Ensemble Neural Relation Extraction with Adaptive Boosting

Dongdong Yang[1], **Senzhang Wang**[2], **Zhoujun Li**[3]
[1] Unversity of Southern California
[2] Nanjing University of Aeronautics and Astronautics
[3] Beihang University
dongdony@usc.edu, szwang@nuaa.edu.cn, lizj@buaa.edu.cn

Abstract

Relation extraction has been widely studied to extract new relational facts from open corpus. Previous relation extraction methods are faced with the problem of wrong labels and noisy data, which substantially decrease the performance of the model. In this paper, we propose an ensemble neural network model - Adaptive Boosting LSTMs with Attention, to more effectively perform relation extraction. Specifically, our model first employs the recursive neural network LSTMs to embed each sentence. Then we import attention into LSTMs by considering that the words in a sentence do not contribute equally to the semantic meaning of the sentence. Next via adaptive boosting, we build strategically several such neural classifiers. By ensembling multiple such LSTM classifiers with adaptive boosting, we could build a more effective and robust joint ensemble neural networks based relation extractor. Experiment results on real dataset demonstrate the superior performance of the proposed model, improving F1-score by about 8% compared to the state-of-the-art models. The code of this work is publicly available on https://github.com/RE-2018/re.

1 Introduction

Many NLP tasks have been built on different knowledgebases, such as Freebase[Bollacker *et al.*, 2008] and DBPedia[Auer *et al.*, 2007]. However, the knowledgebases could not cover all the facts in the real world. Therefore, it is essential to extract more common relational facts automatically in open domain corpus. As known, relation extraction (RE) aims at extracting new relation instances that are not contained in the knowledgebases from the unstructured open corpus. It aligns the entities in the open corpus with those in the knowledgebases and retrieves the entity relations from the real world. For example, if we aim to retrieve a relation from the raw text, "*Barack Obama married Michelle Obama 10 years ago*", a naive approach would be to search the news articles for indicative phrases, such as "*marry*" or "*spouse*". However, the result may be wrong since human language is inherently various and ambiguous.

Previous supervised RE methods require a large amount of labelled relation training data by human-hand. To address this issue, Mintz et al. [Mintz *et al.*, 2009] proposed an approach via aligning the entity in KB for later extraction without plenty of training corpus. However, their assumption - there is only one relation existing in a pair of entities, was irrational. Therefore, later researches assumed more than one relation could exist between a pair of entities. Hoffmann et al. [Hoffmann *et al.*, 2011] proposed a multi-instance learning model with overlapping relations (MultiR) that combined a sentence-level extraction model for aggregating the individual facts. Surdeanu et al. [Surdeanu *et al.*, 2012] proposed a multi-instance multi-label learning model (MIML-RE) to jointly model the instances of a pair of entities in text and all their labels. The major limitation of the above methods is that they cannot deeply capture the latent semantic information from the raw text. It is also challenging for them to seamlessly integrate semantic learning with feature selection to more accurately perform RE.

Recently, deep neural networks are widely explored for relation extraction and have achieved significant performance improvement [Zeng *et al.*, 2015; Lin *et al.*, 2016]. Compared with traditional shallow models, deep models can deeply capture the semantic information of a sentence. Zeng et al. [Zeng *et al.*, 2015] combined the multi-instance learning with piecewise convolutional neural networks to learn more relevant features. Lin et al. [Lin *et al.*, 2016] employed CNN with sentence-level attention over multiple instances to encode the semantics of sentences. Miwa and Bansal [Miwa and Bansal, 2016] used a syntax-tree-based long short-term memory networks (LSTMs) on the sentence sequences. Ye et al.[Ye *et al.*, 2017] proposed a unified relation extraction model that combined CNN with a pair of ranking class ties. However, the main issue of existing deep models is that their performance may not be stable and could not effectively handle the quite imbalanced, noisy, and wrong labeled data in relation extraction even though a large number of parameters in the model.

