The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Learning Structured Representation for Text Classificationxvia Reinforcement Learning

Tianyang Zhang,*[*] Minlie Huang,*[*,†] Li Zhao[‡]
[*]Tsinghua National Laboratory for Information Science and Technology
Dept. of Computer Science and Technology, Tsinghua University, Beijing 100084, PR China
[‡] Microsoft Research Asia
keavilzhangzty@gmail.com; aihuang@tsinghua.edu.cn; lizo@microsoft.com;
[†]Corresponding Author: aihuang@tsinghua.edu.cn (Minlie Huang)

Abstract

Representation learning is a fundamental problem in natural language processing. This paper studies how to learn a structured representation for text classification. Unlike most existing representation models that either use no structure or rely on pre-specified structures, we propose a reinforcement learning (RL) method to learn sentence representation by discovering optimized structures automatically. We demonstrate two attempts to build structured representation: Information Distilled LSTM (ID-LSTM) and Hierarchically Structured LSTM (HS-LSTM). ID-LSTM selects only important, task-relevant words, and HS-LSTM discovers phrase structures in a sentence. Structure discovery in the two representation models is formulated as a sequential decision problem: current decision of structure discovery affects following decisions, which can be addressed by policy gradient RL. Results show that our method can learn task-friendly representations by identifying important words or task-relevant structures without explicit structure annotations, and thus yields competitive performance.

Introduction

Representation learning is a fundamental problem in AI, and particularly important for natural language processing (NLP) (Bengio, Courville, and Vincent 2013; Le and Mikolov 2014). As one of the most common tasks of NLP, text classification depends heavily on the learned representation, and is widely applied in sentiment analysis (Socher et al. 2013), question classification (Kim 2014), and language inference (Bowman et al. 2015).

Mainstream representation models for text classification can be roughly classified into four types. *Bag-of-words representation models* ignore the order of words, including deep average network (Iyyer et al. 2015; Joulin et al. 2017) and autoencoders (Liu et al. 2015). *Sequence representation models* such as convolutional neural network (Kim 2014; Kalchbrenner, Grefenstette, and Blunsom 2014; Lei, Barzilay, and Jaakkola 2015) and recurrent neural network (Hochreiter and Schmidhuber 1997; Chung et al. 2014) consider word order but do not use any structure. *Structured representation models* such as tree-structured LSTM (Zhu, Sobihani, and Guo 2015; Tai, Socher, and Manning 2015) and recursive autoencoders (Socher et al. 2013; 2011; Qian et al. 2015) use pre-specified parsing trees to build structured representations. *Attention-based methods* (Yang et al. 2016; Zhou, Wan, and Xiao 2016; Lin et al. 2017) use attention mechanisms to build representations by scoring input words or sentences differentially.

However, in existing structured representation models, the structures are either provided as input or predicted using supervision from explicit treebank annotations. There has been few studies on learning representations with automatically optimized structures. Yogatama et al. (2017) proposed to compose binary tree structure for sentence representation with only supervision from downstream tasks, but such structure is very complex and overly deep, leading to unsatisfactory classification performance. In (Chung, Ahn, and Bengio 2017), a hierarchical representation model was proposed to capture latent structure in the sequences with latent variables. Structure is discovered in a latent, implicit manner.

In this paper, we propose a reinforcement learning (RL) method to build structured sentence representations by identifying task-relevant structures without explicit structure annotations. Structure discovery in this paper is formulated as a *sequential decision* problem: current decision (or action) of structure discovery affects following decisions, which can be naturally addressed by policy gradient method (Sutton et al. 2000). A delayed reward is used to guide the learning of the policy for structure discovery. The reward is computed from the text classifier's prediction based on the structured representation. The representation is available only when all sequential decisions are completed.

In our RL method, we design two structured representation models: *Information Distilled LSTM (ID-LSTM)* which selects important, task-relevant words to build sentence representation, and *Hierarchical Structured LSTM (HS-LSTM)* which discovers phrase structures and builds sentence representation with a two-level LSTM. The representation models are integrated seamlessly with a policy network and a classification network. The policy network defines a policy for structure discovery, and the classification network makes prediction on top of structured sentence representation and facilitates reward computation for the policy network.

