Building Interactive Intelligent Systems
Assignment 3

The Hong Kong University of Science and Technology, Spring 2019

Deadline: Sun 23:59, 14 April 2019

1 Introduction

The goal of this assignment is to implement a Amazon product review rating classifier(given a review document of a user, you need to predict the overall rating given by the user), which is a similar task as your previous sentiment analysis task, but more complicated. To solve this more complicated problem, we will try to use more complicated models than logistic regression. In this assignment, we are going to implement a word-based Convolutional Neural Network (CNN) using PyTorch. This model follows the CNN architecture we introduced in class. For more detail, you can refer to https://arxiv.org/pdf/1408.5882.pdf. Through this exercise, you will be able to learn another way to train word embeddings (through classification) other than language modeling (as you might have noticed during class, word2vec is very similar to neural language modeling). For this assignment, we are going to use a Amazon reviews Dataset. To start, you can clone the project from https://github.com/hkusthltc/BIIS-hw3. In the repository, dataset and some sample code are provided, but you are encouraged to write the code from scratch.

2 Data

This dataset contains Electrical product reviews and metadata from Amazon. For details you can check http://jmcauley.ucsd.edu/data/amazon/. We extracted a small dataset, and splited it into *train.csv*, *dev.csv*, *test.csv*. To make the task easier, we customize 5-way classification task to 3-way classification task by mapping the rating from $1-5$ to $0-2$

2.1 Data Statistics(1 point)

As you have done for Assignment 1, you should first report several statistics regarding the data, including, but not exclusively,

- Number of sentences
- Number of words
- Number of unique words (vocabulary size) w/ and w/o minimum frequencies (e.g. 3)
- Top 10 most frequent words
- Maximum sentence length (Maximum number of words)
- Average sentence length (Average number of words)
- Sentence length variation (Standard Deviation in number of words)
- Distribution of classes

2.2 Preprocessing(0.5 point)

Just as you should have done in Assignment 1, try to clean the data as much as possible, while minimizing loss of information. Briefly list all the methods you used to clean the data in the report, and write your code inside function `clean`.

3 Implement ConvNet with PyTorch

3.1 WordCNN model(2 point)

Implement a Convolutional Neural Network attached with a Softmax layer and try to find the optimal kernel size, and kernel numbers (better not going deep at all, in this case). Submit this task as `model.py`.

```
import torch
import torch.nn as nn
class ConvNet(torch.nn.Module):
    def __init__(self, vocab_size, embedding_dim):
        super(ConvNet, self).__init__()
        # TODO

    def forward(self, inputs):
        # TODO
        return out
```

3.2 Using Word Embeddings (End-to-End)(1 points)

Instead of using feature vectors, create a randomly initialized word embedding matrix and train it together with the network, and report the results. This process is also called Embedding Lookup.

3.3 Hyperparameters(2 points)

For the better understanding of WordCNN, we want you to explore the following hyperparameters that affect your model performance.

- Learning Rate: [**0.1**, 0.01]
- Dropout probability: [0, 0.1, **0.3**, 0.5]
- Kernel number: [50, **100**, 150]
- Kernel sizes: [[2,3,4], **[3,4,5]**, [4,5,6]]
- Embedding dimension: [50, **100**, 200]

In addition to hyperparameters, we want you to explore the different pooling operations. To be more specific, compare and contrast **Max Pooling** and Average Pooling.

In this assignment, the default hyperparameters are highlighted(bold): Learning Rate: 0.1, Dropout probability: 0.3, Kernel number:100, Kernel sizes: [3,4,5], Embedding dimension:100, and Max Pooling. To compare different setting of hyperparameters, you only need to change one hyperparameter each time, and report the validation set accuracy by using different hyperparameter setting.

4 Results and Analysis

4.1 Evaluation: Accuracy

In this task, accuracy socre will be used to evaluate your results,You do not have to implement it from scratch; you can simply use `sklearn` library functions, such as `sklearn.metrics.accuracy_score`. `sklearn.metrics.classification_report` will also be useful to understand the performance of your model.

4.2 Submit your development set accuracy and test set predictions(2 point)

Report your best accuracy of development set, you are expected to achieve 50-70 accuracy in development set. And submit your test set predictions of test set(Submit this as `submission.csv`) by following the `predict` function in `main.py`.

4.3 Analysis(1.5 point)

Base on the result you get from 3.3, analyze the effect of following hyperparameters in your report:

- Effect of Kernel size
- Effect of number of Kernels
- Effect of dropout
- Effect of learning rate
- Max Pooling vs Average Pooling

5 Bonus(1.5 point)

Dynamic padding for each batch: In our stater code, we pad all input sequence to our predefined max length, but you can also try to do dynamic padding for each batch (pad input batch to the largest length of that batch) by customizing `collate_fn`.

Transfer Learning with Pre-trained Word Embeddings: Try to initialize with pre-trained word2vec, GloVe, FastText, or your Word Embeddings from HW2.

Other CNN Architectures commonly used in NLP: For example, try Character-level CNN, or combine it with Word-level as well.

Write down all details you have done in this part on your report, otherwise you might not get the bonus.

6 Submission

Turn this in by **Sun 23:59, 14 April 2019** by emailing your solution and code to hkust.hltc.courses@gmail.com, with subject line Building Interactive Intelligent Systems HW 3. Remember to state your name, student ID in the email. The zip should only include the following materials:

- Short report of the assignment in pdf format (maximum 3 pages).
- A zipped folder which contains: `preprocess.py`, `main.py`, `model.py` and any other files you have used in your experiments(except dataset). Document it well with comments.
- Your predictions on `test.csv` as `submission.csv`.