To address the above issues, in this paper we propose a novel ensemble deep neural network model to extract relations from the corpus via an Adaptive Boosting LSTMs with Attention model (Ada-LSTMs). Specifically, we first choose bi-directional long short-term memory networks to embed forward and backward directions of a sentence for bet-

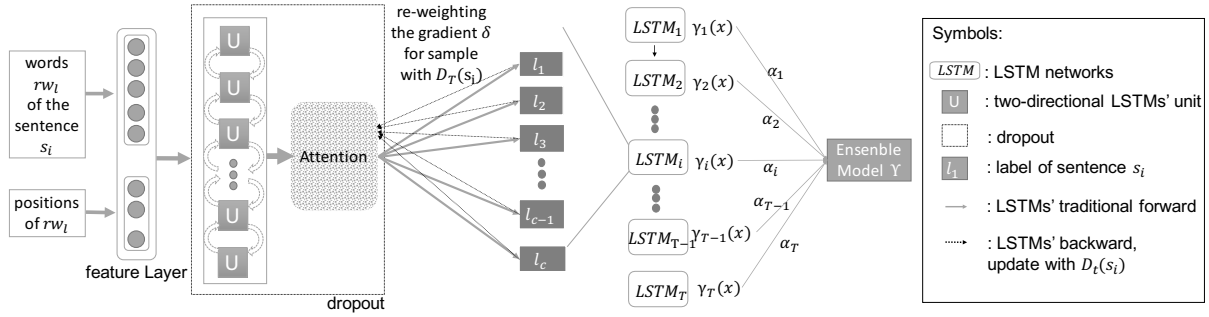

Figure 1: The framework of Ada-LSTMs contains three layers: feature layer, bi-directional Stacked LSTMs' layer with attention and adaptive boosting layer. An input sentence s_i indicates the original sentence with a pair of entities and their relation.

ter understanding the sentence semantics. Considering the fact that the words in a sentence do not contribute equally to the sentence representation, we import attention mechanism to the bi-directional LSTMs. Next we construct multiple such LSTM classifiers and ensemble their results as the final prediction result. Kim and Kang [Kim and Kang, 2010] showed that ensemble with neural networks perform better than one single neural network in prediction tasks. Motivated by their work, we import adaptive boosting and tightly couple it with deep neural networks to more effectively and robustly solve the relation extraction problem. The key role of adaptive boosting in our model is to re-weight the gradient of the training process for the samples. It could adjust the training data distribution based on the performance of the classifier. In this way, those samples which are classified wrong will gain more attention in the following training. Note that attention can distinguish the different importances of the words in a sample sentence, while adaptive boosting can distinguish the different data contributions of the sample sentences to the parameters of the neural networks. The combination of the two can more precisely capture the semantic meaning of the sentences and better represent them, and thus help us train a more accurate robust model.

We summarize the contributions of this paper as follows.

- We propose a Multi-class Adaptive Boosting Neural Networks model, which to our knowledge is the first work that combines adaptive boosting and neural networks for relation extraction.
- We utilize adaptive boosting to tune the gradient descent in NN training. In this way, a large number of parameters in a single NN can be learned more robustly. The ensembled results on multiple NN models can achieve more accurate and robust relation extraction result.
- We evaluate the proposed model on a real data set. The results demonstrate the superior performance of the proposed model which improves F1-score by about 8% compared to state-of-the-art models.

2 Related Work

As an important and fundamental task in NLP, relation extraction has been studied extensively. Many approaches for RE have been developed including distant supervision, deep learning, etc. Distant supervision was firstly proposed to address this issue by [Mintz *et al.*, 2009]. Mintz et al. aligned Freebase relations with Wikipedia corpus to automatically extract instances from a large-scale corpus without hand-labeled annotation. Riedel et al. [Riedel *et al.*, 2010] tagged all the sentences with at least one relation instead of only one. Hoffmann et al. [Hoffmann *et al.*, 2011] also improved the previous work and aimed at solving the overlapping relation problem. Surdeanu et al. [Surdeanu *et al.*, 2012] proposed a multi-instance multi-label method for relation extraction. Angeli et al. [Angeli *et al.*, 2014] put forward a new criterion to sample examples for the distant supervised learning.

