
15:         // Node Updating
16:         $\mathbf{h}_t^{(k)} = \mathbf{LSTM}^{\dagger}(\mathbf{x}_t, \mathbf{h}_{t-1}^{(k)}, \mathbf{r}_t^{(k)}, \theta^{(k)}, \theta^{(s)})$
17:         // Task-dependent Output
18:         $P(Y|X) = \text{OUTPUT-LAYER}(X, \mathbf{h}_T^{(k)}, \theta^{(k)})$
19:     end for
20: end while
```

Based on the validation performance, we choose the size of hidden state as 200 and $l_2$ as 0.0. We apply stochastic gradient descent with the diagonal variant of AdaDelta for optimization (Zeiler 2012).

## Text Classification

To investigate the effectiveness of multi-task learning, we experimented with 16 different text classification tasks involving different popular review corpora, such as books, apparel and movie (Liu, Qiu, and Huang 2017). Each sub-task aims at predicting a correct sentiment label (positive or negative) for a given sentence. All the datasets in each task are partitioned into training, validating, and testing with the proportions of 1400, 200 and 400 samples respectively.

We choose several relevant and representative models as baselines.

- MT-CNN: This model is proposed by (Collobert and Weston 2008) with a convolutional layer, in which lookup-tables are shared partially while other layers are task-specific.
- FS-MTL: Fully shared multi-task learning framework. Different tasks fully share a neural layer (LSTM).
- SP-MTL: Shared-private multi-task learning framework with adversarial learning (Liu, Qiu, and Huang 2017). Different tasks not only have common layers to share information, but have their own private layers.

**Results on Multi-task Learning:** The experimental results show that our proposed models outperform all single-task baselines by a large margin, and here we show the averaged error due to the following reasons: 1) it is easier to show the performance gain of multi-task learning models over single task models. 2) BiLSTM and stacked LSTM are

| Task | Single Task | Multiple Tasks | | | | | Transfer | |
|---|---|---|---|---|---|---|---|---|
| | Avg. | MT-CNN | FS-MTL | SP-MTL | DC-MTL* | IC-MTL* | SP-MTL | IC-MTL* |
| Books | 19.2 | $15.5_{(-3.7)}$ | $17.5_{(-1.7)}$ | $16.0_{(-3.2)}$ | $13.3_{(-5.9)}$ | $13.8_{(-5.4)}$ | $16.3_{(-2.9)}$ | $14.5_{(-4.7)}$ |
| Electronics | 21.4 | $16.8_{(-4.6)}$ | $14.3_{(-7.1)}$ | $13.2_{(-8.2)}$ | $11.5_{(-9.9)}$ | $11.5_{(-9.9)}$ | $16.8_{(-4.6)}$ | $13.8_{(-7.6)}$ |
| DVD | 19.9 | $16.0_{(-3.9)}$ | $16.5_{(-3.4)}$ | $14.5_{(-5.4)}$ | $13.5_{(-6.4)}$ | $12.0_{(-7.9)}$ | $14.3_{(-5.6)}$ | $14.0_{(-5.9)}$ |
| Kitchen | 20.1 | $16.8_{(-3.3)}$ | $14.0_{(-6.1)}$ | $13.8_{(-6.3)}$ | $12.3_{(-7.8)}$ | $11.8_{(-8.3)}$ | $15.0_{(-5.1)}$ | $12.8_{(-7.3)}$ |