To summarize, our contributions are as follows:

- We propose a reinforcement learning method which dis-

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

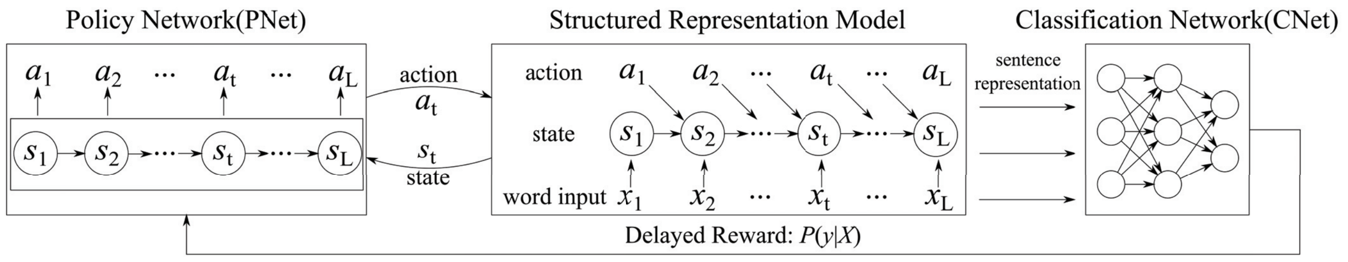


Figure 1: Illustration of the overall process. The policy network (PNet) samples an action at each state. The structured representation model offers state representation to PNet and outputs the final sentence representation to the classification network (CNet) when all actions are sampled. CNet performs text classification and provides reward to PNet.

covers task-relevant structures to build structured sentence representations for text classification problems. We propose two structured representation models: *information distilled LSTM (ID-LSTM)* and *hierarchical structured LSTM (HS-LSTM)*.

- Even without explicit structure annotations, our method can identify task-relevant structures effectively. Moreover, the performance is better or comparable to strong baselines that use pre-specified parsing structures.

Methodology

Overview

The goal of this paper is to learn structured representation for text classification by discovering important, task-relevant structures. We argue that text classification can be improved with an optimized, structured representation.

The overall process is shown in Figure 1. The model consists of three components: **Policy Network** (**PNet**), **structured representation models**, and **Classification Network** (**CNet**). PNet adopts a stochastic policy and samples an action at each state. It keeps sampling until the end of a sentence, and produces an action sequence for the sentence. Then the structured representation models translate the actions into a structured representation. We design two representation models, information distilled LSTM (ID-LSTM) and hierarchically structured LSTM (HS-LSTM). CNet makes classification based on the structured representation and offers reward computation to PNet. Since the reward can be computed once the final representation is available (completely determined by the action sequence), the process can be naturally addressed by policy gradient method (Sutton et al. 2000).

Obviously the three components are interleaved together. The state representation of PNet is derived from the representation models, CNet relies on the final structured representation obtained from the representation model to make prediction, and PNet obtains rewards from CNet's prediction to guide the learning of a policy.

Policy Network (PNet)

The policy network adopts a stochastic policy $\pi(a_t|\mathbf{s_t};\Theta)$ and uses a delayed reward to guide the policy learning. It samples an action with the probability at each state whose representation is obtained from the representation models. In order to obtain the delayed reward which is based on CNet's prediction, we perform action sampling for the entire sentence. Once all the actions are decided, the representation models will obtain a structured representation of the sentence, and it will be used by CNet to compute $P(y|X)$. The reward computed with $P(y|X)$ is used for policy learning.

We briefly introduce state, action and policy, reward, and objective function as follows:

State State encodes the current input and previous contexts, and has different definitions in the two representation models. The detailed definition of state $\mathbf{s_t}$ will be introduced in the following sections.

Action and Policy We adopt binary actions in two settings, but with different meanings. In ID-LSTM, the action space is {*Retain, Delete*}, where a word can be deleted from or retained in the final sentence representation. In HS-LSTM, the action space is {*Inside, End*}, indicating that a word is inside or at the end of a phrase[1]. **Clearly, each action is a direct indicator of structure selection in both representation models.**

We adopt a stochastic policy. Let a_t denote the action at state t, the policy is defined as follows:

$$\pi(a_t|\mathbf{s_t};\Theta) = \sigma(\mathbf{W} * \mathbf{s_t} + \mathbf{b}), \tag{1}$$

where $\pi(a_t|\mathbf{s_t};\Theta)$ denotes the probability of choosing a_t, σ denotes the *sigmoid* function and $\Theta = \{\mathbf{W}, \mathbf{b}\}$ denotes the parameters of PNet.

During training, the action is sampled according to the probability in Eq. 1. During test, the action with the maximal probability (i.e., $a_t^* = argmax_a\pi(a|\mathbf{s_t};\Theta)$) will be chosen in order to obtain superior prediction.

Reward Once all the actions are sampled by the policy network, the structured representation of a sentence is determined by our representation models, and the representation will be passed to CNet to obtain $P(y|X)$ where y is the class label. The reward will be calculated from the predicted distribution ($P(y|X)$), and also has a factor considering the tendency of structure selection, which will be detailed later. This is a typical delayed reward since we cannot obtain it until the final representation is built.

[1]To be precise, *phrase* means a substructure or segment.