With neural networks bursting out many fields of research, researchers also began to apply this new technique to relation extraction. Zeng et al. [Zeng *et al.*, 2014] first proposed a convolutional neural network (CNN) for relation classification. Zhang et al. [Zhang *et al.*, 2015] proposed to utilize bidirectional long short-term memory networks to model the sentence with sequential information about all words.

Recently, attention has been widely used in NLP tasks. Yang et al. [Yang *et al.*, 2016] used a two-layer attention mechanism for document classification, which inspires us to focus on the word understanding level. Liu et al. Lin et al. [Lin *et al.*, 2016] also used the attention level to its CNN architecture and gained a better performance in extraction. Besides, ensemble learning is a well-known machine learning paradigm which tries to learn one hypothesis from training data, ensemble methods. Ratsch, Onoda, and Muller [Rätsch *et al.*, 2001] proposed several regularization methods and generalizations of the original adaptive boosting algorithm to achieve a soft margin. Freund and Schapire et al. [Freund *et al.*, 1996] was the first paper which proposed adaboost. Our paper is developped based on their job. Rokach et al [Rokach, 2010] showed us the technique of generating multiple models strategically and combining these models to improve the performance of many machine learning tasks.

3 Methodology

Given a sentence $s_i \in \mathcal{S}$, where $\mathcal{S}$ is a corpus, and its corresponding pair of entities $\varphi = (e_1, e_2)$, our model aims at measuring the probability of each candidate relation $\Omega_i \in \mathcal{L}$. $\mathcal{L}$ is defined as $\{1, 2, 3, ..., C\}$, where C is the number of relation classes.

Table 1: Notations and their meanings.

Notations	Interpretation
Y	the final trained classification model
$\gamma_t(x)$	a neural network classifier
T	total number of trained neural network classifiers
α_t	the weight of the neural classifier $\gamma_t(x)$
D_t	the weight vector of total samples for their contributions to gradient descent of the t^{th} neural network
$D_t(s_i)$	the weight of sentence s_i for its contribution to gradient descent of the t^{th} neural network
s_i	the i^{th} sentence in the corpus $\mathcal{S}$
rw_k	the k^{th} word in a sentence s_i
x_k	the word embedding for the k^{th} word
d_k^p	the position embeddings for the k^{th} word

Figure 1 shows the overview of the model framework. The model mainly consists of three layers: feature layer, bi-directional LSTMs layer with an attention and adaptive boosting layer. The feature layer makes sentence vectorized and embeds them as the input of the model. The bi-directional LSTMs with attention layer can deeply capture the latent information of each sentence. Attention mechanism could weight each phase in a sentence, which is learned during the training process. The adaptive boosting layer combines multiple classifiers to generate the final weighted joint function for classifying the relation. The essential notations used in this paper and their meanings are given in Table 1. Next, we will introduce the three layers of the proposed model in details in the following sections.

3.1 Embedded Features

The embedded features contain word embeddings and position embeddings. We use two embedded features for relation extraction as the input of the bi-directional long short-term memory neural networks. We describe the embedding features as follows.

Word Embeddings

The inputs are some raw words $\{rw_1,rw_2,...,rw_l\}$, where l is the length of the input sentence. We make every raw word rw_i represented by a real-valued vector w_i via word embedding which is encoded by an embedding matrix $M \in \mathbb{R}^{d^a \times V}$, where V is the representation of a fix-sized vocabulary and d^a is the dimension of the word embedding. In our paper, we use the skip-gram model to train word embeddings.

Position Embeddings

A position embedding is defined as a word distance, which is from the position of the word to the positions of the entities in a sentence. A position embedding matrix is denoted as $P \in \mathbb{R}^{l^p \times d^p}$, where l^p is the number of distances and d^p is the dimension of the position embedding proposed by Ye et al. [Ye *et al.*, 2017]. As there are two entities in a sentence that we need to measure their distances to the word, we have two d^p values. Therefore, the dimension of the word representations is $d^w = d^a + 2 \times d^p$ and the final input vector for raw word rw_i is $x_i = [w_i, d_1^p, d_2^p]$.

