Scientific Research Publishing

Journal of Software Engineering and Applications, 2019, 12, 199-214
http://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Multi-Task Learning for Semantic Relatedness and Textual Entailment

Linrui Zhang, Dan Moldovan

Department of Computer Science, The University of Texas at Dallas, Richardson, USA
Email: lxz132230@utdallas.edu, Moldovan@utdallas.edu

How to cite this paper: Zhang, L.R. and Moldovan, D. (2019) Multi-Task Learning for Semantic Relatedness and Textual Entailment. *Journal of Software Engineering and Applications*, **12**, 199-214.
https://doi.org/10.4236/jsea.2019.126012

Received: May 13, 2019
Accepted: June 21, 2019
Published: June 24, 2019

Abstract

Recently, several deep learning models have been successfully proposed and have been applied to solve different Natural Language Processing (NLP) tasks. However, these models solve the problem based on single-task supervised learning and do not consider the correlation between the tasks. Based on this observation, in this paper, we implemented a multi-task learning model to joint learn two related NLP tasks simultaneously and conducted experiments to evaluate if learning these tasks jointly can improve the system performance compared with learning them individually. In addition, a comparison of our model with the state-of-the-art learning models, including multi-task learning, transfer learning, unsupervised learning and feature based traditional machine learning models is presented. This paper aims to 1) show the advantage of multi-task learning over single-task learning in training related NLP tasks, 2) illustrate the influence of various encoding structures to the proposed single- and multi-task learning models, and 3) compare the performance between multi-task learning and other learning models in literature on textual entailment task and semantic relatedness task.

Keywords

Deep Learning, Multi-Task Learning, Text Understanding

1. Introduction

Traditional Deep Learning models typically care about optimizing a single metric. We generally train a model for a specific task and then fine-tune the model until the system researches to the best performance [1]. A major problem with this single task learning technique is the data insufficient issue, *i.e.* a model requires a large number of training samples to achieve a satisfied accuracy. In recent years, multi-task learning has provided a good solution to solve this issue.

Inspired by human learning activities where people often apply the knowledge learned from previous tasks to help learn a new task, we would also like to concurrently train multiple related tasks, each of which has limited training samples, within a single model, hoping that the knowledge contained in a task can be leveraged by other tasks [2].

In this paper, we implemented a multi-task learning model to joint learn two related NLP tasks, semantic relatedness and textual entailment, simultaneously. The proposed model contains two parts: a shared representation structure and an output structure. Following the previous research [3], the hard parameter sharing approach is used to build the representation structure, *i.e.* the parameters of the representation layers are shared by both tasks. In the representation structure, a variety of encoding models, such as Recurrent Neural Network (RNN) models and Convolutional Neural Network (CNN) models, encoding contexts, including attention layer, max pooling layer and projection layer, and encoding directions (left-to-right or bi-directional) are implemented. The output structure has two output layers and each of them generates training loss for the corresponding task. The multi-task learning approach can be performed by combing and backpropagating the training losses calculated from the two task specific outputs.

The semantic relatedness (a.k.a. semantic textual similarity) and textual entailment are two related semantic level NLP tasks. The first task measures the semantic equivalence between two sentences. The output is a similarity score scaling from 0 to 5. Higher scores indicate higher similarities between sentences. The second task requires two input sentences as well, a premise sentence and a hypothesis sentence. It measures whether the meaning of the hypothesis sentence can be determined from the premise sentence. There are typically three kinds of results: entailment, contradiction, and neutral, indicating that the meaning of the hypothesis sentence can be determined, contradict, or have nothing to do with the meaning of the premise sentence, respectively.

[4] made the first attempt to propose a joint model to predict outputs of the semantic relatedness and textual entailment tasks. They used a multi-layer Bi-LSTMs to joint five NLP tasks: part-of-speech tagging, chunking, syntactic parsing, semantic relatedness and textual entailment. The lower tasks are trained in lower layers of the multi-layer Bi-LSTMs and are used as auxiliary task to improve the performance of the tasks in higher linguistic level. Their model obtained state-of-the-art or competitive results in literature on the five tasks. Different from their work, the contributions of our paper are as follows:

- Unlike the above-mentioned paper that only evaluates the unidirectional influence from semantic relatedness to textual entailment, our work demonstrates the mutual influence between semantic relatedness task and textual entailment task.
- Compared with previous work that joint the tasks solely with a multi-layer Bi-LSTM structure, our work implemented and evaluated the multi-task

learning model based on a variety of structures with different encoding architectures, encoding contexts and encoding directions, and analyzed the impact of different encoding methods to the proposed single- and multi-task learning models.

- Our system achieved comparative results to state-of-the-art multi-task learning and transfer learning models and outperformed the state-of-the-art unsupervised and feature based supervised machine learning models on the proposed tasks.

Next section will give a brief mathematical background of the deep neural structures as well as some preliminary knowledge of multi-task learning. After that, we will illustrate the main structure of our system and discuss the training process. The experimental details and results are described in section 4. In section 5, we will show the results, including feature ablation, comparative studies between the single- and multi-task learning models, and between our model and other state-of-the-art learning models. At the end, we will offer some conclusions and discuss future works.