Objective Function We optimize the parameters of PNet using REINFORCE algorithm (Williams 1992) and policy gradient methods (Sutton et al. 2000), aiming to maximize the expected reward as shown below.

$$\begin{aligned} J(\Theta) &= \mathbb{E}_{(\mathbf{s_t},a_t)\sim P_\Theta(\mathbf{s_t},a_t)} r(\mathbf{s_1}a_1\cdots\mathbf{s_L}a_L) \\ &= \sum_{\mathbf{s_1}a_1\cdots\mathbf{s_L}a_L} P_\Theta(\mathbf{s_1}a_1\cdots\mathbf{s_L}a_L)R_L \\ &= \sum_{\mathbf{s_1}a_1\cdots\mathbf{s_L}a_L} p(\mathbf{s_1})\prod_t \pi_\Theta(a_t|\mathbf{s_t})p(\mathbf{s_{t+1}}|\mathbf{s_t},a_t)R_L \\ &= \sum_{\mathbf{s_1}a_1\cdots\mathbf{s_L}a_L} \prod_t \pi_\Theta(a_t|\mathbf{s_t})R_L. \end{aligned}$$

Note that this reward is computed over just one sample, say $X = x_1x_2\cdots x_L$. Since our state at step $t+1$ is fully determined by the state and action at step t, the probability $p(\mathbf{s_1})$ and $p(\mathbf{s_{t+1}}|\mathbf{s_t}, a_t)$ are equal to 1.

By applying the likelihood ratio trick, we update the policy network with the following gradient:

$$\nabla_\Theta J(\Theta) = \sum_{t=1}^{L} R_L \nabla_\Theta \log \pi_\Theta(a_t|\mathbf{s_t}). \tag{2}$$

Structured Representation Models

Information Distilled LSTM (ID-LSTM) The main idea of Information Distilled LSTM (ID-LSTM) is to build a sentence representation by distilling the most important words and removing irrelevant words in a sentence. In this way, it is expected to learn more task-relevant representations for classification. For instance, in sentiment classification, words like ‘to’, ‘the’ and ‘a’ may rarely contribute to the task. By distilling the most important words in a sentence, the final representation can be purified and condensed for classification.

ID-LSTM translates the actions obtained from PNet to a structured representation of a sentence. Formally, given a sentence $X = x_1x_2\cdots x_L$, there is a corresponding action sequence $A = a_1a_2\cdots a_L$ obtained from PNet. In this setting, each action a_i at word position x_i is chosen from $\{$*Retain, Delete*$\}$ where *Retain* indicates that the word is retained in a sentence, and *Delete* means that the word is deleted and it has no contribution to the final sentence representation. Formally,

$$\mathbf{c_t}, \mathbf{h_t} = \begin{cases} \mathbf{c_{t-1}}, \mathbf{h_{t-1}}, & a_t = Delete \\ \Phi(\mathbf{c_{t-1}}, \mathbf{h_{t-1}}, \mathbf{x_t}), & a_t = Retain \end{cases} \tag{3}$$

where Φ denotes the functions (including all gate functions and the update function) of a sequence LSTM, $\mathbf{c_t}$ is the memory cell, and $\mathbf{h_t}$ is the hidden state at position t. Note that if a word is deleted, the memory cell and hidden state of the current position are **copied** from the preceding position.

State: The state for the policy network is defined as follows:

$$\mathbf{s_t} = \mathbf{c_{t-1}} \oplus \mathbf{h_{t-1}} \oplus \mathbf{x_t}, \tag{4}$$

where $\oplus$ indicates vector concatenation and $\mathbf{x_t}$ is the current word input. To enrich the state representation, the memory state ($\mathbf{c_{t-1}}$) is included.

To make classification, the last hidden state of ID-LSTM is taken as input to the classification network (CNet):

$$P(y|X) = softmax(\mathbf{W_s}\mathbf{h_L} + \mathbf{b_s}), \tag{5}$$

where $\mathbf{W_s} \in \mathbb{R}^{d\times K}$, $\mathbf{b_s} \in \mathbb{R}^K$ are parameters of CNet, d is the dimension of hidden state, $y \in \{c_1, c_2, \cdots, c_K\}$ is the class label and K is the number of categories.

Reward: In order to compute the delayed reward R_L, we use the logarithm of the output probability of CNet, i.e., $P(y = c_g|X)$ where c_g is the gold label of the input X. In addition, to encourage the model to delete more useless words, we include an additional term by computing the proportion of the number of deleted words to the sentence length:

$$R_L = \log P(c_g|X) + \gamma L'/L, \tag{6}$$

where L' denotes the number of deleted words (where the corresponding action a_t is *Delete*). γ is a hyper-parameter to balance the two terms.