3.2 Multi-class Adaptive Boosting Neural Networks

The Multi-class Adaptive Boosting Long Short-term Memory Neural Networks (Ada-LSTMs) is a joint model, in which several neural networks are combined together according to their weight vector α, learned from adaptive boosting algorithm. Before describing the model in detail, we would like to show the motivation for coming up with this model. We first analyse the distribution of a public dataset for relation extraction, which is currently widely used as the benchmark and released by [Riedel *et al.*, 2010]. The data distribution is quite unbalanced. Among the 56 relation ties, 32 of them have less than 100 samples and 12 of them have more than 1000 samples. Besides, as [Liu *et al.*, 2016] discussed, the dataset has wrong labelling and noisy data problems. Thus it is difficult for a single model to achieve promising result on relation extraction with such noisy and distorted training data. Therefore, it is essential to introduce a robust algorithm to alleviate the wrong labelling data issue and the distortions of the data.

In our model, we adopt multi-class adaptive boosting method to improve the robustness of the neural networks for relation extraction. For the neural networks part, we use LSTMs because it is naturally suitable to handle the sequential words in a sentence and captures the meanings well. For the ensemble learning part, adaboost is a widely used ensemble learning method that sequentially trains and ensembles multiple the classifiers. The t^{th} classifier is trained with more emphasis on different weights on the input samples, which is based on a probability distribution D_t. The original adaptive boosting is to solve the binary classification problems and calculate the samples one by one. To make it fit into our model, we make the following modifications as shown in Equations (1)-(8).

The final prediction model Y is obtained by weighted voting as shown in (1), where α_t means the weight of each classifier $\gamma_t(x)$ for our final extractor $Y(x)$. The softmax function f in Equation (1) is to predict the labels of relation types. Here we focus more on the upper level of the model architecture, and more details about the neural classifier $\gamma_t(x)$ will be given in the next section. The result of training the t^{th} classifier is such a hypothesis $h_t : X \to L$ where X is the space of input features and $L = \{1, ..., c\}$ is the space of labels.

$$Y(x) = f(\sum_{t}^{T} \alpha_t \gamma_t(x)) \tag{1}$$

The weight α_t of each NN classifier $\gamma_t(x)$ is updated based on its training error ϵ on the training set as shown in Equation (2). After the t^{th} round the weighted error ϵ_t of the resulting classifier is calculated.

$$\alpha_t = \frac{1}{2} ln \frac{1 - \epsilon_t}{\epsilon_t} \tag{2}$$

During the training process, the weight α of each classifier is learned by a parameter vector D_t. Initially, the element of vector D_t is assigned the equal value for each sentence in the dataset as $D_1(b_i) = \frac{1}{n}$, where n is the number of batches and

b_i is the i^{th} batch of all the samples. In our case we process the samples batch by batch.

The Equations (3)-(4) show how to calculate the training error ϵ. $error(I(\gamma(k) \neq y_k))$ means that when the model output $\gamma(k)$ is not equal to its true label y_k in a batch b_i, we gather the error of that sample. I is the error indicator and K is the batch size. Finally we average the error τ of each batch j as shown in Equation (4).

$$\epsilon = \sum_j e^{\tau_j}, \tau < \frac{1}{2} \tag{3}$$

$$\tau_j = \frac{\sum_k^K error(I(\gamma(k) \neq y_k))}{K} \tag{4}$$

After the calculation of the weight α of each classifier, we could use it to update the D_t as shown in Equations (5)-(6), where Z_t is a normalization constant. D_t is the weight vector for the samples at the epoch t. D_{t+1} is computed from D_t by increasing the probability of incorrectly labeling samples. We maintain the weight $D_t(b_i)$ for the batch b_i during the learning process. Then, we could use it to inform the training process of neural networks, by setting a constraint to gradient descent during back propagation of neural networks.

