# Modelling Sentence Pairs via Reinforcement Learning: An Actor-Critic Approach to Learn the Irrelevant Words

**Mahtab Ahmed, Robert E. Mercer**
Department of Computer Science, University of Western Ontario
London, ON, Canada
mahme255@uwo.ca, mercer@csd.uwo.ca

## Abstract

Learning sentence representation is a fundamental task in Natural Language Processing. Most of the existing sentence pair modelling architectures focus only on extracting and using the rich sentence pair features. The drawback of utilizing all of these features makes the learning process much harder. In this study, we propose a reinforcement learning (RL) method to learn a sentence pair representation when performing tasks like semantic similarity, paraphrase identification, and question-answer pair modelling. We formulate this learning problem as a sequential decision making task where the decision made in the current state will have a strong impact on the following decisions. We address this decision making with a policy gradient RL method which chooses the irrelevant words to delete by looking at the sub-optimal representation of the sentences being compared. With this policy, extensive experiments show that our model achieves on par performance when learning task-specific representations of sentence pairs without needing any further knowledge like parse trees. We suggest that the simplicity of each task inference provided by our RL model makes it easier to explain.

## Introduction

In natural language processing (NLP), most of the tasks require a transformation from an input space to some high dimensional vector space where each dimension captures some aspect of the underlying structure about the data. Learning this kind of representation is a fundamental challenge in NLP and is extensively explored by many recent works (Bengio, Courville, and Vincent 2013), (Le and Mikolov 2014). In order to make correct decisions, the well-studied mainstream tasks, such as comparing a pair of sentences in terms of semantics or paraphrasing, depend heavily on the quality of the learned representation of each sentence since the comparison is made in this representation space (Conneau et al. 2017), (Cer et al. 2018), (Mueller and Thyagarajan 2016) and (Tai, Socher, and Manning 2015).

Mainstream sentence pair comparing models look only at the sentences being compared individually while overlooking the information they share between them (Conneau et al. 2017), (Cer et al. 2018). This is contrary to what humans do, we usually look at the key words of both sentences while comparing them and ignore most of the irrelevant information. For example, while comparing the sentence "*A boy is lying in the snow and is making snow angels*" with the sentence "*Two people wearing snowsuits are on the ground making snow angels*" for the natural language inference task, we can just consider that whether the actors in both of these sentences are "*making snow angels*" or not and take a decision based on that.

Almost all the sentence pair modelling architectures follow a uniform framework: represent the sentences to compare in some high dimensional space using an encoder and compare these representations using a matching module. The encoding section of these models can be classified into four types: *Bag of words* based models which ignore the ordering of the words, *Recurrent and Convolutional neural network* based models which take into consideration the contexts surrounding each word, *Transformer* based models where the complex attention module does the summarizing and finally, the models that work on *predefined structures* like parse trees.

(Iyyer et al. 2015) and (Joulin et al. 2016) propose *bag of words* type models where they ignore the word ordering and instead rely on taking the average of word vectors followed by some linear projection layers. (Liu et al. 2015) also ignore the structure as well as context but relies on an auto-encoder to generate the representation by adapting domain and sentiment supervision.

(Mueller and Thyagarajan 2016) use just one copy of an LSTM to encode the entailment task sentences followed by a Manhattan distance based similarity function for the inference. (Yang et al. 2016), (Conneau et al. 2017), and (Liu et al. 2016) follow the same framework utilizing LSTM followed by a pooling block to summarize the representation even more. (Tai, Socher, and Manning 2015) and (Socher et al. 2013) utilize predefined dependency as well as constituency tree structures and use tree based recurrent networks as the composition function to extract the representation. Furthermore, to get more powerful representations, (Yao Zhou and Pan 2016) encode attention inside the dependency tree variant of tree LSTM whereas (Chen et al. 2017)

extract local and global inference composition by jointly utilizing both standard LSTM as well as tree LSTM.

Recently, Transformer (Vaswani et al. 2017) is getting the spotlight for doing machine translation where the recurrence is mimicked by positional encoding and a series of multi-head attention blocks (Parikh et al. 2016). (Cer et al. 2018) utilize the encoder portion of the Transformer to project the input into a representation space and then use a matching block as the inference layer. (Devlin et al. 2019) create a generalized language representation model using a multi-layer bidirectional Transformer encoder which also provides very good sentence representation.

Unlike most of the existing works which looks at the predefined structures, there are some works which uses automatically optimized structures for learning a representation. (Yogatama et al. 2017) propose a very complex and overly deep model to compose binary tree structures from the supervised downstream task which sometimes gives tree structures that are very different from the standard English syntactic trees. (Chung, Ahn, and Bengio 2017) propose a hierarchical multi-scale recurrent neural network which can capture the latent hierarchical structure in the sequence by encoding the temporal dependencies with different timescales. (Tran and Bisk 2018) propose a model to compose the English syntactic tree structures without even looking at the gold standard label utilizing Kirchhof's matrix tree theorem (Buekenhout and Parker 1998).

In this work, we propose a sentence representation building model using reinforcement learning (RL) for doing the sentence pair modelling task. We devise a training strategy incorporating a policy based actor which takes decision based on the previous context, current input and structure of the counterpart sentence. We use a delayed reward to guide the structure discovery and the reward is computed based on the performance of a sub-optimal critic. (Thrun 1992). We use Monte Carlo sampling for exploring the decision space and the final representation is available when all the sequential decisions are made (Hastings 1970). We fuse the representation module with a policy and critic network where the policy network performs structure discovery based on the response of a sub-optimal critic and the critic gets further tuned on this structured pair representation to adapt itself more on the response of the actor. Even without any explicit structure annotation, our policy based actor builds pretty good sentence representations by filtering out some irrelevant words allowing the critic to get on par or even better performance on these possibly optimal structures.

## Model

In this section, we describe our model in detail. We first explain how the critic is trained in a delayed manner with and without the actor response. Following this, we explain how we train the actor using the response of a trained critic. We conclude this section by giving a high level view of the workflow of our model.