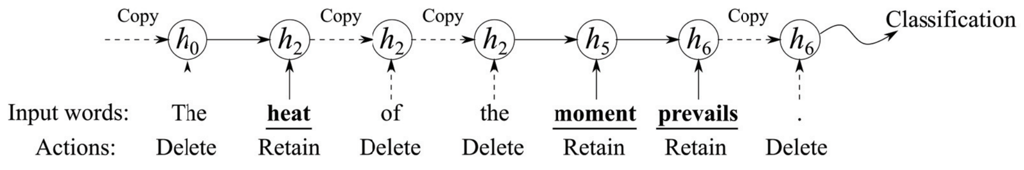
Hierarchically Structured LSTM (HS-LSTM) Hierarchical models have been widely used in document-level classification (Tang, Qin, and Liu 2015; Ghosh et al. 2016) and language modeling (Chung, Ahn, and Bengio 2017). Inspired by these studies, we propose a Hierarchically Structured LSTM (HS-LSTM) that can build a structured representation by discovering hierarchical structures in a sentence. We argue that a better sentence representation can be obtained by identifying sub-structures in a sentence. This process is implemented by sampling an action in $\{$*Inside, End*$\}$ at each word position, where *Inside* indicates that a word is inside of a phrase and *End* means the end of a phrase. HS-LSTM translates the actions to a hierarchical structured representation of the sentence. To be precise, the word **phrase** in this paper, should be interpreted as **substructure or segment**.

In HS-LSTM, there is a two-level structure: a word-level LSTM which connects a sequence of words to form a phrase, and a phrase-level LSTM which connects the phrases to form the sentence representation. The transition of the word-level LSTM depends upon action a_{t-1}. If action a_{t-1} is *End*, the word at position t is the start of a phrase and the word-level LSTM starts with a zero-initialized state. Otherwise the action is *Inside* and the word-level LSTM continues from its previous state. The process is described formally as follows:

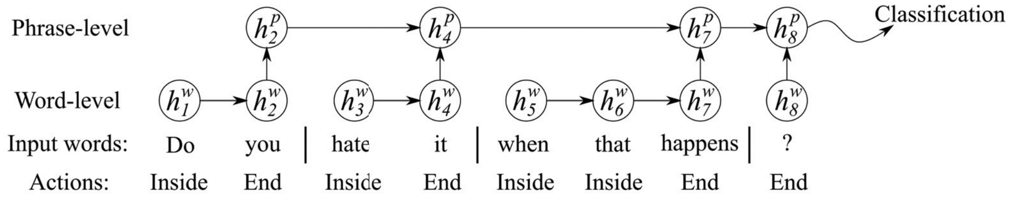
$$\mathbf{c^w_t}, \mathbf{h^w_t} = \begin{cases} \Phi^w(\mathbf{0}, \mathbf{0}, \mathbf{x_t}), & a_{t-1} = End \\ \Phi^w(\mathbf{c^w_{t-1}}, \mathbf{h^w_{t-1}}, \mathbf{x_t}), & a_{t-1} = Inside \end{cases} \tag{7}$$

where Φ^w denotes the transition functions of the word-level LSTM, $\mathbf{c_t}$ is the memory cell, and $\mathbf{h_t}$ is the hidden state at position t.

The transition of the phrase-level LSTM depends on action a_t at the current position, which indicates whether a phrase is completely constructed or not (see Eq. 8). When action a_t is *End*, a phrase ends at position t and the hidden state of the word-level LSTM will be fed into the phrase-level LSTM. Otherwise the action is *Inside* and the phrase-level LSTM is fixed at this step, and the variables are copied


Copy
Copy
Copy
Copy
Classification
Input words: The heat of the moment prevails .
Actions: Delete Retain Delete Delete Retain Retain Delete

(a) Information Distilled LSTM (ID-LSTM)


Phrase-level
Classification
Word-level
Input words: Do you | hate it | when that happens | ?
Actions: Inside End Inside End Inside Inside End End

(b) Hierarchically Structured LSTM (HS-LSTM)

Figure 2: Examples for ID-LSTM and HS-LSTM. In ID-LSTM, unimportant words are removed, and the corresponding hidden states are copied. In HS-LSTM, phrases in a sentence can be discovered and a hierarchical representation is then built by applying a word-level and phrase-level LSTM.

a_{t-1}	a_t	Structure Selection
Inside	Inside	A phrase continues at x_t.
Inside	End	A old phrase ends at x_t.
End	Inside	A new phrase begins at x_t.
End	End	x_t is a single-word phrase.

Table 1: The behavior of HS-LSTM according to action a_{t-1} and a_t.

from the preceding position. Formally,

$$\mathbf{c_t^p}, \mathbf{h_t^p} = \begin{cases} \Phi^p(\mathbf{c_{t-1}^p}, \mathbf{h_{t-1}^p}, \mathbf{h_t^w}), & a_t = End \\ \mathbf{c_{t-1}^p}, \mathbf{h_{t-1}^p}, & a_t = Inside \end{cases} \quad (8)$$

where Φ^p denotes the transitions function of the phrase-level LSTM. Note that the input to the phrase-level LSTM is $\mathbf{h_t^w}$, the hidden state of the word-level LSTM.