| Apparel | 15.7 | $16.3_{(+0.6)}$ | $15.5_{(-0.2)}$ | $13.0_{(-2.7)}$ | $13.0_{(-2.7)}$ | $12.5_{(-3.2)}$ | $13.8_{(-1.9)}$ | $13.5_{(-2.2)}$ |
| Camera | 14.6 | $14.0_{(-0.6)}$ | $13.5_{(-1.1)}$ | $10.8_{(-3.8)}$ | $10.5_{(-4.1)}$ | $11.0_{(-3.6)}$ | $10.3_{(-4.3)}$ | $11.0_{(-3.6)}$ |
| Health | 17.8 | $12.8_{(-5.0)}$ | $12.0_{(-5.8)}$ | $11.8_{(-6.0)}$ | $10.5_{(-7.3)}$ | $10.5_{(-7.3)}$ | $13.5_{(-4.3)}$ | $11.0_{(-6.8)}$ |
| Music | 23.0 | $16.3_{(-6.7)}$ | $18.8_{(-4.2)}$ | $17.5_{(-5.5)}$ | $14.8_{(-8.2)}$ | $14.3_{(-8.7)}$ | $18.3_{(-4.7)}$ | $14.8_{(-8.2)}$ |
| Toys | 16.3 | $10.8_{(-5.5)}$ | $15.5_{(-0.8)}$ | $12.0_{(-4.3)}$ | $11.0_{(-5.3)}$ | $10.8_{(-5.5)}$ | $11.8_{(-4.5)}$ | $10.8_{(-5.5)}$ |
| Video | 17.0 | $18.5_{(+1.5)}$ | $16.3_{(-0.7)}$ | $15.5_{(-1.5)}$ | $13.0_{(-4.0)}$ | $14.0_{(-3.0)}$ | $14.8_{(-2.2)}$ | $13.5_{(-3.5)}$ |
| Baby | 15.9 | $12.3_{(-3.6)}$ | $12.0_{(-3.9)}$ | $11.8_{(-4.1)}$ | $10.8_{(-5.1)}$ | $11.3_{(-4.6)}$ | $12.0_{(-3.9)}$ | $11.5_{(-4.4)}$ |
| Magazines | 10.5 | $12.3_{(+1.8)}$ | $7.5_{(-3.0)}$ | $7.8_{(-2.7)}$ | $8.0_{(-2.5)}$ | $7.8_{(-2.7)}$ | $9.5_{(-1.0)}$ | $8.8_{(-1.7)}$ |
| Software | 14.7 | $13.5_{(-1.2)}$ | $13.8_{(-0.9)}$ | $12.8_{(-1.9)}$ | $10.3_{(-4.4)}$ | $12.8_{(-1.9)}$ | $11.8_{(-2.9)}$ | $11.0_{(-3.7)}$ |
| Sports | 17.3 | $16.0_{(-1.3)}$ | $14.5_{(-2.8)}$ | $14.3_{(-3.0)}$ | $12.3_{(-5.0)}$ | $13.3_{(-4.0)}$ | $13.5_{(-3.8)}$ | $12.8_{(-4.5)}$ |
| IMDB | 17.3 | $13.8_{(-3.5)}$ | $17.5_{(+0.2)}$ | $14.5_{(-2.8)}$ | $13.0_{(-4.3)}$ | $13.5_{(-3.8)}$ | $13.3_{(-4.0)}$ | $13.3_{(-4.0)}$ |
| MR | 26.9 | $25.5_{(-1.4)}$ | $25.3_{(-1.6)}$ | $23.3_{(-3.6)}$ | $21.5_{(-5.4)}$ | $22.0_{(-4.9)}$ | $23.5_{(-3.4)}$ | $22.8_{(-4.1)}$ |
| AVG | 18.0 | $15.5_{(-2.5)}$ | $15.3_{(-2.7)}$ | $13.9_{(-4.1)}$ | **12.5**$_{(-5.5)}$ | $12.7_{(-5.3)}$ | $14.3_{(-3.7)}$ | $13.1_{(-4.9)}$ |

Table 1: Text classification error rates of our models on 16 datasets against typical baselines. The smaller values indicate better performances. The column of **Single Task** (Avg.) gives the average error rates of vanilla LSTM, bidirectional LSTM, and stacked LSTM while the column of **Multiple Tasks** shows the results achieved by corresponding multi-task models. The **Transfer** column lists the results of different models on transfer learning. "*" indicates our proposed models. The numbers in brackets represent the improvements relative to the average performance (Avg.) of three single task baselines.

also the necessary baselines for IC-MTL, since the combination of shared and private layers in IC-MTL is similar to two-layer LSTM.