2. Preliminaries

This section describes the background knowledge of this paper, including an introduction of different encoding structures (CNNs and RNNs), encoding contexts (attention layer, max pooling layer, and projection layer) and encoding directions (left-to-right or bi-directional), and the preliminary of multi-task learning.

2.1. LSTM Neural Network

Recurrent neural network [5] is the most commonly used deep learning structure to model sequential input data, since it can capture the long-term dependencies of inputs. However, due to the vanishing gradient problem [6], some defects occur if the length of the sequences increases. LSTM neural network [7] have been proposed for overcoming the gradient vanishing problem by using a complex activation unit, LSTM unit, which is described below.

A regular LSTM unit contains five components: an input gate i_t, a forget gate f_t, an output gate o_t, a new memory cell $\tilde{c}_t$, and a final memory cell c_t. Three adaptive gates i_t, f_t, o_t and new memory cell $\tilde{c}_t$ are computed based on the previous state h_{t-1}, current input x_t, and bias term b. The final memory cell c_t is a combination of previous cell content c_{t-1} and new memory cell $\tilde{c}_t$ weighted by the forget gate f_t and input gate i_t. The final output of the LSTM hidden state h_t is computed using the output gate o_t and final memory cell c_t. The mathematical representation of the input gate i_t, forget gate f_t, output gate o_t, new memory cell $\tilde{c}_t$, final memory cell c_t and the final LSTM hidden state h_t is shown in Equations (1) to (6).

$$i_t = \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1}\right)\# \tag{1}$$

$$f_t = \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1}\right) \tag{2}$$

$$o_t = \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1}\right) \tag{3}$$

$$\tilde{c}_t = \tanh\left(W^{(c)}x_t + U^{(c)}h_{t-1}\right) \tag{4}$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t \tag{5}$$

$$h_t = o_t \otimes \tanh\left(c_t\right) \tag{6}$$

Sometimes dependencies in sentences do not just appear from left-to-right and a word can have a dependency on another word before it. In this case, Bidirectional LSTM (Bi-LSTM) [8] is used to read input data from both left-to-right and right-to-left directions.

A Bi-LSTM network could be viewed as a network that maintains two hidden LSTM layers together, one for the forward propagation $\vec{h}_t$ and another for the backward propagation $\overleftarrow{h}_t$ at each time-step *t*. The final prediction $\hat{y}_t$ is generated through the combination of the score results produced by both hidden layers $\vec{h}_t$ and $\overleftarrow{h}_t$. Equation (7) to (9) illustrate the mathematical representations of a Bi-LSTM:

$$\vec{h}_t = f\left(\vec{W}x_t + \vec{V}h_{t-1} + \vec{b}\right) \tag{7}$$

$$\overleftarrow{h}_t = f\left(\overleftarrow{W}x_t + \overleftarrow{V}h_{t-1} + \overleftarrow{b}\right) \tag{8}$$

$$\hat{y}_t = g\left(Uh_t + c\right) = g\left(U\left[\vec{h}_{t-1}; \overleftarrow{h}_{t-1}\right] + c\right) \tag{9}$$

Here, $\hat{y}_t$ is the predication of the Bi-LSTM system. The symbols → and ← indicate directions. *W*, *U* are weight matrices that are associated with input x_t and hidden states x_t. *U* is used to combine the two hidden LSTM layers together, *b* and *c* are bias terms, and $g(x)$ and $f(x)$ are activation functions.

2.2. Attention and Projection Layer

Different parts of an input sentence have different levels of significance. For instance, in sentence "the ball is on the field", the primary information of sentence is carried by the words "ball", "on", and "field". LSTM network, though can handle gradient vanishing issue, still have a bias on the last few words over the words appearing in the beginning or middle of sentences. This is clearly not the natural way that we understand sentences. Attention mechanism [9] is a strategy to aggregate more informative words and ignore less important words in input sentences, and it is used to select important local patterns of inputs for the final representation.

The attention mechanism is calculated in three steps. First, we feed the hidden state h_t through a one-layer perceptron to get u_t which could be viewed as a hidden representation of h_t. We latter multiply u_t with a context vector u_w and normalize results through a *Softmax* function to get the weight a_t of each hidden state h_t. The context vector could be viewed as a high-level vector to select informative hidden state and will be jointly learned during the training

process. The final sentence representation is computed as a sum over of the hidden state h_t and its weights a_t. The calculation steps of u_t and a_t are shown in Equation (10) and Equation (11). The mathematic representation that leads to the final sentence representation S is shown in Equation (12):

$$u_t = \tanh\left(Wh_t + b\right) \tag{10}$$

$$a_t = \frac{e^{u_t^{\mathrm{T}} u_w}}{\sum_t e^{u_t^{\mathrm{T}} u_w}} \tag{11}$$

$$S = \sum_t a_t h_t \tag{12}$$

A projection layer is another optimization layer to connect the hidden states of LSTM units to output layers. It is usually used to reduce the dimensionality of the representation (the LSTM output) without reducing its resolution. There are several implementations of such layers and, in this paper, we select a simple implementation which is a feed forward neural network with one hidden layer.