The behavior of HS-LSTM relies on **both** action a_{t-1} and a_t, as summarized in Table 1. As can be seen clearly, the combination of action a_{t-1} and a_t indicates the position of word x_t in a phrase.

State: The state for the policy network is defined as follows:

$$\mathbf{s_t} = \mathbf{c_{t-1}^p} \oplus \mathbf{h_{t-1}^p} \oplus \mathbf{c_t^w} \oplus \mathbf{h_t^w}, \quad (9)$$

where $\oplus$ indicates vector concatenation. The state representation consists of both word-level and phrase-level representations.

To make classification, the last hidden state of the phrase-level LSTM ($\mathbf{h_L^p}$) is taken as input to the classification network (CNet):

$$P(y|X) = softmax(\mathbf{W_s}\mathbf{h_L^p} + \mathbf{b_s}). \quad (10)$$

Reward: Similar to ID-LSTM, the reward R_L is based on CNet's prediction and a term indicating the tendency of structure selection. Unlike ID-LSTM's reward which encourages the model to remove as many words as possible, this reward respects that a good phrase structure should contain neither too many nor too few phrases. We thus employ a unimodal function of the number of phrases to reflect the tendency of structure selection. We use the function $f(x) = x + 0.1/x$, which is a unimodal function with minimum at $1/\sqrt{10} = 0.316$. Formally, we have

$$R_L = \log P(c_g|X) - \gamma(L'/L + 0.1L/L'), \quad (11)$$

where L' denotes the number of phrases (the number of action *End*). γ is a hyper-parameter. The second term encourages the number of phrases to be $0.316L$, where there are about 3∼4 phrases for $L = 10$, which is in line with our observations.

Classification Network (CNet)

The classification network produces a probability distribution over class labels based on the structured representation obtained from ID-LSTM or HS-LSTM. CNet is parameterized by $\mathbf{W_s}$ and $\mathbf{b_s}$ as shown in Eq. 5 and Eq. 10 respectively, and will not be repeated here.

In order to train CNet, we adopt cross entropy as loss function:

$$\mathcal{L} = \sum_{X \in \mathcal{D}} -\sum_{y=1}^{K} \hat{p}(y, X) \log P(y|X), \quad (12)$$

where $\hat{p}(y, X)$ is the gold one-hot distribution of sample X, and $P(y|X)$ is the predicted distribution as defined in Eq. 5 and Eq. 10 respectively.

Training Details

Since PNet, the representation model and CNet are interleaved together, they should be trained jointly. As described in Algorithm. 1, the entire training process consists of three steps. We first pre-train the representation model and CNet, and then pre-train PNet while keeping the parameters of the

Algorithm 1: The Training Process

```
1 Pre-train the representation model (ID-LSTM or
    HS-LSTM) and CNet with predefined structures by
    minimizing Eq. 12;
2 Fix the parameters of the strutured representation
    model and CNet, and Pre-train PNet by Eq. 2;
3 Train all the three components jointly until
    convergence;
```

other two models fixed. At last, we jointly train all the three components.

Since training RL from scratch would be extremely difficult and has high variance, we pretrain the RL module with some warm-start structures. For ID-LSTM, we use the original sentence without any deletion to perform pre-training. For HS-LSTM, we split a sentence into phrases shorter than the square root of sentence length, and also use some very simple heuristics. Note that the structures used for pretraining are quite different from the parsing structures used in previous work because parsing structures heavily rely on parsing tools and are error-prone, and thus more expensive.

Experiments

Experimental Setting and Training Details

The dimension of hidden state in the representation models is 300. The word vectors are initialized using 300-dimensional Glove vectors (Pennington, Socher, and Manning 2014) and are updated together with other parameters. To smooth the update of policy gradient, a suppression factor is multiplied to Eq.2 and is set to 0.1. γ is set to $0.05 \times K$ in the reward of ID-LSTM (Eq. 6) and $0.1 \times K$ in the reward of HS-LSTM (Eq. 11), where K is the number of categories.

During the training process, Adam algorithm (Kingma and Ba 2015) is used to optimize the parameters and the learning rate is 0.0005. We adopted Dropout before the classification layer in CNet, with a probability of 0.5. Mini-batch size is 5.

Datasets and Baselines

Datasets We evaluated our models on various datasets for sentiment classification, subjectivity analysis, and topic classification.

- **MR**: This dataset contains positive/negative reviews (Pang and Lee 2005).
- **SST**: Stanford Sentiment Treebank, a public sentiment analysis dataset with five classes (Socher et al. 2013). [2]
- **Subj**: Subjectivity dataset. The task is to classify a sentence as subjective or objective (Pang and Lee 2004).
- **AG**: AG's news corpus[3], a large topic classification dataset constructed by (Zhang, Zhao, and LeCun 2015). The topic includes World, Sports, Business and Sci/Tech.