**Training the Critic:** As the critic, we have chosen to use InferSent (Conneau et al. 2017), an LSTM based model, to compute the representations of a pair of sentences and then compare them for an underlying task. It first traverses each sentence as a sequence of $T$ words $\{x_t\}_{t=1,\ldots,T}$ from left to right and generates a hidden representation at each time step $\overrightarrow{h_t},\ \forall t \in [1, \ldots, T]$ .

$$\overrightarrow{h_t} = \overrightarrow{\text{LSTM}_t}(x_1, \ldots, x_T) \tag{1}$$

Following this, it employs a max (or mean) pooling block to summarize the hidden states in one dense representation.

$$\overrightarrow{h} = \text{maxpool}(\overrightarrow{h_1}, \ldots, \overrightarrow{h_T}) \tag{2}$$

The next steps are to infer the similarity between the two representations ($\overrightarrow{h_a}$, $\overrightarrow{h_b}$) using standard matching methods and to project the resultant vector into the space of classes, $y$, through a series of fully connected layers as follows

$$x = (\overrightarrow{h_a}, \overrightarrow{h_b}, |\overrightarrow{h_a} - \overrightarrow{h_b}|, \overrightarrow{h_a} * \overrightarrow{h_b}) \tag{3}$$

$$P(y|\mathbf{X}) = \sigma(\mathbf{W}_1\sigma(\mathbf{W}_2 x + \mathbf{b}_2) + \mathbf{b}_1) \tag{4}$$

Finally, it is trained by optimizing a task specific loss function as follows

$$H(p, q) = -\sum_{i=1}^{n} Q(y_i) \log(P(y_i)) \tag{5}$$

However, in order to have a sub-optimal critic, we do not train it in the standard fashion. Instead, we initialize two copies: *final* and *active*. We perform all of the steps above using the *final* version of the critic, compute the gradients with respect to its parameters ($\theta_f = \{\theta_{f_1}, \ldots, \theta_{f_k}\}$) and store them.

$$\frac{\partial H}{\partial \theta_f} = [\frac{\partial H}{\partial \theta_{f_1}}, \ldots, \frac{\partial H}{\partial \theta_{f_k}}] \tag{6}$$

Generally, in batch-wise training, an average loss is calculated for all of the samples in the batch and the network is updated based on that loss. To mimic this behavior, first the *final* version of critic computes loss for each sample in the batch and stores gradients with respect to its parameters for that loss. Next, all of these gradients are accumulated [1] and we update the *active* version ($\theta_a = \{\theta_{a_1}, \ldots, \theta_{a_k}\}$) of critic as follows.

$$\theta_{a_j} = \theta_{a_j} + \sum_{i \in batch} \frac{\partial H_i}{\partial \theta_{f_j}} \tag{7}$$

This kind of updating allows us to get a sub-optimal critic and we can utilize it later when we train an optimal critic in a weighted fashion based on the actor response. This type of training paradigm is different from training with batch size 1, since here, for each sample in a batch, the loss gets calculated with respect to a fixed set of parameters and the corresponding gradients are applied once the traversal over the entire batch is done, whereas in training with batch size 1, for each batch, a loss is calculated with respect to a new set of parameters because of the continual updating. Finally, after going over an entire batch, the parameters of the *final* version of the critic gets updated by its *active* version parameters as $\theta_f = \theta_a$.

[1] We also tried averaging the gradients but the addition works better.

Apart from this, we adopt another way for training the critic as mentioned before which we use during the fine tuning phase based on the actor response. Instead of doing a straight assignment as above ($\theta_f = \theta_a$), the *final* version gets updated in a weighted fashion as follows

$$\begin{aligned} \theta_f &= \theta_a \times \alpha + \theta_f \times (1-\alpha), \\ \text{where } \theta_a &\in \{\theta_{a_1}, \ldots, \theta_{a_k}\}, \theta_f \in \{\theta_{f_1}, \ldots, \theta_{f_k}\} \end{aligned} \tag{8}$$

Here, the hyperparameter $\alpha$ is set to $0.1$ for all experiments.

**Training the Actor:** We adopt the policy gradient method (Sutton et al. 2000) to update the actor. The policy network guides the policy learning using a stochastic policy $\pi_\theta(a_t|\mathbf{s_t};\boldsymbol{\theta})$ along with a delayed reward. At each time step $t$, an action $a_t$ is sampled from the policy following a probability distribution as follows

$$\pi_\theta(a_t|\mathbf{s_t};\boldsymbol{\theta}) = \sigma(\mathbf{s_t}\mathbf{W} + \mathbf{b}) \tag{9}$$

where $\pi_\theta(a_t|\mathbf{s_t};\boldsymbol{\theta})$ denotes the probability of choosing $a_t$ and $\{\mathbf{W}, \mathbf{b}\}$ is the set of parameters of the actor policy network. Since, we want our policy network to consider the representation of the counterpart sentence, we include it along with the current input and previous context as state. Formally, the state is defined as

$$\mathbf{s_t} = [\mathbf{c_{t-1}}; \mathbf{h_{t-1}}; \mathbf{x_t}; \tilde{\mathbf{h}}_T] \tag{10}$$

Here, $\mathbf{c}_{t-1}$, $\mathbf{h}_{t-1}$ denotes the cell state and hidden sate of critic LSTM at time step $t-1$, $\mathbf{x}_t$ denotes the input at time step $t$ and $\tilde{\mathbf{h}}_T$ is the summary vector of the counterpart sentence generated by the critic LSTM. We use the same policy network to sample actions for both the sentences to be compared. Using this state and policy, we perform action sampling for the whole sequence to obtain the delayed reward (Zhang, Huang, and Zhao 2018) as follows

$$\textit{action} = \begin{cases} 1\text{,if } P(\mathcal{Y}) > \epsilon \text{ and } P(\mathcal{Z}) > \pi_\theta \\ 0\text{,if } P(\mathcal{Y}) > \epsilon \text{ and } P(\mathcal{Z}) < \pi_\theta \\ 1\text{,if } P(\mathcal{Y}) < \epsilon \text{ and } P(\mathcal{Z}) < \pi_\theta \\ 0\text{,if } P(\mathcal{Y}) < \epsilon \text{ and } P(\mathcal{Z}) > \pi_\theta \end{cases} \tag{11}$$

where $\mathcal{Y}$ and $\mathcal{Z}$ are uniform random variables, the hyperparameter $\epsilon$ is set to $0.05$ and $\pi_\theta$ is the policy from Eqn. 9. Our action space is limited to produce binary actions which in a sense can be used as a representation selection model. Once all of the actions are sampled for both the sentences, we get a new representation for both of them which is further used by the critic to compute $P(y|\mathbf{X})$ and estimate the reward to guide the policy learning. Formally, for a given sentence $X = x_1, x_2, \ldots, x_L$, there is a corresponding sampled action sequence $A = a_1, a_2, \ldots, a_L$ obtained from the policy where $a_i = 1$ means to keep the word and $a_i = 0$ means to discard it as it has no contribution in the final representation when compared with its counterpart sentence. Using this hypothesis, the critic states are updated as follows