$$c(x) = \begin{cases} e^{\alpha_t}, & \tau < \frac{1}{2} \\ e^{-\alpha_t}, & \tau \geq \frac{1}{2} \end{cases} \tag{5}$$

$$D_{t+1}(b_i) = \frac{D_t(b_i)}{Z_t} c(x) \tag{6}$$

By combing Equations (2)-(6), we have Equation (7). During the training process, if a training sample in batch b_i is trained enough and has been fitted well, its weight $D_t(b_i)$ will drop. Otherwise, the algorithm will increase its weight $D_t(b_i)$ to let it contribute more to the gradient descent of training model. In other words, when the sentences in a batch as a whole are hard to classify, the weight $D_t(b_i)$ will increase correspondingly [Freund *et al.*, 1996]. In this way, we learn D_t as the weights of the samples' impact on the neural networks. Finally, multiple NN classifiers are learned and combined as a joint relation extractor.

$$D_{t+1}(b_i) = \frac{D_t(b_i)}{Z_t} e^{-\alpha_t y_i \gamma_t(x_i)} \tag{7}$$

More details are needed here to explain how D_t affects the the gradient descent in back propagation of the neural networks. We assign the parameter $D_t(b_i)$ to the gradient of the back propagation as shown in Equation (8). Then the whole neural network will update its parameters via back propagation with D_t.

$$\delta_{new} = \delta_{old} \times D_t \times \beta \tag{8}$$

The pseudocode of the Ada-LSTMs model [1] is given in Algorithm 1. m is the total number of training data. n is the number of batches. ϵ_t is the training error of the training samples. δ_{old} is the final layer derivative in back propagation and δ_{new} is the new derivative in back propagation used to update the networks. $\beta = \frac{1}{max(D_t)}$ is the reciprocal of maximal $D_t(b_i)$ value, where i is the index of batch number and s is a sentence in the corpus $\mathcal{S}$. β is a coefficient, aiming at avoiding D_t too small to update the NN. g is a mapping function. f is a softmax function. $Att\text{-}LSTMs$ is the LSTMs with selective attention model, which will be described later.

[1]The code of our model is publicly available at https://github.com/RE-2018/re

Algorithm 1 Ada-LSTMs Model for Relation Extraction.

Input: (s_1, φ_1, Ω_1), (s_2, φ_2, Ω_2),...,(s_m, φ_m, Ω_m), where $s_i \in \mathcal{S}$ is a sentence in the sentence set $\mathcal{S}$, φ_i is a pair of entities and $\Omega_i \in \mathcal{L}$ is their relation tie.
Output: final weighted extrator $Y(x)$

for $t = 1$ to T **do**
  init D_t on $\{1, ..., n\}$
  for s in $\mathcal{S}$ **do**
    look up embedding x for words in s
    $Att\text{-}LSTMs$ FORWARD (x)
    update δ based on Equation (8)
    $Att\text{-}LSTMs$ BACKWARD
    calculate training error ϵ_t of γ_t:
      $\epsilon_t = Pr_{D_t}[\gamma_t(x_i) \neq y_i]$
    select classifier with smallest error ϵ_t on D_t
    calculate α_t, $c(x)$ based on Equation (2)-(5)
    $D_{t+1} = g(D_t, \alpha_t, \Omega, \gamma_t)$
    $\gamma_t : X \rightarrow \mathcal{L}$
  end for
end for
final prediction model: $Y(x) = f(\sum_t^T \alpha_t \gamma_t(x))$

LSTMs with Selective Attention (Att-LSTMs)

In this part, as shown in algorithm 1, we elaborate more details of the proposed neural networks with selective attention (Att-LSTMs), which more specifically is attention-based long short-term neural networks. The recursive neural networks have shown in marvelous priority in modeling sequential data [Miwa and Bansal, 2016]. Therefore, we make use of LSTMs to deeply learn the semantic meaning of a sentence which is composed of a sequence of words for relation extraction.

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ g_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} T_{d^w+d,d} \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \tag{9}$$

$$\begin{aligned} c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot tanh(c_t) \end{aligned} \tag{10}$$

The LSTM's unit is summarized in Equation (9). A sentence is initially vectorized into a sequence of encoded words $\{x_1, x_2, ..., x_l\} \in \mathbb{R}^{d^w}$, where l and d^w are the lengths of the input sentence and the dimension of word representations, respectively. d represents the LSTM dimentionality. As Equations (9)-(10) show, i_t, f_t, c_t, o_t, h_t are the input, forget, memory, output gate and hidden state of the cell at time t, respectively. The current memory cell state c_t is the combination of c_{t-1} and g_t, weighted by i_t and f_t, respectively. σ denotes a non-linear activation function. $\odot$ means the elementswise multiplication. d denotes the dimensionality of LSTM.

