COMP9444 Neural Networks and Deep Learning

Term 3, 2020

Project 2 - Rating and Category Prediction

Due: Thursday 19 November, 23:59 pm
Marks: 30% of final assessment

Introduction

For this assignment you will be writing a Pytorch program that learns to read business reviews in text format and predict a rating (positive or negative) associated with each review, as well as a business category (0=Restaurants, 1=Shopping, 2=Home Services, 3=Health & Medical, 4=Automotive).

Getting Started

Copy the archive `hw2.zip` into your own filespace and unzip it. This should create an `hw2` directory containing the main file `hw2main.py`, configuration file `config.py`, skeleton file `student.py` and data file `train.json` . Your task is to complete the file `student.py` in such a way that it can be run in conjunction with `hw2main.py` by typing

```
python3 hw2main.py
```

You must NOT modify `hw2main.py` in any way. You should only submit `student.py`
(If you wish, you can modify `config.py` in order to switch between CPU and GPU usage)

The provided file `hw2main.py` handles the following:

- Loading the data from `train.json`
- Splitting the data into training and validation sets (in the ratio specified by `trainValSplit`)
- Data Processing: strings are converted to lower case, and lengths of the reviews are calculated and added to the dataset (this allows for dynamic padding). You can optionally add your own tokenization, preprocessing, postprocessing and stop_words. (Note that none of this is necessarily required, but it is possible)
- Vectorization, using torchtext `GloVe` vectors 6B
- Batching, using the `BucketIterator()` prodived by torchtext so as to batch together reviews of similar length. This is not necessary for accuracy but will speed up training since the total sequence length can be reduced for some batches.

You should aim to keep your code backend-agnostic in the sense that it can run on either a CPU or GPU. This is generally achieved using the `.to(device)` function. If you do not have access to a GPU, you sohuld at least ensure that your code runs correctly on a CPU.

Please take some time to read through `hw2main.py` and understand what it does.

Constraints

We have tried to structure `hw2main.py` so as to allow as much flexibility as possible in the design of your `student.py` code. You are free to create additional variables, functions, classes, etc., so long as your code runs correctly with `hw2main.py` unmodified, and you are only using the approved packages (i.e. those available on the CSE machines). You must adhere to these constraints:

1. Your model must be defined in a class named `network`.
2. The `savedModel.pth` file you submit must be generated by the `student.py` file you submit.
3. Make sure you are using `PyTorch1.2` and `TorchText0.4` so that your `savedModel.pth` will have the correct format to run on the CSE machines.
4. Your submission (including `savedModel.pth`) must be under 50MB and you cannot load any external assets in the network class.
5. While you may train on a GPU, you must ensure your model is able to be evaluated on a CPU.

The `GloVe` vectors are stored in a subdirectory called `.vector_cache`. You are restricted to using `GloVe` vectors 6B, but you are free to specify the value of dim (50, 100, 200 or 300). If you are testing your trained model on the CSE machines, you can save disk space by removing the `.vector_cache` subdirectory and then creating a symbolic link by typing:

```
ln -s /home/cs9444/public_html/20T3/hw2/.vector_cache .vector_cache
```

You must ensure that we can load your code and test it. This will involve importing your `student.py` file, creating an instance of your `network` class, restoring the parameters from your `savedModel.pth`, loading our own test dataset, processing according to what you specified in your `student.py` file, and calculating accuracy and score.

You may NOT download or load data other than what we have provided. If we find your submitted model has been trained on external data you will receive zero marks for the assignment.

Question

At the top of your code, in a block of comments, you must provide a brief answer (one or two paragraphs) to this Question:

> Briefly describe how your program works, and explain any design and training decisions you made along the way.

Marking Scheme

After submissions have closed, your code wil be run on a holdout test set (i.e. a set of reviews and ratings that we do not make available to you, but which we will use to test your model). Marks will be allocated as follows:

- 12 marks for Algorithms, Style, Comments and Answer to the Question
- 18 marks based on performance on the (unseen) test set

The performance mark will be a function of the Weighted score, which is:

> (1.0 × Percentage with correct rating and correct category)
> + (0.5 × Percentage with correct category but incorrect rating)

+ (0.1 × Percentage with correct rating but incorrect category)

Groups

This assignment may be done individually, or in groups of two students. Groups are determined by an SMS field called hw2group. Every student has initially been assigned a unique hw2group which is "h" followed by their studentID number, e.g. `h1234567`. If you plan to complete the assignment individually, you don't need to do anything (but, if you do create a group with only you as a member, that's ok too). If you wish to form a group, go to the COMP9444 WebCMS page and click on "Groups" in the left hand column, then click "Create". Enter your Group Name and select the Group Type "hw2". After creating a Group, click "Edit", search for the other member, and click "Add". WebCMS assigns a unique group ID to each group, in the form of "g" followed by six digits (e.g. `g012345`). We will periodically run a script to load these values into SMS. You must ensure there are no more than two members in your group, and no-one is a member of two different groups.

Submission

You should submit your trained model and Python code by typing

```
give cs9444 hw2 student.py savedModel.pth
```

You must submit your trained model `savedModel.pth` as well as the Python code `student.py`

In order to avoid technical problems, you are strongly advised to submit from the command line, and not via the give Web interface. You can submit as many times as you like - later submissions by either group member will overwrite previous submissions by either group member. You can check that your submission has been received by using the following command:

```
9444 classrun -check
```

The submission deadline is Thursday 19 November, 23:59. 15% penalty will be applied to the (maximum) mark for every 24 hours late after the deadline.

Additional information may be found in the FAQ and will be considered as part of the specification for the project. You should check this page regularly.

When you submit, the system will check that your model can be successfully loaded, and evaluate it on data randomly chosen from a third dataset (disjoint from `train.json` and also disjoint from the holdout test set).

Common Questions:

- **Can I train on the full dataset if I find it?** No. You should NOT attempt to reconstruct the test set by searching the Internet. We will retrain a random selection of submissions, as well as those achieving high accuracy. If your code attempts to search or load external assets, or we find a mismatch between your submittied code and saved model, you will receive zero marks.

- **My model is only slightly larger than 50MB, can you still accept it?** No, the 50MB limit is part of the assignment specification and is quite generous. You should be able

to get away with much less.

- **Can we assume you will call `net.eval()` on our model prior to testing?** Yes.
- **Can we assume a max length on the reviews?** No. But nothing will be significantly longer than what is present in the training set.

General Advice:

- If you find your training accuracy is high, but the submission accuracy is low, you are overfitting to the training data.
- Try to be methodical in your development. Blindly modifying code, looking at the output, then modifying again can cause you go around in circles. A better approach is to keep a record of what you have tried, and what outcome you observed. Decide on a hypothesis you want to test, run an experiment and record the result. Then move on to the next idea.
- You should consider the submission test script to be the final arbiter with regard to whether a certain approach is valid. If you try something, and the submission test runs and you get a good accuracy then the approach is valid. If it causes errors then it is not valid.
- Do Not leave this assignment to the last minute. Get started early, and submit early in order to ensure your code runs correctly. Marks from automated testing are final. You should aim to be uploading your final submission at least two hours before the deadline. It is likely that close to the deadline, the wait time on submission test results will increase.

Plagiarism Policy

Your program must be entirely your own work. Plagiarism detection software will be used to compare all submissions pairwise and serious penalties will be applied, particularly in the case of repeat offences.

DO NOT COPY FROM OTHERS; DO NOT ALLOW ANYONE TO SEE YOUR CODE

Please refer to the UNSW Policy on Academic Integrity and Plagiarism if you require further clarification on this matter.

Good luck!