2.3. Basic CNN

Convolutional Neural Network [10] [11], which can extract high-level features from groups of words, is another commonly used deep learning structure to model input data. In a CNN, a word embedding is represented as $w_i \in R^d$, where i is the i^{th} word in the sentence and d is the dimension of the word embedding. Given a sentence with n words, the sentence can thus be represented as an embedding matrix $W \in R^{n \times d}$.

In the convolutional layer, several filters, also known as kernels, $k \in R^{hd}$ will run over the embedding matrix W and perform convolutional operations to generate features c_i. The convolutional operation is calculated as:

$$c_i = f\left(w_{i:i+h-1} \cdot k^{\mathrm{T}} + b\right) \tag{13}$$

where, $b \in R$ is the bias term and f is the activation function. For instance, a sigmoid function. $w_{i:i+h-1}$ is referred to the concatenation of vectors $w_i, \cdots, w_{i+h-1}$. h is the number of words that a filter is applied to and usually there are three filters with h equals to one, two or three to simulate the uni-gram, bi-gram and tri-gram models, respectively.

$$c = \left[c_1, c_2, \cdots, c_{n-h+1}\right] \tag{14}$$

A convolutional layer is usually followed by a max-pooling layer to select the most significant n-gram features across the whole sentence by applying a max operation $\hat{c} = \max\{c\}$ on each filter.

Inspired by [12], in this paper, we implemented a Hierarchical ConvNet as the representative of CNN models. The Hierarchical ConvNet is a network with many CNN layers in a hierarchical level. Each CNN layer is followed by a max-pooling layer to extract features from the CNN outputs. The final representation of the sentence is the concatenation of the max-pooling outputs in different hierarchical levels.

2.4. Multi-Task Learning

Multi-task Learning is a learning mechanism to improve performance on the current task after having learned a different but related concept or skill on a previous task. It can be performed by learning tasks in parallel while using a shared representation such that what is learned for each task can help other tasks be learned better. This idea can be backtracked to 1998, when [13] used the prediction of different characteristics of road as auxiliary tasks for predicting the steering direction in a self-driving car. In recent years, multi-task learning has been used successfully across all applications of machine learning, including, speech recognition [14] [15] and [16] and computer vision [17] and [18].

In natural language processing, [19] proposed a language model using single convolutional neural network architecture to joint train and output a host of language processing predictions, including part-of-speech tags, chunks, named entity tags, semantic roles, etc. In recent years, researchers focused on combining NLP tasks with hierarchical architectures, *i.e.* different NLP tasks are ranked with their linguistic orders and the low-level tasks are supervised at lower layers of the joint model as auxiliary task to improve the performance of high-level tasks, such works include [20] and [4].

3. Approach

3.1. Problem Formulation

In order to formulate the problem, we first give the definition of Multi-task Learning from [2].

***Definition* 1. (*Multi-Task Learning*)** Given m learning tasks $\{T_i\}_{i=1}^{m}$ where all the tasks or a subset of them are related, multi-task learning aims to help improve the learning of a model for T_i by using the knowledge contained in all or some of the m task.

Based on the definition of Multi-task Learning, we can formulate our problem as $\{T_i\}_{i=1}^{2}$, where $m = 2$ corresponding to the relatedness task (T_1) and entailment task (T_2). Both tasks are supervised learning tasks accompanied by a training dataset D_i consisting of n_i training samples, *i.e.*, $D_i = \left\{x_j^i, y_j^i\right\}_{j=1}^{n_i}$, where $x_j^i \in R^{d_i}$ is the j^{th} training instance in T_i and y_j^i is its label. We denote by X^i the training data matrix for T_i, $X^i = \left(x_1^i, \cdots, x_{n_i}^i\right)$ and Y^i for its label. In our case, the two tasks share the same training instance but with different labels ($X^1 = X^2$ and $Y^1 \neq Y^2$). Our object is to design and train a neural network structure to learn a mapping F: $\left\{X_j^1 \to Y_j^1, Y_j^2\right\}_{j=1}^{n_i}$ or $\left\{X_j^2 \to Y_j^1, Y_j^2\right\}_{j=1}^{n_i}$.

3.2. The System Structure

Following the hard parameter sharing approach, we implemented a feed-forward neural network. The main structure of our system is illustrated in **Figure 1**. It contains three major layers: the input layer, the concatenation layer and the output layer.