In our implementation of relation extraction, an input sentence is tagged with the target entities and the relation type. For further usage, we concatenate the current memory cell hidden state vector h_t of LSTM from two directions as the output vector h_k=[$\overrightarrow{h_t}$,$\overleftarrow{h_t}$] at time t. Combining two directions of the sentence could better utilize the features to predict the relation type.

We add an attention model [Xu *et al.*, 2015] to neural networks. The idea of attention is to select the most important piece of information. Since not all words contribute equally to the sentence representation, the important meaning of the sentence could be presented by the informative words to form a more effective vector representation via attention. Finally, we dropout our architecture on both attention layer and bi-directional LSTMs layer.

$$\alpha_t = \frac{exp(e_t)}{\sum_{t=1}^{l} exp(e_t)} = \frac{exp(f_a(h_t))}{\sum_{t=1}^{l} exp(f_a(h_t))} \tag{11}$$

$$c = \sum_{t=1}^{l} \alpha_t h_t \tag{12}$$

The attention mechanism is shown in Equations (11)-(12). For each word location t, f_a is a function learned during training. Specifically, $e_t = f_a(h_t) = \sigma(W h_t + b)$, where W and b will be learned during training and σ is a non-linear function. Then we get a normalized importance weight α_t through a softmax function. l means the length of the sentence sample. Then, we compute the sentence vector c as a sum of adaptive weighted average of state sequence h_t. In this way, we could selectively integrate information word by word with attention mechanism.

$$L_0 = -\sum_{k=1}^{n}\sum_{i=1}^{C} y_{ki} log(q_{ki}) \tag{13}$$

Finally, we use cross entropy [De Boer *et al.*, 2005] to design our loss function L_0 as shown in Equation (13). n is the total number of samples. C is the number of labels. $q = f(c)$, where c is the output of attention layer and f is the softmax function. Our training goal is to minimize L_0.

3.3 Implementation Details

Learning Rate

We followed the method referred to [Kingma and Ba, 2014] to decay the learning rate. The adaptive learning rate decay method is definded as $lr_t \Leftarrow lr_{t-1} * \frac{\sqrt{1-\beta_2^2}}{1-\beta_1^t}$, where lr_t, lr_{t-1} are the current and the last learning rates, respectively.

L2 Regulation

L2 regulation imposes a penalty on the loss goal L_0. For the training goal, we use a negative log likelihood of the relation labels for the pair of entities as function loss. The L2 regulation is as $L2 = \lambda \sum_{i=1}^{n} W_i^2$. It should have the same order of magnitude so that L2 regulation would not weight too much or too little in the training process. We set the constant λ based on the above rule.

Table 2: Parameter Settings

Number of epochs	40
LSTMs' unit size	350
Dropout probability	0.5
Batch size	50
Position dimension	5
Word dimension	50
Unrolled steps of LSTMs	70
Number of neural networks	20
Initial learning Rate	10^{-3}
L2 regulation Coefficient	10^{-4}

4 Experiments

4.1 Dataset

We evaluate our model on the public dataset [2], which is developed by [Riedel *et al.*, 2010]. The dataset was generated via aligning the relations in Freebase with the New York Times corpus (NYT). The dataset induces the relationship for entities of NYT corpus into 56 relationships. The training part is gained by aligning the sentences from 2005 to 2006 in NYT and contains 176,662 non-repeated sentences, among which there are 156,662 positive samples and 20,000 no-answer (NA) negative samples. The testing part is gained in 2007 and contains 6,944 non-repeated samples, among which there are 6,444 positive samples and 500 NA negative samples.

4.2 Experiment Settings

Word Representations

Similar to [Ye *et al.*, 2017], we keep the words that appear more than 100 times to construct word dictionary. In our paper, the vocabulary size of our dataset is 114,042. We use word2vec[3] to train the word embedding on the NYT corpus. We set word-embedding to be 50-dimensional vectors. Additionally, the vectors will concatenate two position embedding, 2×5 dimensional vector, as its final word embedding.

