# Sentence-level Sentiment 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, we believe that you have already learned basic skills in deep learning. In this homework, you should apply recurrent neural network (RNN) to sentence-level sentiment classification problem and implement one of the self-attention skills to improve the performance of your network.

We have learned from the lecture that RNN can encode the input sequence $X = \{x_1, x_2, \ldots, x_n\}$ into hidden states $H = \{h_1, h_2, \ldots, h_n\}$ with the size $n \times u$ , where $u$ is the dimension of each hidden state. In self-attention mechanism [1], we transform the "vector" hidden states $H$ into "matrix" hidden states $M$ :

$$A = softmax(W_{s2}tanh(W_{s1}H^T))$$
$$M = AH$$

where $W_{s1}$ is a $d_a \times u$ weight matrix and $W_{s2}$ is a $r \times d_a$ matrix. $A$ indicates $r$ different attention weight vectors with the size $1 \times n$ and the sum of each weight vector is 1 (because each weight vector is normalized by softmax function). $M$ denotes the matrix hidden states. In addition, we add a penalization term to the loss function to avoid redundant weight vectors :

$$P = ||(AA^T - I)||_F^2$$

where $||\cdot||_F$ stands for the Frobenius norm of a matrix and $I$ indicates the identity matrix. For more details, please refer to [1].

In this homework, you should implement #todo parts in codes, including basic rnncells in rnn_cell.py, rnn framework in model.py, and self-attention mechanism in model.py. We encourage you to rewrite part of the codes in main.py to use TensorBoard to visualize your experimental results.

## Requirements

- python >=3.6
- TensorFlow >= 1.11.0, <=1.14.0

## Dataset Description

Stanford Sentiment Treebank (SST) dataset contains 11,855 sentences, and has been split into the training / validation / test parts, respectively containing 8,544 / 1,101 / 2,210 sentences. We also provide pre-trained word embeddings in vector.txt and load them with $build\_vocab$ function in main.py.

## Python Files Description

- `main.py` contains the main script to run the whole program.
- `model.py` contains the main script for model implementation.
- `rnn_cell.py` contains various basic rnncells.

## Command

- `Train`: `python main.py`
- `Test`: `python main.py --is_train=False --inference_version=xxx`

**NOTE:** Feel free to fine-tune all the hyperparameters to get better results. You can choose the checkpoint with the best performance on validation set as inference_version. After you run the test command, you'll get $result.txt$ and **you should submit this file with your codes and report**.

## Report

Everyone needs to perform the following experiments in this homework:

1. Plot the loss value and accuracy value of one-layer RNN with 3 kinds of rnncells (i.e., BasicRNNCell, GRUCell, BasicLSTMCell) against to every epoch during training (on both training parts and validation parts). Compare and analyze the performance of 3 kinds of rnncells.
2. Compare the performance between your model with self-attention mechanism and without self-attention mechanism. Plot the loss value and accuracy value of the network against to every epoch during training (on both training parts and validation parts).
3. Plot the loss value and accuracy value of your final network against to every epoch during training (on both training parts and validation parts).
4. Describe and analyze the details and the performance of your final network.

**NOTE:** TensorBoard may be helpful when you plot figures in your experiment.

**NOTE:** The accuracy of your final network should be better than 42% or some penalty will imposed on your score.

## Submission Guideline

You need to submit a report document and codes, as required as follows:

- **Report:** well formatted and readable summary to describe the results, discussions and your analysis. Source codes should *not* be included in the report. Only some essential lines of codes are permitted. The format of a good report can be referred to a top-conference paper.
- **Codes:** organized source code files of your final network with README for extra modifications or specific usage. Ensure that TAs can easily 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)**.

You should submit a `.zip` file named after your student number, organized as below:

- `Report.pdf`
- `codes/`
  - `*.py`
  - `result.txt`
  - `README.md`

## Deadline

**November 5th**

**TA contact:** 黄斐, huangfei382@163.com

## Reference

[1] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In Proceedings of International Conference on Learning Representations.