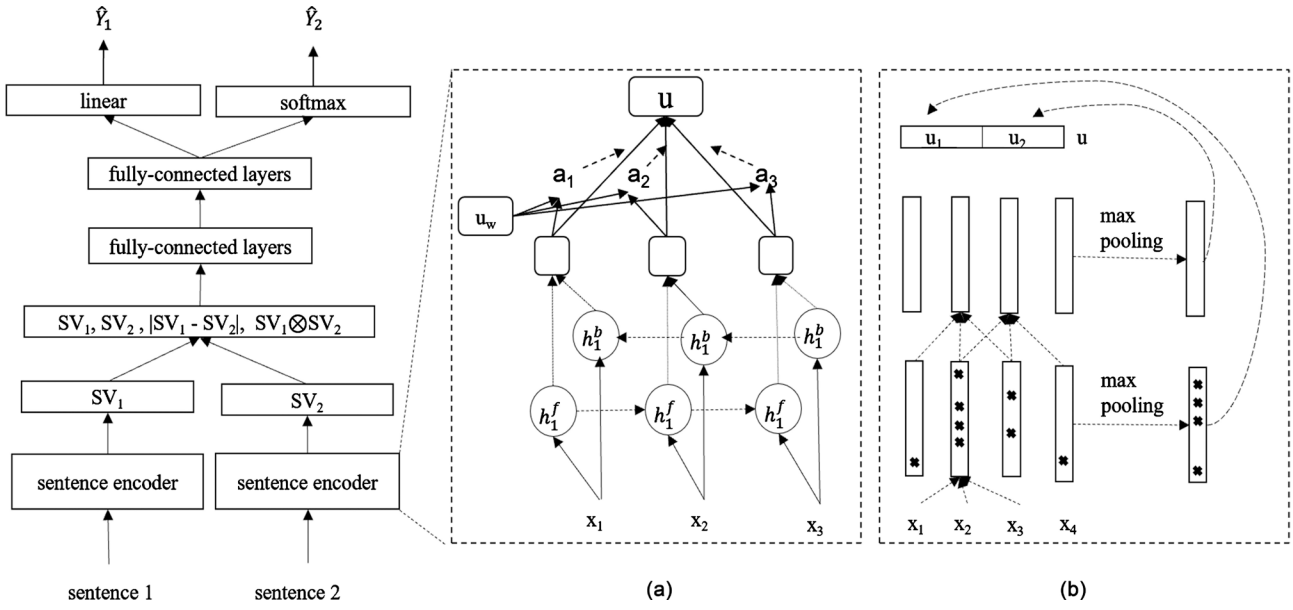

Figure 1. The main structure of our system. (a) Bi-LSTM with attention layer. (b) Hierarchical ConvNet with 2 CNN Layers and tri-gram filter.

In the input layer, two sentence embedding layers will first transform the input sentences into semantic vectors, which can represent the semantic meanings of these sentences, using a variety of encoding structures. The part (a) and (b) of the **Figure 1** show two examples of sentence encoder structures, a Bi-LSTM with attention and a Hierarchical ConvNet with two CNN layers.

Except for the two examples shown in **Figure 1**, we implemented and experimented with a verity of RNN and CNN based structures. Specifically, for the RNN based structures, we implemented a regular LSTM structure and compared its performance with Bi-LSTM structure to show the effect of different encoding directions (left-to-right and bi-directional) to the system. In addition, we added three different encoding layers (attention layer, max pooling layer and projection layer) on top of the Bi-LSTM structure to evaluate the influence of various encoding contexts to the system. For the CNN-based structures, we experimented on different features of the Hierarchical ConvNet, such as different CNN filters (uni-gram, bi-gram and tri-gram filters) and the number of CNN layers (from one to four).

The concatenation layer aims to create a vector that can combine the information of the two sentence vectors. Following the previous research [21], we formed a semantic vector by concatenating the sentence vector pairs, together with the element-wise absolute different and multiplication between them. The concatenated vectors could be represented as (SV1, SV2, |SV1 − SV2|, SV1 ⊗ SV2).

The input layer and the concatenation layer are shared by both tasks. During the training process, the input sentence pairs of both tasks will be processed by these shared layers and the parameters in these shared layers will be affected by both tasks simultaneously.

On top of the shared structure, we build two output layers, one for each task, to generate task specific outputs for the given two tasks. In term of machine learning, the semantic relatedness task is a regression task, so a *linear* function is used as the activation function to generate the relatedness scores between sentence pairs. The textual entailment task is a classification problem, so a *softmax* function is selected as the activation function to generate a probabilistic distribution of the entailment labels between the sentence pairs.

3.3. Training

The system can be learned by jointing and optimizing the two task specific loss functions simultaneously. For the relatedness task (T_1), mean square error loss between the system output y and the ground-true $\hat{y}$ score labeled in the corpora are used as the training loss function. The mathematical formula is:

$$L_{mse} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \tag{15}$$

where, n is the number of training samples, and $i \in [1, n]$ is the index number of the training samples.

For the entailment task (T_2), cross-entropy loss between the system output $\hat{y}$ and the ground-true label y is used as the loss function. The mathematical formula can be described as:

$$L_{se} = -\sum_{i=1}^{n}\sum_{j=1}^{3} y_i^j \log \hat{y}_i^j \tag{16}$$

where, n is the number of training samples, $i \in [1, n]$ is the index number of the training samples and $j \in [1, 3]$ is the index number of class labels.

The joint loss function is obtained by taking a weighted sum of the loss functions of each of the two tasks, which is written as:

$$Loss = \lambda_1 L_{mse} + \lambda_2 L_{ce} \tag{17}$$

where λ_1 and λ_2 are the weights of the loss function of similarity and entailment task and they will be added as hyperparameters during the training process. During the experiments, we first fine-turn the λ in a large range ϵ [0, 10000] and then realize the system can achieve the best performance when λ is narrow down to ϵ [1, 2].

