Introduction to NLP (CS7.401)

Spring 2023, IIIT Hyderabad
Assignment 2

Dr. Manish Shrivastava

Deadline. 03rd March, 2023.

General Instructions

1. The assignment must be implemented in Python.
2. The submitted assignment must be your original work. Please do not copy any part from any source including your friends, seniors, and/or the internet. If any such attempt is caught, then serious actions including an F grade in the course is possible.
3. A single .zip file needs to be uploaded to the Moodle course portal.
4. Your grade will depend on the correctness of answers and output. In addition, due consideration will be given to the clarity and details of your answers and the legibility and structure of your code.
5. Please start early since no extension to the announced deadline would be possible.

Neural POS Tagging

Design, implement and train a neural sequence model (RNN, LSTM, GRU, etc.) of your choice to (tokenize and) tag a given sentence with the correct part-of-speech tags. For example, given the input

```
Mary had a little lamb
```

your model should output

```
Mary    NOUN
had VERB
a   DET
little  ADJ
lamb    NOUN
```

Note that the part-of-speech tag is separated from each word by a tab `\t` character.

Tune for optimal hyperparameters (embedding size, hidden size, number of layers, learning rate, complexity of decoding network) and report accuracy, precision, recall and F1-score of your trained model (refer to **this function**).

Analyse the results (both the scores as well as the optimal hyperparameters).

Dataset

Use the Universal Dependencies dataset, downloadable **here**. We recommend the files located at `ud-treebanks-v2.11/UD_English-Atis/en_atis-ud-{train,dev,test}.conllu`. Use the first, second and fourth columns *only* (word index, lowercase word, and POS tag).

The UD dataset *does not* include punctuation. You may filter the input sentence to remove punctuation before tagging it.

Note that many languages' data are downloadable from this resource. We expect a model trained on the English data at least, but you are free to train on other languages in addition.

Submission Format

Zip the following into one file and submit in the Moodle course portal. Filename should be `<roll number>_<assignment2>.zip`; *e.g.*, `2022xxxxxx_assignment2.zip`.

- Source Code
  - `pos_tagger.py`: Runs the POS tagger (no command line arguments), which should ask for a sentence, and output its POS tags in the format shown above,
- Pretrained Model
  - A `.pt` file which saves your pretrained model
- Report (PDF)
  - The complete scores of your model(s)
  - Hyperparameters used to train the model(s)
  - Your analysis of the results
- README on how to execute the file, load the pretrained model, etc.

Note that there are (at least) four files in total.

Grading

Evaluation will be individual and will be based on your viva, report, and code review. During your evaluation, you will be expected to walk us through your code and explain your results. You will be graded based on correctness of your code, accuracy of your results and quality of the code.

- Neural POS Tagger: 70 marks
  - Model implementation: 40 marks
  - Hyperparameter tuning: 5 marks
  - Results and analysis: 25 marks
- Quality and Efficiency: 10 marks
- Viva during Evaluation: 20 marks

Resources

1. **POS Tagging**
2. **RNNs and LSTMs**

FAQ

1. **Can I submit my code in Jupyter Notebook?** No, the final submission should be a Python script. You may work using Jupyter Notebooks, but make sure to convert them to .py files before submitting.
2. **Do I need to code everything from scratch?** No. You are free to use neural and LSTM layers from deep learning frameworks like PyTorch / Keras etc.
3. **My model takes too long for train/test. What should I do?** Ensure you are using a GPU environment with proper batch sizes for training. Look into optimisations that can be made in the forward function or elsewhere.
4. **My model size is greater than the limit allowed on moodle. What should I do?** Place the read-only link in the readme file after uploading the model to any cloud storage. Mention every specific step involved in running the model.
5. **Can I use libraries for automatic preprocessing?** Yes, you can use libraries like torchtext and NLTK for this (including word tokenization).