Sentence-level Senteiment Classification with RNN

Background

In this homework, we focus on fine-grained sentence-level sentiment classification problem. After implementing the details about MLP and CNN in python with Numpy and TensorFlow framework, you have learned basic skills in deep learning. In this homework, you can apply all the skills in sentence-level sentiment classification problem to improve the performance of your network. However, at the beginning, you should implement some **#todo** parts in codes, including basic rnncells in `cell.py`, loading word vectors in `main.py`, some networks in `model.py`, and TensorBoard in `main.py`.

We suggest you implement this project as follows,

1. Implement the basic framework in `main.py` and `model.py`;
2. Implement the rnncells in `cell.py` with basic TensorFlow api (using or copying high-level rnncell api is not permitted);
3. Implement TensorBoard in `main.py` with any statistics you need;
4. Implement any networks, such as other rnncells, multilayer networks, dropout layer, recursive networks, CNNs, different optimizers and meta-parameters, to improve the accuracy in testset. (10%~20% bonus)

Method	Fine-grained
SVM [Pang and Lee 2008]	40.7
MNB [Wang and Manning 2012]	41.0
bi-MNB [Wang and Manning 2012]	41.9
RNN [Socher et al. 2011b]	43.2
RNTN [Socher et al. 2013b]	45.7
MV-RNN [Socher et al. 2012]	44.4
AdaMC-RNN [Dong et al. 2014]	45.8
AdaMC-RNTN [Dong et al. 2014]	46.7
DRNN [Irsoy and Cardie 2014]	49.8

CNN [Kim 2014]	48.0
DCNN [Kalchbrenner et al. 2014]	48.5
LSTM [Tai et al. 2015]	46.4(1.1)
Bi-directional LSTM [Tai et al. 2015]	49.1(1.0)
Tree-LSTM [Tai et al. 2015]	51.0(0.5)
TW-LSTM (ours)	49.9(0.4)
TW-LSTM+p (ours)	50.6(0.4)
TE-LSTM (ours)	50.3(0.2)
TE-LSTM+p (ours)	51.3(0.4)
TW-LSTM+c (ours)	52.0(0.4)
TW-LSTM+c,p (ours)	52.1(0.4)
TE-LSTM+c (ours)	52.3(0.4)
TE-LSTM+c,p (ours)	52.6(0.6)

Requirements

Python2 or Python3, TensorFlow >= 1.0.0

Dataset Description

Stanford Sentiment Treebank (SST) dataset contains 11,855 sentences, and has splited into the training / validation / test parts, respectively containing 8,544 / 1,101 / 2,210 sentences.

Python Files Description

In this homework, we provide unfinished implementation of RNN in Tensorflow framework.

`main.py`: the main script for running the whole program.

`model.py`: main script for model implementation, including some utility functions.

`cell.py`: basic rnncells.

Report

In the experiment report, you need to answer the following basic questions:

1. Plot the loss value and accuracy value of one-layer RNN with 3 rnncells against to every epoch during training using TensorBoard.

2. Compare and analyze the performance of the 3 rnncells.
3. Plot the loss value and accuracy value of your final networks against to every epoch during training using TensorBoard.
4. Describe and analyze the details and the performance of your final networks.

Submission Guideline

You need to submit both report and codes, which are:

- **report**: well formatted and readable summary including your results, discussions and ideas. Source codes should not be included in report writing. Only some essential lines of codes are permitted for explaining complicated thoughts.
- **codes**: organized source code files with README for extra modifications or specific usage. Ensure that others can successfully reproduce your results following your instructions. **DO NOT include model weights/raw data/compiled objects/unrelated stuff over 50MB (due to the limit of XueTang)**

Deadline: Nov. 20th

TA contact info: Hao Zhou (周昊)，zhh032002@qq.com