Hyper-parameter settings

Table 2 shows the parameter settings. We set some parameters empirically, such as the batch size, the word dimension, the number of epochs. We set the weights of L2 penalty as 10^{-4} and the learning rate as 10^{-3}, which both are chosen from $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. We select 350 LSTM's units based on our empirically parameter study from the set $\{250, 300, 350, 400, 450\}$. The selection for the number of classifiers will be discussed in the experiment results.

4.3 Evaluation

To evaluate the proposed method, we select the following state-of-the-art feature-based methods for comparison through held-out evaluation:

Mintz [Mintz *et al.*, 2009] is a traditional distant supervised model via aligning relation data on Freebase.

MultiR [Hoffmann *et al.*, 2011] is a graphical model of multi-instance to handle the overlapping relations problem.

[2] http://iesl.cs.umass.edu/riedel/ecml/

[3] http://code.google.com/p/word2vec

Table 3: P@N comparison with state-of-the-art methods.

P@N(%)	One				Two				All			
	100	200	300	Avg	100	200	300	Avg	100	200	300	Avg
CNN+ATT	76.2	65.2	60.8	67.4	76.2	65.7	62.1	68.0	76.2	68.6	59.8	68.2
PCNN+ATT	73.3	69.2	60.8	67.8	77.2	71.6	66.1	71.6	76.6	73.1	67.4	72.2
Rank+ExATT	-	-	-	-	-	-	-	-	83.5	82.2	78.7	81.5
Ada+LSTM	**82.0**	**81.0**	**76.7**	**79.9**	**85.0**	**80.5**	**77.6**	**81.0**	**95.0**	**92.5**	**92.0**	**93.1**

MIML [Surdeanu *et al.*, 2012] jointly models both multiple instances and multiple relations.

CNN+ATT, PCNN+ATT [Lin *et al.*, 2016] add attention mechanism to CNN and PCNN models which are proposed by Zeng et al. [Zeng *et al.*, 2014; Zeng *et al.*, 2015]. Compared to CNN, PCNN adopts convolutional architecture with piecewise max pooling to learn relevant features.

Rank+ExATT Ye et al. [Ye *et al.*, 2017] aggregate ranking method to attention CNN model.

In our experiments, we run the model nearly 40 epochs. Each epoch has 3533 steps (batches). At the first 10 epochs, the loss of the model drops quickly and then the loss becomes relatively stable. Therefore, in the following experiments, we select the Ada-LSTMs model with 10-30 rounds training steps as the final joint extractor for relation extraction.

We compare our Ada-LSTMs model with the above baselines and the Precision Recall (PR) curves are shown in Figure 2. From the result, one could conclude that:

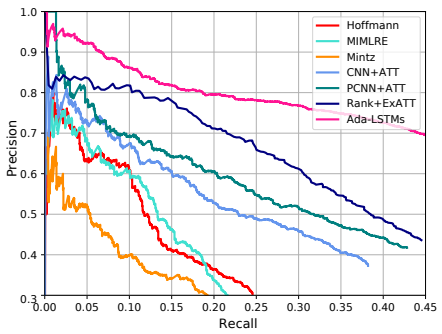

Figure 2: Precision-Recall curves of different methods.

(1) Our proposed method Ada-LSTMs outperforms all the baseline methods. The F1-score of our model is 0.54, which is the highest and outperforms the latest state-of-the-art model Rank+ExATT by nearly 8%.

(2) Our method Ada-LSTMs has a more robust performance because the precision-recall curve is more smooth than other methods. With the increase of recall, the decay tendency of precision is obviously slower than others. Especially when recall is low, the precision of Ada-LSTMs still performs well unlike the others dropping rapidly.

We next evaluate our model via the precision@N(P@N), which means the top N precisions of the results, as shown in Table 3. One, Two, All mean that we randomly select one, two and use all the sentences for each entity pair, respectively. Here we only select the top 100, 200, 300 precisions for our experiment. Experiment data of other methods (CNN+ATT, PCNN+ATT, Rank+ExATT) are obtained from their published papers. The results show that our model outperforms all the baselines in P@N(One, Two, All, Average). Compared to Rank+ExATT, the latest state-of-the-state model, our model has an a significant improvement on the average of P@100, P@200, P@300 by about 11.6% on average.