Table 1 shows the overall results on the 16 different tasks under three settings: single task, multiple task, and transfer learning. Generally, we can see that almost all tasks benefit from multi-task learning, which boosts the performance by a large margin. Specifically, DC-MTL achieves the best performance, surpassing SP-MTL by 1.4%, which suggests that explicit communication makes it easier to shared information. Although the improvement of IC-MTL is not as large as DC-MTL, IC-MTL is efficient to train. Additionally, the comparison between IC-MTL and SP-MTL shows the effectiveness of selectively sharing schema. Moreover, we may further improve our models by incorporating the adversarial training mechanism introduced in SP-MTL, as it is an orthogonal innovation to our methods.

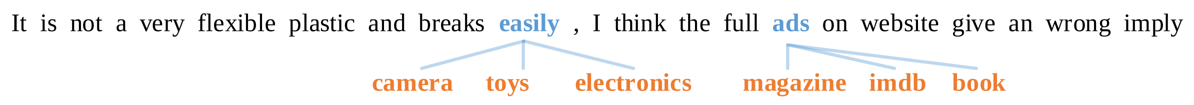
**Evaluation on Transfer Learning:** We next present the potential of our methods on transfer learning, as we expect that the shared knowledge learned by our model architectures can be useful for new tasks. In particular, the virtual node in the IC-MTL model can condense shared information into a common space after multi-task learning, which allows us to transfer this knowledge to new tasks. In order to test the transferability of the shared knowledge learned by IC-MTL, we design an experiment following the supervised pre-training paradigm. Specifically, we adopt a 16-fold "leave-one-task-out" paradigm; we take turns choosing 15 tasks to train our model via multi-task learning, then the learned shared layer is transferred to a second network that is used to test on the remaining *target task* $k$. The parameters of the transferred layer are kept frozen, and the remaining parameters of the new network are randomly initialized.

Table 1 shows these results in the "`Transfer`" column, in which the task in each row is regarded as the target task. We observe that our model achieves a $4.9\%$ average improvement in terms of the error rate over the single tasking setting (13.1 vs. 18.0), surpassing SP-MTL by $1.2\%$ in average (13.1 vs. 14.3). This improvement suggests that our retrieval method with the selective mechanism (the attention layer in eq. 8) is more efficient in finding the relevant information from the shared space compared to SP-MTL, which reads the shared information without any selective mechanism and ignores the relationship between tasks.

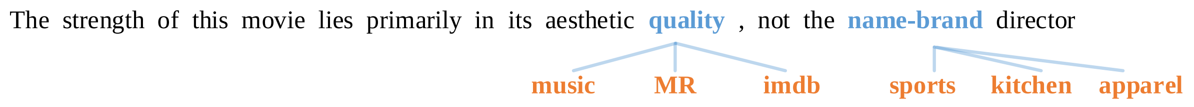
## Sequence Tagging

In this section, we present the results of our models on the second task of sequence tagging. We conducted experiments by following the same settings as (Yang, Salakhutdinov, and Cohen 2016). We use the following benchmark datasets in our experiments: Penn Treebank (PTB) POS tagging, CoNLL 2000 chunking, CoNLL 2003 English NER. The statistics of the datasets are described in Table 3.

**Results and Analysis:** Table 4 shows the performance of the models on the sequence tagging tasks. DC-MTL and IC-MTL significantly outperform the three strong multi-task learning baselines, Specifically, IC-MTL achieves a performance gain of $0.53\%$ in terms of F1 score over the best competitor FS-MTL on the CoNLL2003 dataset, indicating that our models are able to make use of the shared information by modelling the relationship between different tasks. Our models also achieve slightly better F1 scores on the CoNLL2000 and PTB datasets when compared to the best baseline model FS-MTL.


It is not a very flexible plastic and breaks easily , I think the full ads on website give an wrong imply
camera
toys
electronics
magazine
imdb
book

(a) Kitchen Task


The strength of this movie lies primarily in its aesthetic quality , not the name-brand director
music
MR
imdb
sports
kitchen
apparel

(b) Video Task

Figure 3: Illustrations of the most relevant tasks (top 3) for each word in "Kitchen" and "Video" tasks. Here we choose some typical words to visualize in blue. And the orange words represent the relevant tasks.