$$\mathbf{c}_t, \mathbf{h}_t = \begin{cases} \mathbf{c}_{t-1}, \mathbf{h}_{t-1}, & \text{action = 0} \\ f(\mathbf{c}_{t-1}, \mathbf{h}_{t-1}, \mathbf{x}_t) & \text{action = 1} \end{cases} \tag{12}$$

where $f$ denotes all the gate and update functions from the critic, and $\mathbf{c}_t$, $\mathbf{h}_t$ and $\mathbf{x}_t$ denote the cell state, hidden state and input at time step $t$, respectively. In summary, if a word gets deleted at time step $t$, the cell state and hidden state are just copied from time step $t-1$, otherwise it is generated using the standard LSTM gates of critic.

The parameters of our policy net are optimized using the REINFORCE algorithm (Williams 1992) with an objective being maximizing the expected delayed reward as follows

$$\begin{aligned} J(\theta) &= \mathbb{E}_{(\mathbf{s_t}, a_t) \sim P_\theta(\mathbf{s_t}, a_t)} r(\mathbf{s_1} a_1 \ldots \mathbf{s_T} a_T) \\ &= \sum_{\mathbf{s_1} a_1 \ldots \mathbf{s_T} a_T} P_\theta(\mathbf{s_1} a_1 \ldots \mathbf{s_T} a_T) R_T \\ &= \sum_{\mathbf{s_1} a_1 \ldots \mathbf{s_T} a_T} P_\theta(\mathbf{s_1}) \prod_t \pi_\theta(a_t|\mathbf{s_t}) P_\theta(\mathbf{s_{t+1}}|\mathbf{s_t}, a_t) R_T \end{aligned} \tag{13}$$

The delayed reward $R_T$ is computed using the logarithm of the output probability distribution of the critic $\log P(y|\mathbf{X})$ over just one sample. Since our state at time step $t+1$ is fully determined by the state and action at time step $t$, we can make $P_\theta(\mathbf{s_1}) = P_\theta(\mathbf{s_{t+1}}|\mathbf{s_t}, a_t) = 1$ in Eqn. 13. Also as we perform the action sampling for both sentences to be compared, we change the objective function as follows

$$\begin{aligned} J(\theta) = &\sum_{\mathbf{s_1} a_1 \ldots \mathbf{s_T} a_T} \prod_{t \in l} \pi_\theta(a_t|\mathbf{s_t}) R_T + \\ &\sum_{\mathbf{s_1} a_1 \ldots \mathbf{s_T} a_T} \prod_{t \in r} \pi_\theta(a_t|\mathbf{s_t}) R_T \end{aligned} \tag{14}$$

where $l$ denotes the left sentence and $r$ denotes the right sentence. By applying the likelihood ratio trick, we update the policy network with the following gradient

$$\begin{aligned} \frac{\partial J(\theta)}{\partial \theta} = &\sum_{t \in l} R_T \frac{\partial}{\partial \theta} \log \pi_\theta(a_t|\mathbf{s_t}, \boldsymbol{\theta_t}) + \\ &\sum_{t \in r} R_T \frac{\partial}{\partial \theta} \log \pi_\theta(a_t|\mathbf{s_t}, \boldsymbol{\theta_t}) \end{aligned} \tag{15}$$

The policy network controls the number of deleted words by using a unimodal function $f(x) = x + \beta/x$ with the reward and maximize them jointly. The reward $R_T$ is defined as

$$\begin{aligned} R_T = \log P(y|\mathbf{X}) - \frac{\gamma Z}{2} \Big( &\text{abs}(\frac{L_l{}'}{L_l} + \frac{\beta L_l}{L_l{}'}) \\ &+ \text{abs}(\frac{L_r{}'}{L_r} + \frac{\beta L_r}{L_r{}'}) \Big) \end{aligned} \tag{16}$$

where $L_l'$ and $L_r'$ are the number of words with corresponding action value 1, and $L_l$ and $L_r$ are the actual lengths for the left and right sentences respectively. $\gamma$ is a hyperparameter, $Z$ is the number of classes and $\beta$ is the proportion of words we want deleted.

Like the critic, we initialize two versions of the actor: *final* and *active*. For each sample in a batch, the *active* version of the actor policy network is updated by the gradient of the *final* version of the actor policy network parameters

$$\theta_{a_j} = \theta_{a_j} + \sum_{i \in batch} \frac{\partial J(\theta)_i}{\partial \theta_{f_j}} \tag{17}$$

Once a batch is finished, the *final* version of the actor policy is updated in a similar way as the critic using Eqn. 8.

**Algorithm 1** Overall Training

Initialize two copies of Critic network $C_f$ and $C_a$
Denote the LSTMs of Critic network as $L_f$ and $L_a$
Set of Critic Parameters $\{\theta_{f_1} \dots \theta_{f_k}\} \cup \{\theta_{a_1} \dots \theta_{a_k}\}$
Discount factor $\alpha \in [0, 1]$ and Learning rate $\lambda \in [0, 1]$
**for** batch $\in$ Batches **do**
  **for** sample $\in$ batch **do**
    $x, y \leftarrow sample$
    $\nabla\theta_f \leftarrow \nabla P_{\theta_f}(out^*|x, y, \theta_f),\ \forall\theta_f \in \{\theta_{f_1} \dots \theta_{f_k}\}$
    $\theta_{a_j} = \theta_{a_j} + \lambda \times \nabla\theta_{f_j},\ j \in \{1 \dots k\}$
  **end for**
  $\theta_{f_j} = \theta_{a_j},\ j \in \{1 \dots k\}$
**end for**
Train Actor on Critic ($C_f$) response using Algorithm 2
Initialize $\{\theta_{a_1} \dots \theta_{a_k}\}$ by $\{\theta_{f_1} \dots \theta_{f_k}\}$
**for** batch $\in$ Batches **do**
  **for** sample $\in$ batch **do**
    $x, y \leftarrow sample$, $\tilde{x} \leftarrow \emptyset$, $\tilde{y} \leftarrow \emptyset$
    $\tilde{h}_{T_x} \leftarrow L_f(y)$, $\tilde{h}_{T_y} \leftarrow L_f(x)$
    Calculate $s_{t_x}$ using word vector $x_t$, sentence vector $\tilde{h}_{T_x}$, cell and hidden states of $L_f$ at previous time step with Eqn. 10
    Calculate $s_{t_y}$ using word vector $y_t$, sentence vector $\tilde{h}_{T_y}$, cell and hidden states of $L_f$ at previous time step with Eqn. 10
    Calculate $\pi_{\theta_x}$ using $s_{t_x}$ and $\pi_{\theta_y}$ using $s_{t_y}$ with Eqn. 9
    Sample actions $a_x$ using policy $\pi_{\theta_x}$ and $a_y$ using policy $\pi_{\theta_y}$ with Eqn. 11
    **for** action in $a_x$ **do**
      **if** action = 1 at index $t$ **then**
        keep the word at index $t$ and add it to $\tilde{x}$
      **end if**
    **end for**
    **for** action in $a_y$ **do**
      **if** action = 1 at index $t$ **then**
        keep the word at index $t$ and add it to $\tilde{y}$
      **end if**
    **end for**
    $\nabla\theta_f \leftarrow \nabla P_{\theta_f}(out^*|\tilde{x}, \tilde{y}, \theta_f),\ \forall\theta_f \in \{\theta_{f_1} \dots \theta_{f_k}\}$