Baselines We chose three types of baselines: basic neural models using no particular structure, models relying on pre-specified parsing structure, and models distilling important information by attention mechanism.

- **LSTM**: A sequence LSTM. The version we used is proposed in (Tai, Socher, and Manning 2015).
- **biLSTM**: A bi-directional LSTM, commonly used in text classification.
- **CNN**: Convolutional Neural Network (Kim 2014).
- **RAE**: Recursive autoencoder which is defined on predefined parsing structure (Socher et al. 2011).
- **Tree-LSTM**: Tree-structured Long Short-Term Memory relying on predefined parsing structure (Tai, Socher, and Manning 2015).
- **Self-Attentive**: Structured Self-Attentive model, a self-attention mechanism and a special regularization term are used to construct sentence embedding (Lin et al. 2017).

The dimension of hidden vectors and the word vectors used in these baselines are the same to our models. Other parameter settings are consistent with the references.

Models	MR	SST	Subj	AG
LSTM	77.4*	46.4*	92.2	90.9
biLSTM	79.7*	49.1*	92.8	91.6
CNN	81.5*	48.0*	93.4*	91.6
RAE	76.2*	47.8	92.8	90.3
Tree-LSTM	80.7*	**50.1**	93.2	91.8
Self-Attentive	80.1	47.2	92.5	91.1
ID-LSTM	81.6	50.0	93.5	92.2
HS-LSTM	**82.1**	49.8	**93.7**	**92.5**

Table 2: Classification accuracy on different datasets. Results marked with * are re-printed from (Tai, Socher, and Manning 2015), (Kim 2014), and (Huang, Qian, and Zhu 2017). The rest are obtained by our own implementation.

Classification Results

Classification results as listed in Table 2 show that our models perform competitively across different datasets and different tasks. Our models outperform basic models using no structure (LSTM, biLSTM, and CNN) , models using parsing structures (RAE and Tree-LSTM), and attention-based models (Self-Attentive). Comparing to pre-specified parsing structures, automatically discovered structures seem to be more friendly for classification. These results demonstrate the effectiveness of learning structured representations by discovering task-relevant structures.

[2]SST provides both phrase-level and sentence-level annotations. We randomly added 5 labeled phrases for each sentence to the training data, different from (Tai, Socher, and Manning 2015) which used all annotated phrases for each sentence.

[3]http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

Origin text	Cho continues her exploration of the outer limits of raunch with considerable brio .
ID-LSTM	**Cho continues** ~~her~~ **exploration** ~~of the outer limits of~~ **raunch** ~~with~~ **considerable brio .**
HS-LSTM	Cho | continues her exploration | of the outer limits of raunch | with considerable | brio .
Origin text	Much smarter and more attentive than it first sets out to be .
ID-LSTM	**Much smarter** ~~and more~~ **attentive** ~~than it first sets out to be~~ .
HS-LSTM	Much smarter | and more attentive | than it first sets out to be .
Origin text	Offers an interesting look at the rapidly changing face of Beijing .
ID-LSTM	**Offers** ~~an~~ **interesting look** ~~at the~~ **rapidly changing face** ~~of~~ **Beijing .**
HS-LSTM	Offers | an interesting look | at the rapidly changing | face of Beijing | .

Table 3: Examples of the structures distilled and discovered by ID-LSTM and HS-LSTM.

Structure Analysis

To investigate the discovered structures and how they influence classification performance, we presented structure analysis from both qualitative and quantitative perspectives.

ID-LSTM

Qualitative analysis: ID-LSTM can effectively remove the task-irrelevant words, and is still able to offer competitive performance. As shown in Table 3, irrelevant words such as 'and' , 'of' and 'the' are removed by ID-LSTM. Our model can even delete consecutive subsequences in a sentence, such as 'than it first sets out to be' or 'the outer limits of'. It is interesting that the classifier can classify the text correctly even without such irrelevant words. Taking sentiment classification as an example, we observed that the retained words by ID-LSTM are mostly sentiment words and negation words, indicating that the model can distill important, task-relevant words. These examples demonstrate that a purified representation can benefit classification tasks. Thus, we argue that not all words are necessary for a particular classification task.

Dataset	Length	Distilled Length	Removed
MR	21.25	11.57	9.68
SST	19.16	11.71	7.45
Subj	24.73	9.17	15.56
AG	35.12	13.05	22.07

Table 4: The original average length and distilled average length by ID-LSTM in the test set of each dataset.