| | Interpretable Sub-Structures | | | | Explanations |
|---|---|---|---|---|---|
| **Short-term** | movie<br>senti-words | works<br>works adv-words | product<br>adj-words | name<br>a brand | Interpretable phrases to be shared across tasks |
| **Long-term** | higher<br>but higer than | to<br>never too | who<br>like ? | how<br>get made | Interpretable sentence patterns, they usually determine the sentence meanings. |

Table 2: Multiple interpretable sub-structures learned by shared layers. "senti-words" refers to "boring, interesting, amazing" etc. "adv-words" refers to "easily, fine, great" etc. "adj-words" refers to "stable, great, fantastic" etc. These structures show that short-term and long-term dependencies between different words can be learned, which usually control the sentiment of corresponding sentences.

| **Dataset** | **Task** | **Training** | **Dev.** | **Test** |
|---|---|---|---|---|
| CoNLL 2000 | Chunking | 211,727 | - | 47,377 |
| CoNLL 2003 | NER | 204,567 | 51,578 | 46,666 |
| PTB | POS | 912,344 | 131,768 | 129,654 |

Table 3: The sizes of the sequence labelling datasets in our experiments, in terms of the number of tokens.

| **Model** | CoNLL2000 | CoNLL2003 | PTB |
|---|---|---|---|
| LSTM + CRF | 94.46 | 90.10 | 97.55 |
| MT-CNN | 94.32 | 89.59 | 97.29 |
| FS-MTL | 95.41 | 90.94 | 97.55 |
| SP-MTL* | 95.27 | 90.90 | 97.49 |
| DC-MTL | 95.49 | 91.25 | 97.61 |
| IC-MTL | **95.61** | **91.47** | **97.69** |

Table 4: Performance of different models on Chunking, NER, and POS respectively. All the results without marks are reported in the corresponding paper. LSTM+ CRF: Proposed by (Huang, Xu, and Yu 2015). MT-CNN: Proposed by (Collobert and Weston 2008). FS-MTL: Proposed by (Yang, Salakhutdinov, and Cohen 2016).

## Discussion and Qualitative Analysis

In order to obtain more insights and detailed interpretability of how messages are passed between tasks in our proposed models, we design a series of experiments targeting the following aspects:

1. Can the relationship between different tasks be learned by DC-MTL?
2. Are there interpretable structures that the shared layer in IC-MTL can learn? Are these shared patterns similar to linguistic structures, and can they be transferred for other tasks?

**Explicit Relationship Learning** To answer the first question, we visualize the weight $\alpha_t$ of DC-MTL in equation 3. As each task can receive messages from any other task in DC-MTL, $\alpha_t$ directly indicates the relevance of other tasks to the current task at time step $t$. As shown in Figure 3, we analyze the relationships learned by our models on randomly sampled sentences from different tasks. We find that the relationship between tasks cannot be modelled by a static score. Rather, it depends on the specific sample and context. Consider the example sentence in Figure 3-(a), drawn from the KITCHEN task. Here, the words "easily" and "ads" are influenced by different sets of external tasks, in which those words express sentiment. For example, in the CAMERA and TOYS tasks, "`breaks easily`" is usually used to express negative sentiment, while the word "`ads`" often appears in the MAGAZINE task to express negative sentiment. Figure 3-(b) shows a similar case on "`quality`" and "`name-brand`".

**Interpretable Structure Learning** To answer the second question, we visualize $\beta$ in equation 8 inside the IC-MTL model. As different tasks can read information from shared layers in IC-MTL, visualizing $\beta$ allows us to analyze what kinds of sentence structures are shared. Specifically, each word $w_t^{(k)}$ can receive shared messages: $w_1^{(s)}$ ... $w_T^{(s)}$ and the amount of messages is controlled by the scores $\beta$. To illustrate the interpretable structures learned by the shared layer in IC-MTL, we randomly sample several examples from different tasks and visualize their shared structures. Three random sampled cases are described as in Figure 4.