    $\theta_{a_j} = \theta_{a_j} + \lambda \times \nabla\theta_{f_j},\ j \in \{1 \dots k\}$
  **end for**
  $\theta_{f_j} = \theta_{f_j} \times (1 - \alpha) + \theta_{a_j} \times \alpha,\ j \in \{1 \dots k\}$
  Initialize $\{\theta_{a_1} \dots \theta_{a_k}\}$ by $\{\theta_{f_1} \dots \theta_{f_k}\}$
**end for**

**Algorithm 2** Actor Training

Start with a pre-trained Critic $C$ having an LSTM cell $L$
Number of random sampling $N \in [1, n]$
Initialize two copies of Actor policy networks $\pi_{\theta_f}, \pi_{\theta_a}$
Set of Actor Parameters $\{\theta_{f_1} \dots \theta_{f_k}\} \cup \{\theta_{a_1} \dots \theta_{a_k}\}$
Discount factor $\alpha \in [0, 1]$, Learning rate $\lambda \in [0, 1]$
Function to calculate the length $F(.)$
No. of classes $Z \in [1, n]$
Proportion of the words to be deleted $\beta \in [1, n]$

**Workflow**: We now give a high level view of the work flow of our entire model. An algorithmic presentation detailing these steps is provided on this page.

**Algorithm 2** Actor Training (continued)

**for** batch $\in$ Batches **do**
  **for** sample $\in$ batch **do**
    $x, y \leftarrow sample$
    Left Summary $x' \leftarrow L(x)$, Right Summary $y' \leftarrow L(y)$
    States $s_X \leftarrow \emptyset$ , $s_Y \leftarrow \emptyset$
    Actions $a_X \leftarrow \emptyset$ , $a_Y \leftarrow \emptyset$
    Loss $l \leftarrow \emptyset$, $j \leftarrow 0$
    **while** $j < N$ **do**
      $\tilde{x} \leftarrow \emptyset$, $\tilde{y} \leftarrow \emptyset$
      Calculate $s_{t_x}$ using word vector $x_t$, sentence vector $\tilde{h}_{T_x}$, cell and hidden states of $L$ at previous time step with Eqn. 10
      Calculate $s_{t_y}$ using word vector $y_t$, sentence vector $\tilde{h}_{T_y}$, cell and hidden states of $L$ at previous time step with Eqn. 10
      Calculate $\pi_{\theta_x}$ using $s_{t_x}$, $\pi_{\theta_y}$ using $s_{t_y}$ with Eqn. 9
      Sample actions $a_x$ using policy $\pi_{\theta_x}$ and $a_y$ using policy $\pi_{\theta_y}$ with Eqn. 11
      $a_{X_j} \leftarrow a_{X_j} + a_{t_x}$, $a_{Y_j} \leftarrow a_{Y_j} + a_{t_y}$
      $s_{X_j} \leftarrow s_{X_j} + s_{t_x}$, $s_{Y_j} \leftarrow s_{Y_j} + s_{t_y}$
      **for** action in $a_x$ **do**
        **if** action = 1 at index $t$ **then**
          keep the word at index $t$ and add it to $\tilde{x}$
        **end if**
      **end for**
      **for** action in $a_y$ **do**
        **if** action = 1 at index $t$ **then**
          keep the word at index $t$ and add it to $\tilde{y}$
        **end if**
      **end for**
      $l \leftarrow l + C(\tilde{x}, \tilde{y}) + \frac{\gamma Z}{2} \left( \left| \frac{F(\tilde{x})}{F(x)} + \frac{\beta * F(x)}{F(\tilde{x})} \right| + \left| \frac{F(\tilde{y})}{F(y)} + \frac{\beta * F(y)}{F(\tilde{y})} \right| \right)$

      $j \leftarrow j + 1$
    **end while**
    $l_a \leftarrow (\sum_{i=1}^{N} l_i)/N$
    Initialize the gradients $\nabla\theta_f \in \{\nabla\theta_{f_1} \dots \nabla\theta_{f_k}\}$ by 0
    $j \leftarrow 0$
    **while** $j < N$ **do**
      $i \leftarrow 0$
      **while** $i < F(a_X)$ **do**
        $R_L \leftarrow (l_{j,i} - l_a) \times \alpha$
        $\nabla\theta_f \leftarrow \nabla\theta_f + R_L \nabla \log \pi_{\theta_f}(a^*_{X_{j,i}}|s_{X_{j,i}}, \theta_f)$
        $\forall\theta_f \in \{\theta_{f_1} \dots \theta_{f_k}\}$
        $i \leftarrow i + 1$
      **end while**
      $i \leftarrow 0$
      **while** $i < F(a_Y)$ **do**
        $R_L \leftarrow (l_{j,i} - l_a) \times \alpha$
        $\nabla\theta_f \leftarrow \nabla\theta_f + R_L \nabla \log \pi_{\theta_f}(a^*_{Y_{j,i}}|s_{Y_{j,i}}, \theta_f)$
        $\forall\theta_f \in \{\theta_{f_1} \dots \theta_{f_k}\}$
        $i \leftarrow i + 1$
      **end while**
      $j \leftarrow j + 1$
    **end while**
    $\theta_{a_j} = \theta_{a_j} + \lambda \times \nabla\theta_{f_j},\ j \in \{1 \dots k\}$
  **end for**
  $\theta_{f_j} = \theta_{f_j} \times (1 - \alpha) + \theta_{a_j} \times \alpha,\ j \in \{1 \dots k\}$
  Initialize $\{\theta_{a_1} \dots \theta_{a_k}\}$ by $\{\theta_{f_1} \dots \theta_{f_k}\}$
**end for**

We start by training a sub-optimal critic in a delayed manner by first initializing two versions of it (*final* and *active*). For each sample in a batch, we update the *active* version using the gradients of the *final* version. While doing this, we keep the parameters of the *final* version fixed. Once the entire batch is looked at, the *active* version is used to update the *final* version through a straightforward assignment.