Quantitative analysis: We presented further analysis on what irrelevant information is removed and what important information is distilled by ID-LSTM. We compared the original sentence length and distilled length given by ID-LSTM in the test set of each dataset, as shown in Table 4. For sentiment classification, we removed about 9.68/7.45/15.56 words from the sentences on MR/SST/Subj respectively. For topic classification, we removed about 22 words (from 35.12 to 13.05) from the sentences on AG. Interestingly, ID-LSTM removes about or more than half of the words from the sentence. This indicates that text classification can be done with highly purified, condensed information. It infers that such classification tasks can be done with only important keywords, while our model has an effect of distilling these important, task-relevant keywords for the task.

Word	Count	Deleted	Percentage
of	1,074	947	88.18%
by	161	140	86.96%
the	1,846	1558	84.40%
's	649	538	82.90%
but	320	25	7.81%
not	146	0	0.00%
no	73	0	0.00%
good	70	0	0.00%
interesting	25	0	0.00%

Table 5: The most/least deleted words in the test set of SST.

Furthermore, we analyzed what types of words are removed and retained, taking sentiment classification as an example. The most and least deleted words by ID-LSTM in the SST dataset are listed in Table 5, ordered by deletion percentage ($Deleted/Count$). On one hand, the most deleted words are non-content words (prepositions, articles, etc.), generally irrelevant to sentiment classification. This shows that ID-LSTM is able to filter irrelevant words. On the other hand, ID-LSTM is able to retain important words for the task. For instance, as we may know, sentiment and negation words are important for sentiment classification (Zhu et al. 2014; Qian et al. 2017). As can be seen in Table 5, sentiment words such as 'good' and 'interesting' are rarely removed. ID-LSTM doesn't remove 'not' or 'no'. For transitional words, 'but' appears 320 times and only 7.81% of them are removed.

To summarize, the qualitative and quantitative results demonstrate that ID-LSTM is able to remove irrelevant words and distill task-relevant ones in a sentence. ID-LSTM is effective to *extract task-specific keywords*, and the purified, condensed representation is classification-friendly.

HS-LSTM

Qualitative analysis:

Table 3 shows some interesting structures discovered by HS-LSTM. In the given examples, phrases such as 'much smarter', 'and more attentive' and 'an interesting book', are important task-relevant phrases. We also observed that structures discovered by HS-LSTM are more flexible in length and content than traditional *phrases* since HS-LSTM sometimes fails to find the correct boundary between phrases. However, as we mentioned, this is extremely difficult for any

Structure	Sentence
Predefined	The film | is | one | of | the year | 's | best | .
Discovered by RL	The film is | one of the year | 's best | .
Predefined	A wonderfully warm human | drama | that | remains vividly in memory | long | after viewing | .
Discovered by RL	A wonderfully | warm | human drama that remains vividly | in memory long after viewing | .
Predefined	The actors are fantastic | . | They | are | what | makes it | worth | the trip | to the theater | .
Discovered by RL	The actors are fantastic | . They are what makes it worth | the trip to the theater .

Table 6: The comparison of the predefined structures and those discovered by HS-LSTM.

model without explicit structure annotations.

In Table 6, we listed some examples to show the difference between the predefined structures (used for pretraining) and those discovered by HS-LSTM. These examples show that our RL method learns to build quite different structures from the predefined ones. The predefined structures are built with some very simple heuristics, and consequently fragmented. In comparison, HS-LSTM tends to discover more complete and longer phrases.

Type	Examples
Noun Phrase	a spiffy animated feature the creative community the originally noble motive
Verb Phrase	coming back quickly realize lost opportunities
Prep. Phrase	from their new home of a complex man of a harmonic family life
Special Phrase	as predictable as the outcome throwing caution to the wind a dozen years later

Table 7: Phrase examples discovered by HS-LSTM.

Nevertheless, HS-LSTM indeed has the ability to identify common types of phrases with clear boundary. We listed more examples of different types found by HS-LSTM in Table 7. Noun and prepositional phrases found by HS-LSTM are usually long, expressive, and task-relevant. In comparison, verb phrases are shorter than noun phrases. Besides grammatical phrases, our model also finds some interesting special phrases, as shown in Table 7.

Quantitative analysis: First of all, we compared HS-LSTM with other structured models to investigate whether classification tasks can benefit from the discovered structure. The baselines include the models that rely on parsing structure, and Com-Tree-LSTM that learns to compose deep tree structure (Yogatama et al. 2017). We also compared with Par-HLSTM which has the same structured representation model except that the phrase structure is given by Stanford parser (Klein and Manning 2003) instead of RL. The results in Table 8 show that HS-LSTM outperforms other structured models, indicating that the discovered structure may be more task-relevant and advantageous than that given by parser.

Then, we computed the statistics of the structures discovered by HS-LSTM in Table 9, including the average phrase number and words per phrase. The average number of words in a phrase is stable across different datasets: about 4 or 5 words per structure. However, this might be accordant with the term for encouraging structure selection (see Eq. 11).