From the experiments we conducted in visulizing $\beta$ in IC-MTL, we observed the following:

- The proposed model can not only utilize the shared information across different tasks, but can tell us what kinds of features are shared. As shown in Table 2, the short-term and long-term dependencies between different words can be captured. For example, the word "`movie`" is prone to connecting to emotional words, such as "`boring, amazing, exciting`" while "`products`" is more likely to make friends with "`stable, great, fantastic`".
- Comparing Figure 4-(b) and (c), we can see how task SOFTWARE borrows useful information from task KITCHEN. Concretely, the sentence "`I` **`would have`** `to buy the software again`" in the task "`Software`" has negative emotion. In this sentence, the key pattern is "`would have`", which does not appear too much in the training set of SOFTWARE. Fortunately, the training samples in the task KITCHEN provide more hints about this pattern.
- As shown in Figure 4-(a) and (b), the shared layer has learned an informative sentence pattern "`would have to ...`" from the training set of task KITCHEN. This pattern is useful for the sentiment prediction of another task SOFTWARE, which suggests that we can analyze the sharabla patterns in an interpretable way for IC-MTL model.

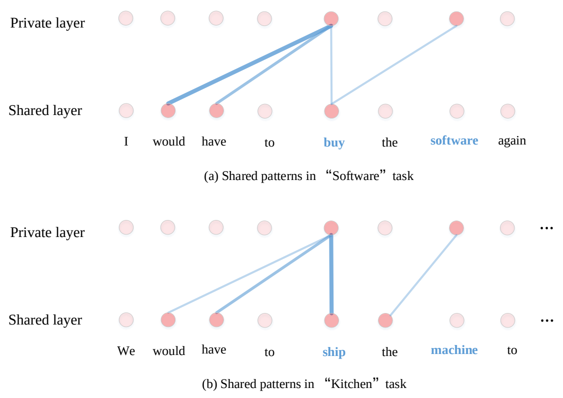


Figure 4: Illustrations of the learned patterns captured by the shared layer under different tasks. The boldness of the line indicates the strength $\beta$ of the relationship. The first sentence comes from the development set of task SOFTWARE while the second one belongs to the training set of task KITCHEN. We choose three typical words "`buy`", "`software`" and "`machine`" to visualize in these sampled sentences.

## Related Work

Neural network-based multi-task frameworks have achieved success on many NLP tasks, such as POS tagging (Yang, Salakhutdinov, and Cohen 2016, Søgaard and Goldberg 2016), parsing (Peng, Thomson, and Smith 2017, Guo et al. 2016), machine translation (Dong et al. 2015, Luong et al. 2015, Firat, Cho, and Bengio 2016), and text classification (Liu, Qiu, and Huang 2016, Liu, Qiu, and Huang 2017). However, previous work does not focus on explicitly modelling the relationships between different tasks. These models are often trained with an opaque neural component, which makes it hard to understand what kind of knowledge is shared. By contrast, in this paper, we propose to explicitly learn the communication between different tasks, and learn some interpretable shared structures.

Before the bloom of neural-based models, **non-neural multi-task learning** methods have also been proposed to model the relationships between tasks. For example, (Bakker and Heskes 2003) learn to cluster tasks by using Bayesian approaches. (Kim and Xing 2010) utilizes a given tree structure to design a regularizer, while (Chen et al. 2010) learns a structured multi-task problem over a given graph. These models adopt complex learning strategies and introduce a priori information between different tasks, which are usually not suitable for sequence modelling. In this paper, we provide a new perspective on how to model the relationships using distributed graph models and *message passing*, which can be learned dynamically rather than following a pre-defined structure.

The technique of message passing is used ubiquitously in computer software (Berendsen, van der Spoel, and van Drunen 1995) and programming languages (Serlet, Boynton, and Tevanian 1996). Recently, there has also been growing interest in developing graph neural networks (Kipf and Welling 2016) or neural message passing algorithms (Gilmer et al. 2017) for learning representations of irregular graph-structured data. In this paper, we formulate multi-task learning as a communication problem over graph structures, allowing different tasks to communicate via message passing.