After training this sub-optimal critic, we use its response to train an actor with a policy gradient method. To accomplish this, we again start by having two versions of actor (*final* and *active*). With each sample in the batch, Eqn. 10 gets an $s_t$ for each word in the two sentences in the sample. Next, we use these two sets of $s_t$'s in Eqn. 9 to calculate the policies ($\pi_\theta$'s) for each word in the two sentences. We then use each policy with the sampling strategy as defined in Eqn. 11 to get the corresponding action ($a_t$) associated with each word in the two sentences. Each action represents whether to delete or keep a word. Next, we modify both of the sentences according to these sampled actions. These modified sentences are used with the trained critic to calculate a reward. This reward is modified with a term that reflects the ratio of the number of words that we delete from both of the sentences (the subtrahend in Eqn. 16). We store all of the $s_t$'s, $a_t$'s as well as the associated reward and repeat this strategy $N$ times. We then calculate an average reward. The objective function in Eqn. 14 and the gradients in Eqn. 15 are calculated $N$ times using the $N$ sets of the logarithm of policies and the $N$ differences between the $N$ rewards and the average reward. We again adopt the delaying strategy in updating the actor parameters.

For each sample in the batch, the just calculated gradients of the parameters of the *final* version of the actor are used to update the *active* version. We keep the *final* version parameters constant until all the samples in the batch are looked at. Eqn. 8 is used to update the parameters of the *final* version of actor by its *active* version in a weighted fashion. The *active* version is then set to this new *final* version. We continue this updating of actor until all the batches are looked at.

After the actor is trained, the previously trained sub-optimal critic is further tuned using the response of this trained actor. Rather than looking at the real sentence pairs, now the critic looks at the actor-modified sentences. A loss is calculated based on the critic response and the *active* version is updated through the gradients of the parameters of the *final* version. Eqn. 8 updates the *final* version of the critic once an entire batch is looked at.

**Ensemble Method**: In order to verify the contribution and structure selection ability of actor, we also perform an ensemble decision check by combining the responses of two critics (i.e., InferSent), one trained just on the raw words without any actor and one trained on the response of an actor. We hypothesize that the raw InferSent model should get more importance than the one which needs an actor. Both model weights are initialized to $0.5$ weight. The weight of the one without any actor goes from $0.5$ to $1.0$ and the one with an actor goes from $0.5$ to $0$. The final decision $F_d$ is taken by doing a weighted average on the response of both of the participating entities: $C_d$ (critic decision) and $AC_d$ (actor-critic decision).

$$F_d = C_d \times w + AC_d \times (1 - w) \tag{18}$$

Here, $w$ and $(1 - w)$ are the weights on the critic and actor-critic decisions and they are selected through a grid search over the validation set.

## Datasets, Experimental Setup, Results and Analysis

In this section, we explain the experimental setup along with the results obtained and a thorough analysis. We first describe our training corpora as well as all of the benchmarks used in other standard sentence pair modelling studies. Following this, we explain the technical details of our proposed architecture along with its hyper-parameter settings. We also present the detailed results obtained with our RL model and compare with some of the top performing models on their selected datasets. Additionally, we give a qualitative analysis by showing the predictions of our models on some random test samples for all of the tasks. Finally, we conclude this section by giving some insight into the performance of our model by analyzing the generated structures.

**Datasets**: Model evaluation uses three datasets: paraphrase identification, natural language inference, and question-answer pair modelling.

- **MSRP**: Given a pair of sentences, the task is to identify whether or not they are paraphrases of each other (Dolan, Quirk, and Brockett 2004).
- **SICK**: The dataset contains sentences derived from video and image annotations and the task is to classify a given sentence pair into three classes: Entailment, Neutral and Contradiction (Marelli and others 2014).
- **AI2-8grade**: The task is to do a true-false question selection where each data sample consists of a pair of sentences with one being the question and the other being the evidence formed by replacing the *wh* in the question by the answer (Baudiš, Stanko, and Šedivý 2016).

**Experimental Setup**: The LSTM hidden state dimension is set to $1024$. Word vectors are initialized with the $300$ dimension GloVe embeddings (Pennington, Socher, and Manning 2014) and are not updated during training. To smooth the update during critic training, the gradients are divided by $B^2$ where $B$ is the batch size which is $5$. The learning rate is reduced by a factor of 2 if $\sqrt{\sum_{i=1}^{k} \|\nabla\theta_i^2\|}$ is more than a threshold, which is $5$ for our experiments. To smooth the policy gradient update, we add a $\gamma$, $\beta$, and $Z$ scaled regularized reward as shown in Eqn. 16. The values of $\gamma$ and $\beta$ are $0.1$ and $0.15$, respectively, and $Z$ is the number of classes.

During training, the critic parameters are updated using stochastic gradient descent (Bottou 2010) with an initial learning rate of $0.1$, whereas for training actor, we use Adam (Kingma and Ba 2014) to update the parameters with a fixed learning rate of $0.01$.

