Building Interactive Intelligent Systems
Assignment 4

The Hong Kong University of Science and Technology, Spring 2019

Deadline: Sun 23:59, 28st April 2019

1 Introduction

The goal of this assignment is to perform Language Modeling based on Recurrent Neural Network model. You first need to load and clean and analyze the data. Furthermore, implement Recurrent Neural Network (RNN) Language Model model using PyTorch.

For this assignment, we are going to use a The Microsoft Research Sentence Completion Challenge Dataset from Sherlock Holmes novels. We already provide a subsets of the original challenge, however if your machine is not powerful enough you can further reduce the number of file. In case you do so, please report the files name used for the training.

2 Data

Please include the following subsections in your report, and implement your solution using `Dataset` and `DataLoader` of the pytorch library. Submit these as single file `dataloader.py`.

2.1 Data Statistics

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)
- Coverage of your limited vocabulary (UNK token rate)
- Top 10 most frequent words

Please submit your code for this task as `statistics.py`.

2.2 Preprocessing

If you look at the statistics from above, you can see that the original data is quite noisy. 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 submit your code as `preprocess.py`.

2.3 Train-Valid Split

We normally split the training dataset into Training and Validation, as you would have learned in previous assignments. Do not forget to handle this during this homework as well.

2.4 Dataloader output

The corpus is considered as a long continuous string of text, thus the output of the dataloader should be batch with a certaing window size. For instance, with the alphabet as the sequence batch size 4 and window size 6, we have the following:

a	b	c	d	e	f
g	h	i	j	k	l
m	n	o	p	q	r
s	t	u	v	z	x

Notice that each column is treated as independent sample by the model, which means that the dependence of e.g. 'g' on 'f' can not be learned, but allows more efficient batch processing. Indeed, in this way you do not need to implement any padding function.

3 Implement RNN with PyTorch

Implement a the Recurrent Neural Networks versions.

```
class RNN_LM(torch.nn.Module):
    def __init__():
        super(RNN_LM, self).__init__()
        emb = nn.Embedding(vocab_size, emb_size)
                # TODO
    def forward(self, inputs):
                # TODO
        return out
```

Notice, you should use nn.Embedding (Embedding Lookup), as in the previous assignments. Implement also the loss function (e.g. CrossEntropyLoss) and the optimizer (e.g. Adam). Submit this task as `RNN.py`.

4 Results and Analysis

Create a training function and a test function and submit a single file `main.py`. Language modeling task is evaluated using the perplexity score. Implement such score and implement an early stopping criteria using such score in the validation set. (Hint perplexity is simply e^{loss}, where $loss$ is the average loss value).

4.1 Hyper-parameters

Tune 5 combination of the following parameters. You are free to implement any regularization technique (e.g. Dropout or L2 reg.).

- Learning Rate: [0.01, 0.001, 0.0001]

- Hidden Dimension Size: [128, 256, 512]
- Number of Hidden Layers: [1, 2]

Report the validation perplexity and **discuss** the results achieved. Use the model with the best Hyper-parameters and report the test set perplexity.

4.2 Transfer Learning with Pre-trained Word Embeddings

Try to initialize nn.Embeddings with pre-trained word2vec, GloVe, FastText. Report and discuss whether this helps in this task.

4.3 Language Generation

RNN language model can be used as a text generator. This can be implemented by using the predicted word at time t as RNN input at time $t + 1$. Thus starting from a random token we can generate text which will result similar to the corpus. Try to implement such setting and generate a short paragraph starting from: 'I', 'What' and 'Anyway'.

5 Bonus

The above model is trained using word level information. However is possible to use char or subwords as input. Thus train a character level or a subword level language model using the same corpus and report the perplexity in this case.

6 Submission

Turn this in by **Sun 23:59, 28st April 2019** by emailing your solution and code to `hkust.hltc.courses@gmail.com`, with subject line Building Interactive Intelligent Systems HW 4. Remember to state your name, student ID in the email. The zip should only include the following materials:

- Short report (**MANDATORY**) of the assignment in pdf format (maximum 3 pages).
- A zipped folder which contains: `dataloader.py`, `statistics.py`, `preprocess.py`, `RNN.py`, `main.py`, and any other files you have used in your experiments. Document it well with comments.