4. Experimental Results

This section shows the experimental results of the proposed model. The details of the experiments, including the use of the corpus, the evaluation metrics and the parameter settings, will be discussed first and the experimental results of the RNN and CNN based models will be shown afterwards.

4.1. Corpus and Evaluation Metrics

The Sentence Involving Compositional Knowledge (SICK) benchmark [22] is used to evaluate the performance of our system. The corpus contains a large

number of sentence pairs with rich lexical, syntactic and semantic phenomena, and a semantic relatedness score and entailment labels are labeled for each sentence pair. An example of the SICK benchmark is shown in Table 1.

We followed the standard split for the training, developing, and testing sets of the corpus. The accuracy is used as the evaluation method for the entailment task. The mathematic representation of the accuracy is:

$$\text{Accuracy} = \frac{N_{\text{correct}}}{N_{\text{total}}} \tag{18}$$

where N_{correct} is the number of examples that has correct entailment labels. The and Pearson correlation coefficient (Pearson's r) is used as the evaluation method for the relatedness task. The mathematic representation of the Pearson's r is:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{19}$$

where X and Y are the predicated and ground true relatedness score of the testing examples. cov is the covariance and σ_X, σ_Y are the standard deviation of X, Y.

4.2. Experiment Settings

The neural network model was trained using the gradient-based optimization *Adam* [23] with the learning rate of 0.01 and backpropagation. The word embeddings are initialized with 300-d Glove embeddings.

For the RNN models, the hidden layer size of LSTM is 128 and the hidden layer size of the first fully connected layer is 128 and 256 corresponding to the LSTM and Bi-LSTM models. The hidden layer size of the second fully connected layer is 64.

For the CNN models, the parameters of the filters are length = 128, stride = 1 and padding = 1, and the layers of the Hierarchical ConvNet is from 1 to 4. The hidden layer size of the fully connected layers is the same as the RNN models. We run a max epoch of 20 and mini-batch of 64. All the experiments were performed using PyTorch [24] on Nvidia GTX 1080 8 GBytes GPU server and Linux 16.04-64 bit based operating system.

4.3. Experimental Results with RNN Models

For each RNN model, we compared between the single- and multi-task learning

Table 1. An example of SICK dataset.

Sentence	Relatedness	Entailment
A: A player is running with the ball. B: Two teams are competing in a football match.	2.6	Neutral
A: A woman is dancing and singing in the train. B: A woman is performing in the rain.	4.4	Entailment
A: Two dogs are wrestling and hugging. B: There is no dog wrestling and hugging.	3.3	Contradiction

models and illustrated the influence of different encoding methods (directions and contexts) to these models. Figure 2 and Figure 3 show the performances of single- and multi-task learning models with different encoding directions (left-to-right or bi-directional) and contexts (attention, max-pooling or projection layers) on textual entailment and semantic relatedness tasks.

4.4. Experimental Results with CNN Models

For the CNN models, we showed the performance a Hierarchical ConvNet with different convolutional layers and filters. Figure 4 and Figure 5 illustrate the performance of the Hierarchical ConvNet with one to four convolutional layers and three convolutional filters (uni-gram, bi-gram and tri-gram filters). CNN-2 means the Hierarchical ConvNet contains 2 convolutional layers.

5. Results Analysis and Comparison

In this section, we will analyze the results of our experiments, including the

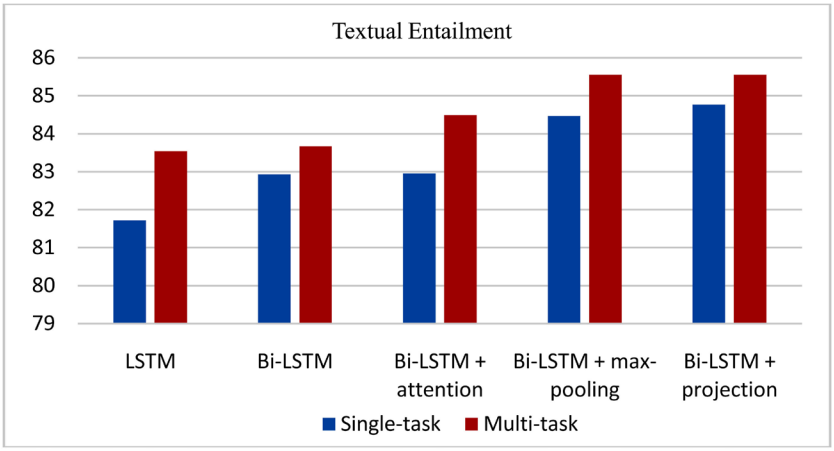

Figure 2. The accuracy of the textual entailment task on the model with different encoding contexts.