**Results and Analysis**: Table 1 compares the performance of our two models on the three tasks to some top performing generalized sentence encoders and some designed for

| Model | MSRP Acc. | AI2-8grade Acc. | SICK Acc. |
|---|---|---|---|
| InferSent (Conneau et al. 2017) † | 74.46 | 74.71 | 84.07 |
| LSTM (Conneau et al. 2017) † | 70.74 | 74.93 | 76.80 |
| BiGRU Last Encoder (Conneau et al. 2017) † | 70.46 | 74.61 | 81.47 |
| Inner Attention (Lin et al. 2017) † | 69.74 | 74.73 | 72.01 |
| ConvNet Encoder (Zhao, Lu, and Poupart 2015) † | 73.96 | **75.26** | 83.82 |
| InferSent + RL | 74.74 | 73.84 | 84.57 |
| InferSent + RL (Ensemble) | **76.12** | 74.91 | **86.12** |
| Seq-LSTMs (Yao Zhou and Pan 2016) | 71.70 | 63.30 | - |
| Seq-GRUs (Yao Zhou and Pan 2016) | 71.80 | 62.40 | - |
| Tree LSTM (Yao Zhou and Pan 2016) | 73.50 | 69.10 | - |
| Tree LSTM + Attn. (Yao Zhou and Pan 2016) | 75.80 | 72.50 | - |
| Tree GRU (Yao Zhou and Pan 2016) | 73.96 | 70.60 | - |
| Tree GRU + Attn. (Yao Zhou and Pan 2016) | 74.80 | 72.10 | - |
| RNN (Baudiš, Stanko, and Šedivý 2016) | - | 36.10 | - |
| CNN (Baudiš, Stanko, and Šedivý 2016) | - | 38.40 | - |
| RNN-CNN (Baudiš, Stanko, and Šedivý 2016) | - | 37.60 | - |
| Attn1511 (Baudiš, Stanko, and Šedivý 2016) | - | 35.80 | - |
| Ubu.RNN (Baudiš, Stanko, and Šedivý 2016) | - | 44.10 | - |
| Illinois-LH (Lai and others 2014) | - | - | 84.60 |
| UNAL NLP (Jimenez and others 2014) | - | - | 83.10 |
| SNLI-Transfer 3-class LSTM (Bowman et al. 2015) | - | - | 80.80 |
| MaLSTM features + LSTM (Mueller and Thyagarajan 2016) | - | - | 84.20 |
| ECNU (Zhao and others 2014) | - | - | 83.60 |

Table 1: Performance comparison of our model on different tasks against some existing top performing models. We mark models that we implemented as †.

a specific task using accuracy as the evaluation metric. To do a fair comparison, we used the official implementation of InferSent, LSTM, BiGRU Last encoder, Inner attention, ConvNet encoder with the same settings as ours[2]. On the MSRP task, our ensemble model achieves $76.12\%$ accuracy which is better than all of the sequential and tree based sentence encoders. It is noteworthy that these tree based models have access to parse trees which are expensive to compute. We get better performance without this information. On the AI2-8grade dataset, performance ($73.84\%$ for InferSent + RL and $74.91\%$ for InferSent + RL (Ensemble)) is below the existing models; however, it is still on par. Later, we show that our model (InferSent + RL) is removing around $90\%$ of the content from this dataset yet is still able to achieve this comparable performance. Finally, state of the art performance ($86.12\%$ accuracy) is achieved on the natural language inference task on the SICK dataset. Even though our critic model is much simpler having just a unidirectional LSTM, it is doing much better than (Bowman et al. 2015) with $80.80\%$ accuracy who use transfer learning and (Lin et al. 2017) with $72.01\%$ accuracy who use an attention block on top of a bidirectional LSTM. Lastly, our version of InferSent with an actor is better than the standard InferSent suggesting that our actor has been able to correctly remove the irrelevant words.

Table 2 gives the ensemble method's final $w$ values and final results. Adding RL alone improves InferSent slightly on two of the three datasets. Carefully re-introducing the effect of an actor-free InferSent with an ensemble technique further improves the InferSent + RL to being state of the art on the MSRP and SICK datasets. On the AI2-8grade dataset, only one model is doing better than our ensemble model.

| Dataset | w | $1-w$ | Acc. |
|---|---|---|---|
| MSRP | 0.53 | 0.47 | 76.12 |
| AI2-8grade | 0.76 | 0.24 | 74.91 |
| SICK | 0.65 | 0.35 | 86.12 |

Table 2: The $w$ values that give the best results when combining the critic ($w$) with the trained actor-critic ($1-w$).

Table 3 shows the performance of our InferSent + RL model on some examples from the test sets of the three corpora. For the SICK dataset, our RL model removes prepositions, articles, and adjectives like colours, which seem not to have any impact on the semantics. It also removes the common phrases like "a piece of", "There is" and "of the". On the MSRP dataset, our model again removes articles and prepositions which seem not to be of concern when checking for paraphrasing. We notice that our model tries to keep the same subset of words from both sentences in most of the cases and removes the words interspersed among those shared words. For example, in the first example in this group, our model deletes the phrase "to the section that" from the left sentence and some smaller phrases like "in a" and "to the" from both of the sentences. This reduction makes the

[2]Available at https://github.com/facebookresearch/InferSent.

| Dataset | Sentence 1 | Sentence 2 | GT | Pr |
|---|---|---|---|---|
| SICK | ~~A brown~~ dog ~~is~~ attacking ~~another~~ animal in front ~~of the~~ man in pants | ~~There is~~ no dog wrestling and hugging | N | N |
| | ~~A~~ cat ~~is~~ crawling under ~~a piece of~~ furniture | ~~An~~ animal ~~is~~ crawling under ~~a piece of~~ furniture | E | E |
| | ~~A~~ blonde boy ~~in green is~~ sitting ~~on a~~ swing | ~~A~~ blonde boy ~~in green is~~ standing ~~on a~~ swing | C | E |
| MSRP | They did not ~~read~~ footnotes ~~in a~~ document said ~~the~~ official referring ~~to the section that~~ contained ~~the~~ State Departments dissent | They did not ~~read~~ footnotes ~~in a~~ document ~~he~~ said referring ~~to the~~ annex | 1 | 1 |
| | The company ~~didn't~~ detail ~~the~~ costs ~~of the~~ replacement ~~and~~ repairs | But company officials expect ~~the~~ costs ~~of the~~ replacement ~~work to run into the~~ millions ~~of~~ dollars | 0 | 0 |
| | We are piloting ~~it there to see~~ whether we ~~roll it~~ ~~out to other~~ products | Macromedia ~~is~~ piloting ~~this~~ product activation system ~~in~~ Contribute ~~to~~ test whether ~~to roll it out to~~ ~~other~~ products | 1 | 0 |
| AI 2-8grade | cell wall structure is found in a plant cell but not in an animal cell | ~~The cell~~ wall ~~provides structural~~ support and protection | 1 | 1 |
| | | Pores ~~in the cell~~ wall ~~allow water and nutrients to move into and out of the cell~~ | 1 | 1 |
| | | ~~The cell~~ wall ~~also~~ prevents ~~the plant~~ ~~cell~~ from ~~bursting when water enters the cell~~ | 1 | 1 |
| | positive effect of recycling aluminum cans to manufacture new beverage containers is warming | ~~also~~ maintains ~~the ozone~~ layer ~~that helps~~ protect ~~life~~ from damaging UV | 0 | 0 |
| | The layered mixture of gases surrounding Earth is called the atmosphere | Without ~~it Earth would be a~~ harsh barren world | 1 | 0 |

