Assignment 2: Neural POS Tagging

Course: Introduction to Natural Language Processing

Deadline: 23 February | 23:59

General Instructions

1. The assignment must be implemented in Python.
2. The assignment must be done using PyTorch. Usage of other frameworks will not be accepted.
3. 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.
4. A single .zip file needs to be uploaded to the Moodle Course Portal.
5. 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.
6. Please start early since no extension to the announced deadline would be possible.

1 Feed Forward Neural Network POS Tagging

Design and implement a model which uses Feed Forward Neural Networks for POS Tagging. The Network should take the embeddings for the token, the p previous, and s successive tokens as input where p and s are fixed for a given network and output the POS tag.
For Example: In the sentence "An apple a day keeps the doctor away", to

get the POS tag for the word "a", a network with $p = 2$ and $s = 3$ would take the embeddings for ["An", "apple", "a", "day", "keeps", "the"] and output the POS tag "DET".

2 Recurrent Neural Network POS Tagging

Design and implement a model which uses Recurrent Neural Networks (Vanilla RNN, LSTM, or GRU) for POS Tagging. The model should take the embeddings for all tokens in a sentence and output the corresponding POS tags in sequence.
For Example: In the sentence "An apple a day keeps the doctor away", the model takes the embeddings for ["An", "apple", "a", "day", "keeps", "the", "doctor", "away"] and outputs the POS tags for all the words in the sentence ["DET", "NOUN", "DET", "NOUN", "VERB", "DET", "NOUN", "ADV"]

3 Hyperparameter Tuning

Try out different hyperparameters and evaluate to find the optimal configuration.

3.1 Feed Forward Neural Network POS Tagger

Experiment with different hyperparameter configurations like number of hidden layers, hidden layer sizes, embedding sizes, activation functions, etc. Report dev set evaluation metrics for three different configurations, and the test set evaluation metrics for the configuration performing best in the dev set. Report graphs for context_window $\in \{0...4\}$ vs dev set accuracy, where $p = s =$ context_window for one such configuration.

3.2 Recurrent Neural Network POS Tagger

Experiment with different hyperparameter configurations like number of stacks, bidirectionality, hidden state sizes, embedding sizes, activation functions, classification layer hyperparameters, etc. Report dev set evaluation metrics for three different configurations, and the test set evaluation metrics for the configuration performing best in the dev set. Report epoch vs dev set accuracy graphs for the three configurations being evaluated.

4 Evaluation Metrics

Use different metrics like accuracy, recall, and F1 scores (micro, macro) to evaluate your POS Tagger on the dev and the test set. Generate Confusion Matrices for the dev and test sets to better evaluate your taggers' tag based performance.

5 Analysis

Based on the observed metrics and graphs, draw conclusions from them and explain your results.

Dataset

Use the Universal Dependencies dataset, downloadable here. We recommend the files located at `ud-treebanks-v2.13/UD_English-Atis/en_atis-ud-{train,dev,test}.conllu`. Use the first, second, and fourth columns only (word index (starting from 1), lowercase word, and POS tag).

You may use the conllu library for parsing `.conllu` files. 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 files into one archive and submit it through the Moodle course portal. The filename should be `<roll number>_assignment2.zip`, for example, `2021114017_assignment2.zip`.

- **Source Code**
  - `pos_tagger.py`: Runs the POS tagger (command line arguments -f for FFN, -r for RNN), which should prompt for a sentence and output its POS tags in the specified format.
    ```
    python pos_tagger.py -f
    > An apple a day
    ```

```
an DET
apple NOUN
a DET
day NOUN
```

  - Code for generating graphs, and evaluating and training models.
  - You are encouraged to write modular code and include other intermediary files in your submission.

- **Pretrained Models**
  - `.pt` files containing your pretrained models or the drive link to them in your README.
- **Report (PDF)**
  - Hyperparameters used to train the model(s).
  - Corresponding graphs and evaluation metrics
  - Your analysis of the results.
- **README**
  - Instructions on how to execute the file, load the pretrained model, implementation assumptions etc.

Ensure that all necessary files are included in the zip archive. There should be, at least, four files in total.

Grading

Evaluation will be individual and 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 the correctness of your code, accuracy of your results, and the quality of the code.

Implementation: 50 marks

Hyperparameter Tuning Report: 20 marks

Analysis: 10 marks

Viva during Evaluation: 20 marks

Resources

1. Neural Networks and Deep Learning
2. How to Create a Neural Network (and Train it to Identify Doodles)
3. POS Tagging
4. RNNs and LSTMs
5. Understanding LSTM Networks
6. You can also refer to other resources, including lecture slides!