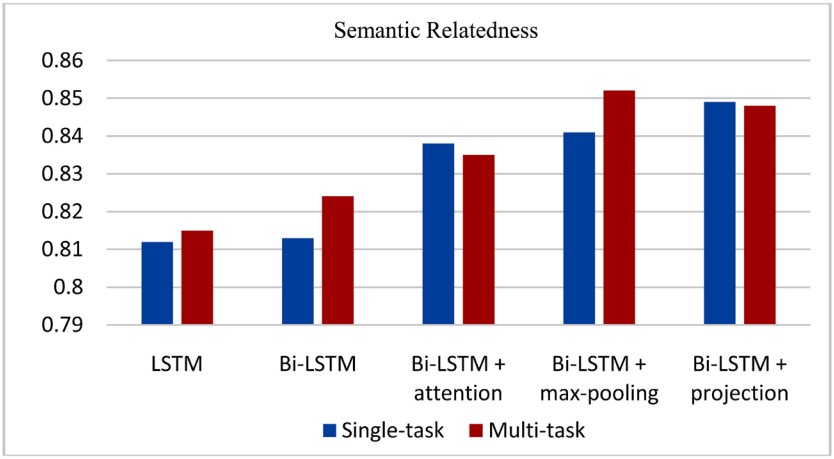

Figure 3. The Pearson's r score of the semantic relatedness task on the model with different encoding contexts.

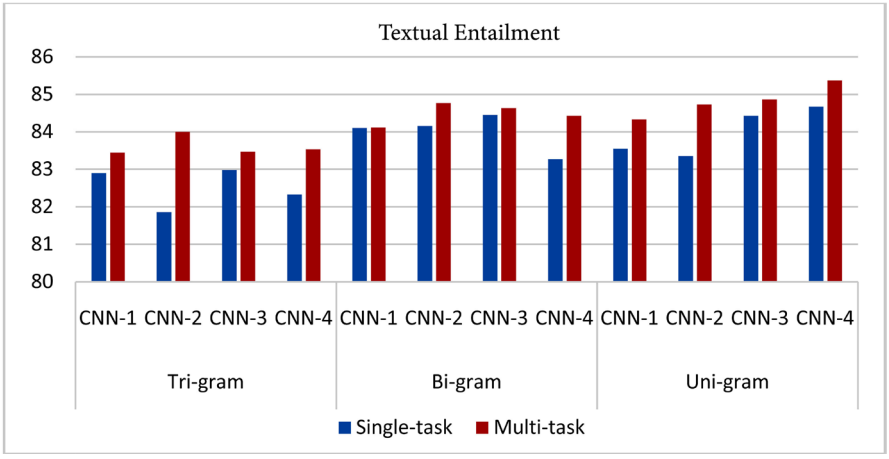

Figure 4. The accuracy of the textual entailment task on the model with different filters and CNN layers.

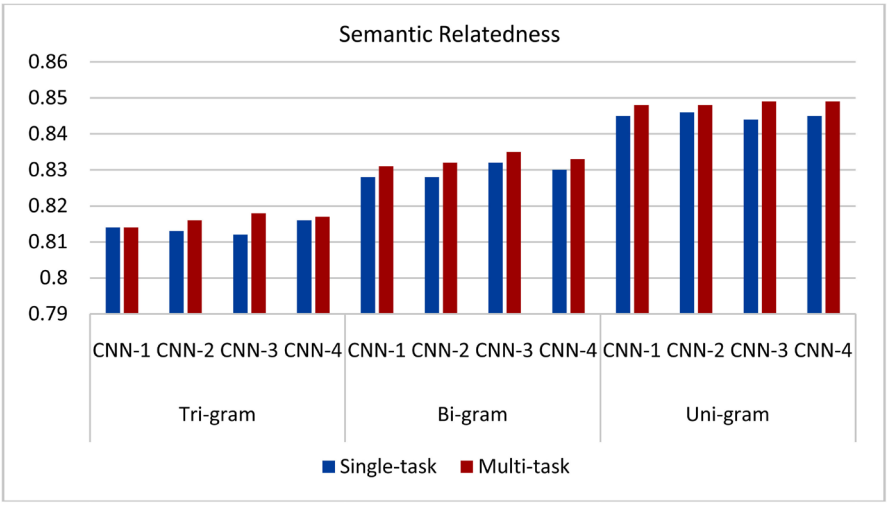

Figure 5. The Pearson's r score of the semantic relatedness task on the model with different filters and CNN layers.

comparisons 1) between the proposed single- and multi-task learning models on the given tasks, 2) among various encoding methods of the proposed RNN and CNN models, and 3) between our multi-task learning model and other state-of-the learning models in literature.

5.1. The Comparison between Single- and Multi-Task Learning Models

From the experiments, it is obvious that multi-task learning can achieve better results than single task learning on both tasks. In addition, we can observe that the performance improvement has a bias on textual entailment task over semantic relatedness task. This observation can be explained by the task hierarchy theory in multi-task learning. In multi-task learning, the common features learned from multiple tasks are usually more sensitive to the high-level tasks than to the low-level tasks. In [4], they assumed that textual entailment task is in

a higher linguistic level than semantic relatedness task and our experimental results are consistent with this assumption.

5.2. The Analysis of RNN Models

Observing that Bi-LSTM performs consistently better than LSTM under every scenario from **Figure 2** and **Figure 3**, we can conclude that bi-directional encoding is a better way to encoding sentences than unidirectional encoding for the given tasks. In addition, we also observe that the proposed encoding contexts (attention layer, max pooling layer and projection layer) can all increase the system performance of the baseline Bi-LSTM model.