Table 3: Example predictions from the test set. **GT**: ground truth, **Pr**: predicted.

sentences quite similar and eventually the predicted decision is 1. On the other hand, our model makes a wrong prediction on the third example. It tries to do the same thing by removing phrases, but as it does not have any world knowledge about "Macromedia Contribute" being a product, it fails to map it to the word "product" in the left sentence. Finally, on the AI2-8grade dataset, our model removes most of the words from the answer sentence and ends up keeping just the key words to match with the corresponding question. In the first example, our model keeps the word "wall" in all of the answer sentences making the mapping easier with the question. In the second example, our model keeps the key words like "protect", "damaging" and "UV" in the answer sentence making it clear that the question and the answer topics are completely different and as a result the prediction is 0. Finally, in the third example, our model incorrectly removes the word "Earth" from the answer sentence which makes it difficult for the critic to correctly map it to the question.

Table 4 exhibits how much information is removed and how much is kept for each task by our InferSent + RL model. For the paraphrase identification task, about 33% of the original content is removed giving better results than the sequence based models. For the question-answer selection task on the AI2-8grade dataset, our model removes around 90% of the original content and still achieves on par performance. In this dataset, the question-answer pairs are selected from 2nd to 8th grade books. To increase their readability (Leskovec, Milic-Frayling, and Grobelnik 2005), some easy

| Dataset | Left | | Right | |
|---|---|---|---|---|
| | Before | After | Before | After |
| MSRP | 18.55 | 12.11 | 18.51 | 12.10 |
| AI2-8grade | 21.52 | 2.37 | 15.34 | 2.02 |
| SICK | 9.68 | 5.02 | 9.52 | 4.93 |

Table 4: The original average length and the average length after filtering through RL.

to read words are added around the key words of those sentences. Our model suggests that the easy to read words are not important for the inference. Finally, to do natural language inference (NLI) on the SICK dataset, our model removes about 45% of the content and gets better performance than all of the existing models. These results indicate that the three tasks can be done with more condensed and purified information without losing too much generalizability.

We also analyze the type of words deleted from each corpus with the InferSent + RL model and report the results in Table 5[3]. We use the harmonic mean of the ratio of the number of times a word is deleted to its frequency in the corpus (as a percentage) and the number of occurrences of that word to sort the list. It is clear that in two corpora, non-content words (i.e., prepositions, articles) are deleted most of the

[3]A more complete table is available at https://github.com/navid5792/SS_actor_critic/blob/master/Words\%20Deleted.pdf.

| MSRP | | AI2-8grade | | SICK | |
|---|---|---|---|---|---|
| **Word** | **HM** | **Word** | **HM** | **Word** | **HM** |
| a | 184.39 | a | 197.31 | a | 197.14 |
| and | 181.79 | the | 196.93 | is | 194.76 |
| the | 179.23 | of | 196.13 | the | 194.07 |
| in | 162.63 | to | 195.07 | of | 184.80 |
| for | 160.64 | that | 194.90 | white | 167.68 |
| to | 160.31 | in | 193.43 | black | 167.38 |
| is | 156.13 | are | 193.00 | in | 159.06 |
| with | 151.28 | by | 192.48 | and | 153.53 |
| on | 147.50 | most | 191.75 | blue | 147.53 |
| it | 138.55 | for | 191.58 | red | 146.92 |

Table 5: Top 10 deleted words from the test set of all three corpora. **HM**: harmonic mean


Figure 1: Effect of RL sample counts on Validation accuracy.

time. However, in the SICK dataset, there are words like "black", "white", "blue" and "red" which are deleted most often because for doing the NLI task, these words contribute very little and can be ignored. For the other two corpora, our model also deletes words like "this", "have", "likely", "has", "than" and "they" quite often but the rate of deleting them is not as high as the ones reported in Table 5.

Table 5 might suggest that stop words be removed in a preprocessing step. For tasks like semantic relatedness our research suggests not. Stop words with semantic polarity, like "no" and "not", are very important as they control the overall sentiment of the sentence (Ahmed and Mercer 2019). We have trained InferSent on the SICK dataset with stop words removed. The evaluation indicates a poorer performance (79.95%) than InferSent with stop words (84.07%). Similar results are obtained for the other two datasets.

In RL, the search space is huge making it hard to find the best action with the maximum reward. To explore a larger region of this space, we choose to experiment with changing the number of sample counts and storing the average reward based on each possible action. Figure 1 depicts the effect of different sample counts on validation accuracy. We choose the range of sample counts to be from 1 to 20, train the actor based on the number of samples and record the validation accuracies. We further train as many critics as the number of actors and also record the validation performances. Finally, sample count 5 was selected, as our model was obtaining the best validation performance with this number.

In this study, we adopt a delayed update mechanism in

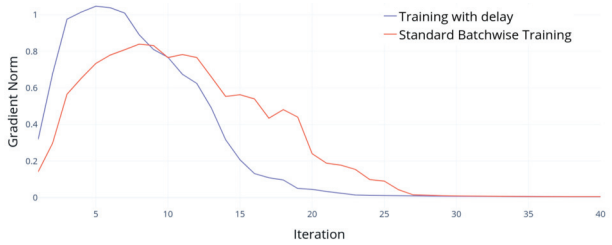


Figure 2: Change in gradient norm with training iterations while training with and without delay.

training both actor and critic. This framework allows us to fine tune critic partially without relying on all that actor suggests. To check the effectiveness of this type of training compared to the standard batch gradient descent, we have plotted the gradient norm changes at each epoch in Figure 2. As evidenced, change is initially quite drastic but eventually becomes much smoother with the number of iterations and convergence is much quicker compared to standard training.