More recently, (Liu and Huang 2018) propose to learn multi-task communication by explicitly passing gradients. Both our work try to incorporate inductive bias to multi-task learning. However, the difference is that we focus on the structural bias while (Liu and Huang 2018) introduced an additional loss function.

## Conclusion and Outlook

We have explored the problem of learning the relationships between multiple tasks, formulating the problem as message passing over a graph neural network. Our proposed methods explicitly model the relationships between different tasks and achieve improved performance in several multi-task and transfer learning settings. We also show that we can extract interpretable shared patterns from the outputs of our models. From our experiments, we believe that learning interpretable shared structures is a promising direction, which is also very useful for knowledge transfer.

## Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Natural Science Foundation of China (No. 61751201, 61672162), STCSM (No.16JC1420401, No.17JC1404100), and Natural Sciences and Engineering Research Council of Canada (NSERC).

## References

Bakker, B., and Heskes, T. 2003. Task clustering and gating for bayesian multitask learning. *JMLR* 4(May):83–99.

Berendsen, H. J.; van der Spoel, D.; and van Drunen, R. 1995. Gromacs: a message-passing parallel molecular dynamics implementation. *Computer Physics Communications* 91(1-3):43–56.

Chen, X.; Kim, S.; Lin, Q.; Carbonell, J. G.; and Xing, E. P. 2010. Graph-structured multi-task regression and an efficient optimization method for general fused lasso. *stat* 1050:20.

Cheng, J.; Dong, L.; and Lapata, M. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on EMNLP*, 551–561.

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

Dong, D.; Wu, H.; He, W.; Yu, D.; and Wang, H. 2015. Multi-task learning for multiple language translation. In *Proceedings of the ACL*.

Firat, O.; Cho, K.; and Bengio, Y. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of NAACL-HLT*, 866–875.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *ICML*, 1263–1272.

Guo, J.; Che, W.; Wang, H.; and Liu, T. 2016. Exploiting multi-typed treebanks for parsing with deep multi-task learning. *arXiv preprint arXiv:1606.01161*.

Hashimoto, K.; Tsuruoka, Y.; Socher, R.; et al. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on EMNLP*, 1923–1933.

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

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Kalchbrenner, N.; Grefenstette, E.; and Blunsom, P. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*.

Kim, S., and Xing, E. P. 2010. Tree-guided group lasso for multi-task regression with structured sparsity.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Li, Y., and Tian, X. 2015. Graph-based multi-task learning. In *Communication Technology (ICCT), 2015 IEEE 16th International Conference on*, 730–733. IEEE.

Li, S., and Zong, C. 2008. Multi-domain sentiment classification. In *Proceedings of the ACL*, 257–260.

Liu, P., and Huang, X. 2018. Meta-learning multi-task communication. *arXiv preprint arXiv:1810.09988*.

Liu, P.; Qiu, X.; and Huang, X. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of IJCAI*.

Liu, P.; Qiu, X.; and Huang, X. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of ACL*, volume 1, 1–10.

Luong, M.-T.; Le, Q. V.; Sutskever, I.; Vinyals, O.; and Kaiser, L. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Netzer, R. H., and Miller, B. P. 1995. Optimal tracing and replay for debugging message-passing parallel programs. *The Journal of Supercomputing* 8(4):371–388.

Peng, H.; Thomson, S.; and Smith, N. A. 2017. Deep multi-task learning for semantic dependency parsing. In *Proceedings of the 55th ACL*, volume 1, 2037–2048.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. *Proceedings of the EMNLP* 12:1532–1543.

Serlet, B.; Boynton, L.; and Tevanian, A. 1996. Method for providing automatic and dynamic translation into operation system message passing using proxy objects. US Patent 5,481,721.

Søgaard, A., and Goldberg, Y. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of ACL*.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd ACL*, 1556–1566.

Yang, Z.; Salakhutdinov, R.; and Cohen, W. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.

Zeiler, M. D. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.