The effect of classifier number

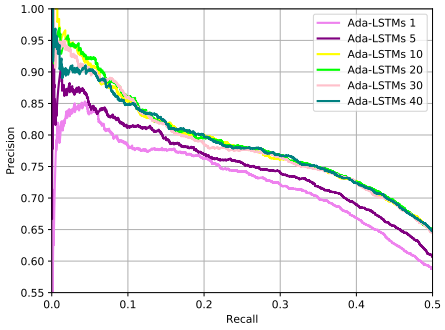

Figure 3: Precision-Recall curves of different classifiers number.

To study the impact of the classifier number on our model performance, we set different numbers of classifiers. The result is given in Figure 3. One can see that when the classifier number of Ada-LSTMs is relatively small, the algorithm performance increases significantly with the increase of classifier number. Ada-LSTMs 10 $>$ Ada-LSTMs 5 $>$ Ada-LSTM 1. However, when the classifier number becomes large, the performance improvement gets less significant. The PR curves of Ada-LSTMs with 10, 20, 30, and 40 classifiers are quite similar. As mentioned, adaptive boosting plays two roles in our model due to ensembling the models and updating the gradient descent during the back propagation of neural networks via re-weighting.

5 Conclusions

In this paper, we proposed to integrate attention-based LSTMs with adaptive boosting model for relation extraction. Compared to the previous models, our proposed model is more effective and robust. Experimental results on the widely used dataset show that our method significantly outperforms the baselines. In the future, it would be interesting to apply to the proposed framework to other tasks, such as image retrieval and abstract extraction.

References

[Angeli *et al.*, 2014] Gabor Angeli, Julie Tibshirani, Jean Wu, and Christopher D Manning. Combining distant and partial supervision for relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1556–1567, 2014.

[Auer *et al.*, 2007] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[Bollacker *et al.*, 2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008.

[De Boer *et al.*, 2005] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[Freund *et al.*, 1996] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *The International Conference on Machine Learning*, volume 96, pages 148–156. Bari, Italy, 1996.

[Hoffmann *et al.*, 2011] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics, 2011.

[Kim and Kang, 2010] Myoung-Jong Kim and Dae-Ki Kang. Ensemble with neural networks for bankruptcy prediction. *Expert systems with applications*, 37(4):3373–3379, 2010.

[Kingma and Ba, 2014] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Lin *et al.*, 2016] Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Neural relation extraction with selective attention over instances. Association for Computational Linguistics, 2016.

[Liu *et al.*, 2016] Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. Learning natural language inference using bidirectional lstm model and inner-attention. *arXiv preprint arXiv:1605.09090*, 2016.

[Mintz *et al.*, 2009] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009.

[Miwa and Bansal, 2016] Makoto Miwa and Mohit Bansal. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*, 2016.

[Rätsch *et al.*, 2001] Gunnar Rätsch, Takashi Onoda, and K-R Müller. Soft margins for adaboost. *Machine learning*, 42(3):287–320, 2001.

[Riedel *et al.*, 2010] Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer, 2010.

[Rokach, 2010] Lior Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1):1–39, 2010.

[Surdeanu *et al.*, 2012] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics, 2012.

[Xu *et al.*, 2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *The International Conference on Machine Learning*, pages 2048–2057, 2015.

[Yang *et al.*, 2016] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

[Ye *et al.*, 2017] Hai Ye, Wenhan Chao, and Zhunchen Luo. Jointly extracting relations with class ties via effective deep ranking. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

[Zeng *et al.*, 2014] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. Relation classification via convolutional deep neural network. In *Proceedings of 24th International Conference on Computational Linguistics (COLING)*, pages 2335–2344, 2014.

[Zeng *et al.*, 2015] Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 17–21, 2015.

[Zhang *et al.*, 2015] Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. Bidirectional long short-term memory networks for relation classification. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pages 73–78, 2015.