Among these encoding contexts, max pooling layer and projection layer can achieve approximately the same performance and can both surpass the performance of attention layer. This is because the limited amount of training data is slightly insufficient to train the proposed model, so the model starts to overfit the training data after the first several iterations of training. Projection layer and max pooling layer can avoid overfitting by reducing the dimensionality of the sentence representation. On the contrary, attention layer is used to select important components of sentences which does not have the ability to overcome overfitting. As a result, projection layer and max pooling layer show a relatively strong performance over attention layer.

5.3. The Analysis of CNN Models

We observe from **Figure 4** and **Figure 5** that uni-gram filter has the best performance compared to bi-gram and tri-gram filters on both single- and multi-task learning models and this indicates that single word is better than group of words in the CNN model for the given tasks.

We also observe that increasing the CNN layers of the Hierarchical ConvNet can hardly improve the system performance. The reason is also overfitting. Even though, increasing the number of CNN layers can gain the representation ability of the system, it also increases the complexity of the system and raises the risk of overfitting.

5.4. Comparison with State-of-the-Art Learning Models

Comparisons can also be made between our system with some of the recent state-of-the art learning models on the same benchmark, including the best supervised learning model Dependency-tree LSTM [25] and the best hand-engineered models Illinois-LH [26], the best unsupervised sentence representation model fastText [27] and SkipThought [28], the best transfer learning model InferSent [29] and the previously mentioned the multitask-learning Joint Model [4]. The results of the comparison are listed in **Table 2**.

From the results, we can observe that our system outperforms the best unsupervised and feature engineered systems in literature on textual entailment task and achieves very competitive results compared to the transfer learning and

Table 2. The system performance of various architectures trained in different ways. Joint Model used mean squared error as the evaluation method for relatedness task, thus are not listed in the table.

Model		Relatedness	Entailment
Unsupervised Model	FastText	0.815	78.3
	SkipThought	0.858	79.5
Feature Enginnerred Model	Dependency-Tree LSTM	0.868	--
	Illinois-LH	--	84.5
Transfer Learning Model	InferSent	0.885	86.3
Multi-task Learning Model	Joint	--	86.8
	Ours-RNN	0.848	85.6
	Ours-CNN	0.849	85.4

multi-task learning models. In addition, the performance of our model on semantic relatedness task is comparable to other models in literature.

The reason that the transfer learning outperforms our models is that transfer learning model takes advantage of knowledge learned from external tasks. For instance, the InferSent system is pre-trained with SNLI dataset, containing 520 K training instances on textual entailment tasks. When being applied to SICK benchmark, the knowledge learned from previous task can be directly transferred to a new task and improved the learning ability of the new task. On the contrary, our models do not rely on previous learned knowledge and were trained absolutely from scratch.

The reason that the state-of-the-art multi-task learning model can outperform our models is that it used a hierarchical architecture. Research [20] has shown that hierarchical architecture is a better way than parallel architecture to combine multiple tasks with different level, because such architecture can strength the influence form the low-level to high-level task and increase the performance of the high-level task. On the other side, parallel architecture allows us to observe the mutual influence between different tasks, instead of solely showing the influence from low-level task to high-level task in hierarchical architecture.

6. Conclusion and Future Work

In this paper, we explored the multi-task learning mechanisms in training related NLP tasks. We performed single- and multi-task learning on textual entailment and semantic relatedness task with a variety of Deep Learning structures. Experimental results showed that learning these tasks jointly can lead to much performance improvement compared with learning them individually.

We believe that this work only scratches the surface of multi-task learning on training related NLP tasks. Larger dataset, better architecture engineering and probably combining pre-training knowledge in the training process could bring

the system performance to the next level.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Ruder, S. (2017) An Overview of Multi-Task Learning in Deep Neural Networks. arXiv: 1706.05098

[2] Zhang, Y. and Yang, Q. (2017) A Survey on Multi-Task Learning. arXiv: 1707.08114

[3] Caruana, R. (1993) Multitask Learning: A Knowledge-Based Source of Inductive Bias. *Proceedings of the Tenth International Conference on Machine Learning*, Amherst, 27-29 June 1993, 41-48. https://doi.org/10.1016/B978-1-55860-307-3.50012-5

[4] Hashimoto, K., Xiong, C., Tsuruoka, Y. and Socher, R. (2017) A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. *Proceedings of the* 2017 *Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark, 7-11 September 2017, 1923-1933. https://doi.org/10.18653/v1/D17-1206

[5] Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986) Learning Representations by Back-Propagating Errors. *Nature*, **323**, 533-536. https://doi.org/10.1038/323533a0

[6] Hochreiter, S., Bengio, Y., Frasconi, P. and Schmidhuber, J. (2001) Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies.

