Assignment 1: Multilingual POS Tagging

CMU 11737, Fall 2020. Due Sept 21, 11:59pm

1 Overview

A number of tasks in natural language processing can be framed as sequence tagging, i.e. predicting a sequence of labels, one for each token in the sentence. In this homework, you will be looking at part of speech tagging for multiple languages, and investigating the challenges when the available labeled data is scarce. The goal of this homework is to make you familiar with deep learning frameworks (Pytorch), computing resources (AWS), and multilingual datasets.

2 Models for sequence tagging

For any given sequence of tokens, $\mathbf{x} = (x_1, \ldots, x_n)$, sequence tagging predicts a sequence of labels of the same length, $\mathbf{y} = (y_1 \ldots, y_n)$, where $y_i \in \{1, \ldots, L\}$, the labels of our interest. In discriminative models, we model any sequence of tags for input sequence with a scoring function $s(\mathbf{y}, \mathbf{x})$, such that the best prediction of the model corresponds to the following inference problem:

$$\hat{y} = \arg\max_{\mathbf{y}} s(\mathbf{y}, \mathbf{x}) = \arg\max_{\mathbf{y}} \sum_{i=1}^{n} \psi(y_i, i, x)$$

This classifier, when predicting the label y_i can use any of the features of the input sequence $\mathbf{x}$ and the position i. As a baseline in this homework, we will provide a bidirectional LSTM model to generate the features of the input sequence h_i on top of which we apply a feed-forward layer to predict the POS tag at every step i. (A more sophisticated model could incorporate the sequential nature of the labels, for example, conditional random fields (CRFs), although recent work like BERT has shown great results without explicitly incorporating any such information as well).

3 Data

You will evaluate your models on a subset of the treebank of Universal Dependencies (UD) v2.6. Universal Dependencies (UD) is a framework for consistent annotation of grammar (parts of speech, morphological features, and syntactic dependencies) across different human languages. In this task, you will only focus on parts of speech tags. You will evaluate your models on 8 languages (with a mix of high resource and low resource). We define languages with more than 60K tokens in their dataset as high-resource and others as low-resource, for this task. We will consider languages from the following language families.

- Germanic: English (en), Afrikaans (af)
- Slavic: Czech (cs)
- Romance: Spanish (es)
- Semitic: Arabic (ar)
- Baltic: Lithuanian (lt)
- Armenian: Armenian (hy)
- Dravidian: Tamil (ta)

4 Results

We provide a baseline implementation of a BiLSTM model along with the required data written in Pytorch. The provided link also contains a model trained on English UD data. Please refer to the comments under `assign1/main.py` to understand what each part of the code does.

To be able to run this code, you will need access to a machine with GPU(s). On this machine, you will need to install `python` ($\geq$ 3.6), `pytorch` ($\geq$ 1.0) and `torchtext` (0.7). We recommend you run your experiments in a conda environment which you can reuse for your future assignments and projects as well. The required libraries can then be installed by creating and activating a new conda environment and the following commands:

```
conda create -n 11737hw python=3.6
conda install pytorch torchtext cudatoolkit=10.1 -c pytorch
```

To run the code and reproduce the results:

1. Download the code and data from here.
2. Evaluate the model on the English UD test data.

   ```
   python main.py --mode eval --lang en
   ```

   This will load the provided model file trained on the English data and return the per-label accuracy on the test set.
3. Train a new model for each language in the provided list above.

   ```
   python main.py --mode train --lang lang-code
   ```

   Valid language codes are (en, es, cs, ar, af, lt, hy, ta)
4. Evaluate the newly trained models on their respective test sets.

Models are evaluated using per-label accuracy on the test set.

5 Requirements and Grading

You are expected to do the following:

1. Basic Requirement: Run the provided model on the provided English UD data and reproduce the results on its test set (expected accuracy: 91.66%). This will earn you a passing B grade

2. Multilingual Results: Reproduce the results on the other provided languages. This will earn you B+.

   Expected per-label accuracy:

en	cs	es	ar	af	lt	hy	ta
91.66	94.41	93.65	94.35	90.00	76.53	81.34	40.34

3. Report: Write and submit a report (max 3 pages) describing and analyzing the results. The report should be written in the ACL format. This will earn you an A-. Few examples of analyses:

   - How the performance changes across language families, typology, dataset size, etc.
   - Error analysis: Does the model perform better or worse on certain kinds of tags. What are the implications of this?
   - What happens when you change training set sizes?
   - What happens if you change the hyperparameters of the model (look at `config.json` in the codebase)?
   - What happens if the labels are noisy?

   Note that these are just examples and not an exhaustive list. You can choose to run any of these or do a different analysis of your own which you think might be interesting.

4. Make changes to the existing preprocessing/model/algorithm and show an improvement in the reported results. This will earn you an A, or A+ for particularly extensive or interesting improvements. Some suggestions for improvement:

   - Add a CRF layer on top of the baseline model.
   - Add a CNN input layer to capture character level features.
   - Pre-train the model parameters with a language modeling objective
   - Add multilingual pre-trained embeddings (Polyglot, mBERT) to your model. If you do this, make sure you also report at least one set of results without external resources such as this.
   - Auxiliary Losses

   For this part, you are allowed to make use of existing code with proper citation as long as it is incorporated in the given codebase without making drastic changes.

6 Submission

Your submission consists of two parts: Code and a writeup. Put all your code in a folder named `code` including instructions on how to run if you have implemented additional code, rename the writeup as `writeup.pdf` and compress both of them in a tar ball and name it `assign1.tar.gz`. This file must be submitted to Canvas (link on the course website) before Sept 21, 11.59pm ET.