Models	SST-binary	AG's News
RAE	85.7	90.3
Tree-LSTM	87.0	91.8
Com-Tree-LSTM	86.5*	—
Par-HLSTM	86.5	91.7
HS-LSTM	**87.8**	**92.5**

Table 8: Classification accuracy from structured models. The result marked with * is re-printed from (Yogatama et al. 2017).

Dataset	Length	#Phrases	#Words per phrase
MR	21.25	4.59	4.63
SST	19.16	4.76	4.03
Subj	24.73	4.42	5.60
AG	35.12	8.58	4.09

Table 9: Statistics of structures discovered by HS-LSTM in the test set of each dataset.

To summarize, our HS-LSTM has the ability of discovering *task-relevant structures* and then building better structured sentence representations. The qualitative and quantitative results demonstrate that these discovered structures are task-friendly and suitable for text classification.

Conclusion

This paper has presented a reinforcement learning method which learns sentence representation by discovering task-relevant structures. In the framework of RL, we adopted two representation models: ID-LSTM that distills task-relevant words to form purified sentence representation, and HS-LSTM that discovers phrase structures to form hierarchical sentence representation. Extensive experiments show that our method has state-of-the-art performance and is able to discover interesting task-relevant structures without explicit structure annotations.

As future work, we will apply the method to other types of sequences since the idea of structure discovery (or restructuring the input) can be generalized to other tasks and domains.

Acknowledgment

This work was partly supported by the National Science Foundation of China under grant No.61272227/61332007. We would like to thank Prof. Xiaoyan Zhu for her generous support to this work.

References

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35(8):1798–1828.

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*, 632–642.

Chung, J.; Ahn, S.; and Bengio, Y. 2017. Hierarchical multiscale recurrent neural networks. In *ICLR*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop*.

Ghosh, S.; Vinyals, O.; Strope, B.; Roy, S.; Dean, T.; and Heck, L. 2016. Contextual lstm (clstm) models for large scale nlp tasks. In *SIGKDD workshop Oral presentation*.

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

Huang, M.; Qian, Q.; and Zhu, X. 2017. Encoding syntactic knowledge in neural networks for sentiment classification. *ACM Transactions on Information Systems (TOIS)* 35(3):26.

Iyyer, M.; Manjunatha, V.; Boyd-Graber, J.; and Daumé III, H. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL*, 1681–1691.

Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2017. Bag of tricks for efficient text classification. In *EACL*, 427–431.

Kalchbrenner, N.; Grefenstette, E.; and Blunsom, P. 2014. A convolutional neural network for modelling sentences. In *ACL*, 655–665.

Kim, Y. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, 1746–1751.

Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Klein, D., and Manning, C. D. 2003. Accurate unlexicalized parsing. In *ACL*, 423–430.

Le, Q., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *ICML*, 1188–1196.

Lei, T.; Barzilay, R.; and Jaakkola, T. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. In *EMNLP*, 1565–1575.

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 *ICLR*.

Liu, B.; Huang, M.; Sun, J.; and Zhu, X. 2015. Incorporating domain and sentiment supervision in representation learning for domain adaptation. In *IJCAI*, 1277–1283.

Pang, B., and Lee, L. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*, 271.

Pang, B., and Lee, L. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, 115–124.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, 1532–1543.

Qian, Q.; Tian, B.; Huang, M.; Liu, Y.; Zhu, X.; and Zhu, X. 2015. Learning tag embeddings and tag-specific composition functions in recursive neural network. In *ACL*, 1365–1374.

Qian, Q.; Huang, M.; Lei, J.; and Zhu, X. 2017. Linguistically regularized lstm for sentiment classification. In *ACL*, 1679–1689.

Socher, R.; Pennington, J.; Huang, E. H.; Ng, A. Y.; and Manning, C. D. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, 151–161.

Socher, R.; Perelygin, A.; Wu, J. Y.; Chuang, J.; Manning, C. D.; Ng, A. Y.; Potts, C.; et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, volume 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 *NIPS*, 1057–1063.

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

Tang, D.; Qin, B.; and Liu, T. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, 1422–1432.

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. J.; and Hovy, E. H. 2016. Hierarchical attention networks for document classification. In *NAACL-HLT*, 1480–1489.

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

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *NIPS*, 649–657.

Zhou, X.; Wan, X.; and Xiao, J. 2016. Attention-based lstm network for cross-lingual sentiment classification. In *EMNLP*, 247–256.

Zhu, X.; Guo, H.; Mohammad, S.; and Kiritchenko, S. 2014. An empirical study on the effect of negation words on sentiment. In *ACL*, 304–313.

Zhu, X.; Sobihani, P.; and Guo, H. 2015. Long short-term memory over recursive structures. In *International Conference on Machine Learning*, 1604–1612.