## Conclusion

We have proposed a reinforcement learning method to train a sentence pair modelling architecture using an actor-critic framework. With the help of an actor, our critic model learns to compare two sentences by looking more at content words rather than all words in the sentence, similar to how humans do it. To take an action, our actor model looks not only at the content of just one sentence, it also considers the mutual information shared between the two sentences. Results show that our model gets on par or better performance compared to the existing models by overlooking the irrelevant content.

## References


Ahmed, M., and Mercer, R. E. 2019. Efficient transformer-based sentence encoding for sentence pair modelling. In Meurs, M.-J., and Rudzicz, F., eds., *Advances in Artificial Intelligence*, 146–159. Springer International Publishing.

Baudiš, P.; Stanko, S.; and Šedivý, J. 2016. Joint learning of sentence embeddings for relevance and entailment. In *Proc. of the 1st Wksp. on Representation Learning for NLP*, 8–17.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE Trans. on Pattern Anal. and Mach. Intell.* 35(8):1798–1828.

Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *Proc. of Computational Statistics 2010*, (COMPSTAT'2010), 177–186.

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. In *Proc. of the 2015 Conf. on Empirical Methods in Natural Language Processing*, 632–642.

Buekenhout, F., and Parker, M. 1998. The number of nets of the regular convex polytopes in dimension $\leq 4$. *Discrete Mathematics* 186(1-3):69–94.

Cer, D.; Yang, Y.; Kong, S.-y.; Hua, N.; Limtiaco, N.; John, R. S.; Constant, N.; Guajardo-Cespedes, M.; Yuan, S.; Tar, C.; Sung, Y.-H.; Strope, B.; and Kurzweil, R. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Chen, Q.; Zhu, X.; Ling, Z.; Wei, S.; Jiang, H.; and Inkpen, D. 2017. Enhanced LSTM for natural language inference. In *Proc. of the 55th Annual Meeting of the Association for Computational Linguistics*, 1657 – 1668.

Chung, J.; Ahn, S.; and Bengio, Y. 2017. Hierarchical multiscale recurrent neural networks. In *International Conference on Learning Representations*.

Conneau, A.; Kiela, D.; Schwenk, H.; Barrault, L.; and Bordes, A. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 670–680.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1 (Long and Short Papers)*, 4171–4186.

Dolan, B.; Quirk, C.; and Brockett, C. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proc. of the 20th Int. Conf. on Computational Linguistics*, 350.

Hastings, W. K. 1970. *Monte Carlo Sampling Methods using Markov Chains and Their Applications*. Oxford University Press.

Iyyer, M.; Manjunatha, V.; Boyd-Graber, J.; and Daumé III, H. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Int. Joint Conf. on Natural Language Processing (Volume 1: Long Papers)*, 1681–1691.

Jimenez, S., et al. 2014. UNAL-NLP: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *Proc. of the 8th Int. Workshop on Semantic Evaluation (SemEval 2014)*, 732–742.

Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2016. Bag of tricks for efficient text classification. In *Proc. of the 15th Conf. of the European Chapter of the Association for Computational Linguistics, Vol. 2 Short Papers*, 427–431.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Lai, A., et al. 2014. Illinois-LH: A denotational and distributional approach to semantics. In *Proc. of the 8th Int. Workshop on Semantic Evaluation*, 329–334.

Le, Q., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *Int. Conf. on Machine Learning*, 1188–1196.

Leskovec, J.; Milic-Frayling, N.; and Grobelnik, M. 2005. Extracting summary sentences based on the document semantic graph. manuscript.

Lin, Z.; Feng, M.; Santos, C. N. d.; Yu, M.; Xiang, B.; Zhou, B.; and Bengio, Y. 2017. A structured self-attentive sentence embedding. In *Int. Conf. on Learning Representations*.

Liu, B.; Huang, M.; Sun, J.; and Zhu, X. 2015. Incorporating domain and sentiment supervision in representation learning for domain adaptation. In *Proc. of the Twenty-Fourth Int. Joint Conf. on Artificial Intelligence*, 1277–1283.

Liu, Y.; Sun, C.; Lin, L.; and Wang, X. 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. *arXiv preprint arXiv:1605.09090*.

Marelli, M., et al. 2014. SemEval-2014 Task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proc. of the 8th Int. Wkshp. on Semantic Evaluation*, 1–8.

Mueller, J., and Thyagarajan, A. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proc. of the Thirtieth AAAI Conf. on Artificial Intelligence*, 2786–2792.

Parikh, A.; Täckström, O.; Das, D.; and Uszkoreit, J. 2016. A decomposable attention model for natural language inference. In *Proc. of the 2016 Conf. on Empirical Methods in Natural Language Processing*, 2249–2255.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J. m.-s. p. n. s.; Manning, C. D.; Ng, A.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of the 2013 Conf. on Empirical Methods in Natural Language Processing*, 1631–1642.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 1057–1063.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Int. Joint Conf. on Natural Language Processing (Volume 1: Long Papers)*, 1556–1566.

Thrun, S. B. 1992. Efficient exploration in reinforcement learning. Technical report, Carnegie Mellon University.

Tran, K., and Bisk, Y. 2018. Inducing grammars with and for neural machine translation. In *Proc. of the 2nd Workshop on Neural Machine Translation and Generation*, 25–35.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, 5998–6008.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8(3-4):229–256.

Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; and Hovy, E. 2016. Hierarchical attention networks for document classification. In *Proc. of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1480–1489.

Yao Zhou, C. L., and Pan, Y. 2016. Modelling sentence pairs with tree-structured attentive encoder. In *The Int. Conf. on Computational Linguistics (COLING)*.

Yogatama, D.; Blunsom, P.; Dyer, C.; Grefenstette, E.; and Ling, W. 2017. Learning to compose words into sentences with reinforcement learning. In *International Conference on Learning Representations*.

Zhang, T.; Huang, M.; and Zhao, L. 2018. Learning structured representation for text classification via reinforcement learning. In *Proc. of the Thirty-Second AAAI Conf. on Artificial Intelligence*, 6053–6060.

Zhao, J., et al. 2014. ECNU: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *Proc. of the 8th Int. Workshop on Semantic Evaluation (SemEval 2014)*, 271–277.

Zhao, H.; Lu, Z.; and Poupart, P. 2015. Self-adaptive hierarchical sentence model. In *Proc. of the Twenty-Fourth Int. Joint Conf. on Artificial Intelligence*, 4069–4076.