[7] Hochreiter, S. and Schmidhuber, J. (1997) Long Short-Term Memory. *Neural Computation*, **9**, 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

[8] Schuster, M. and Paliwal, K.K. (1997) Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing*, **45**, 2673-2681. https://doi.org/10.1109/78.650093

[9] Lin, Z., Feng, M., dos Santos, C.N., Yu, M., Xiang, B., Zhou, B. and Bengio, Y. (2017) A Structured Self-Attentive Sentence Embedding. arXiv: 1703.03130

[10] Lawrence, S., Giles, C.L., Tsoi, A.C. and Back, A.D. (1997) Face Recognition: A Convolutional Neural-Network Approach. *IEEE Transactions on Neural Networks*, **8**, 98-113. https://doi.org/10.1109/72.554195

[11] Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012) Imagenet Classification with Deep Convolutional Neural Networks. *NIPS*12 *Proceedings of the* 25*th International Conference on Neural Information Processing Systems*, Lake Tahoe, Nevada, 3-6 December 2012, 1097-1105.

[12] Zhao, H., Lu, Z. and Poupart, P. (2015) Self-Adaptive Hierarchical Sentence Model. *Twenty-Fourth International Joint Conference on Artificial Intelligence*, Buenos Aires, Argentina, 25-31 July 2015, 4069-4076.

[13] Caruana, R. (1997) Multitask Learning. *Machine Learning*, **28**, 41-75. https://doi.org/10.1023/A:1007379606734

[14] Kim, S., Hori, T. and Watanabe, S. (2017) Joint CTC-Attention Based End-to-End Speech Recognition Using Multi-Task Learning. 2017 *IEEE International Conference on Acoustics*, *Speech and Signal Processing* (*ICASSP*), New Orleans, LA, USA, 5-9 March 2017, 4835-4839. https://doi.org/10.1109/ICASSP.2017.7953075

[15] Wu, Z., Valentini-Botinhao, C., Watts, O. and King, S. (2015) Deep Neural Net-

works Employing Multi-Task Learning and Stacked Bottleneck Features for Speech Synthesis. 2015 *IEEE International Conference on Acoustics, Speech and Signal Processing* (*ICASSP*), Brisbane, QLD, Australia, 19-24 April 2015, 4460-4464. https://doi.org/10.1109/ICASSP.2015.7178814

[16] Saon, G., Kurata, G., Sercu, T., Audhkhasi, K., Thomas, S., Dimitriadis, D., Cui, X., Ramabhadran, B., Picheny, M., Lim, L.-L., *et al.* (2017) English Conversational Telephone Speech Recognition by Humans and Machines. arXiv: 1703.02136 https://doi.org/10.21437/Interspeech.2017-405

[17] Long, M. and Wang, J. (2015) Learning Multiple Tasks with Deep Relationship Networks. arXiv: 1506.021172

[18] Kendall, A., Gal, Y. and Cipolla, R. (2018) Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 18-23 June 2018, 7482-7491.

[19] Collobert, R. and Weston J. (2008) A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. *Proceedings of the* 25*th International Conference on Machine Learning*, Helsinki, Finland, 5-9 July 2008, 160-167. https://doi.org/10.1145/1390156.1390177

[20] Søgaard, A. and Goldberg, Y. (2016) Deep Multi-Task Learning with Low Level Tasks Supervised at Lower Layers. *Proceedings of the* 54*th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany, 7-12 August 2016, 231-235. https://doi.org/10.18653/v1/P16-2038

[21] Shao, Y. (2017) HCTI at SemEval-2017 Task 1: Use Convolutional Neural Network to Evaluate Semantic Textual Similarity. *Proceedings of the* 11*th International Workshop on Semantic Evaluation* (*SemEval*-2017), Vancouver, Canada, 3-4 August 2017, 130-133. https://doi.org/10.18653/v1/S17-2016

[22] Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S. and Zamparelli, R. (2014) Semeval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment. *Proceedings of the* 8*th International Workshop on Semantic Evaluation* (*SemEval* 2014), Dublin, Ireland, 23-24 August 2014, 1-8. https://doi.org/10.3115/v1/S14-2001

[23] Kingma, D.P. and Ba, J. (2014) Adam: A Method for Stochastic Optimization. arXiv: 1412.6980

[24] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. and Lerer, A. (2017) Automatic Differentiation in Pytorch. 31*st Conference on Neural Information Processing Systems* (*NIPS* 2017), 4-9 December 2017, Long Beach, CA, USA, 1-4.

[25] Tai, K.S., Socher, R. and Manning, C.D. (2015) Improved Semantic Representations from Tree-Structured Long Short Term Memory Networks. arXiv: 1503.00075 https://doi.org/10.3115/v1/P15-1150

[26] Lai, A. and Hockenmaier, J. (2014) Illinois-LH: A Denotational and Distributional Approach to Semantics. *Proceedings of the* 8*th International Workshop on Semantic Evaluation* (*SemEval* 2014), Dublin, Ireland, 23-24 August 2014, 329-334. https://doi.org/10.3115/v1/S14-2055

[27] Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T. (2017) Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, **5**, 135-146. https://doi.org/10.1162/tacl_a_00051

[28] Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R., Urtasun, R., Torralba, A. and Fidler, S. (2015) Skip-Thought Vectors. *Advances in neural information processing*

systems 28 (*NIPS* 2015), Montreal, Canada, 7-12 December 2015, 3294-3302.

[29] Conneau, A., Kiela, D., Schwenk, H., Barrault, L. and Bordes, A. (2017) Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. arXiv: 1705.02364.