Slot Tagging of Natural Language Utterances

**Due: Mon Nov 4, 2024 11:59pm**

The goal of this homework is to train a model to tag slots in natural language utterances of users addressing a virtual personal assistant. The tagged slots and the associated values are traditionally used to accomplish user requests. For example, if a user asks to learn about movies of a specific director, we would need to identify the name of the director (in addition to the user’s intention to find out about movies, as we did in homework 1), and issue a query to the backend knowledge graph to get information for formulating a response back to the user.

You will be using the same dataset as in homework 1, but your labels, this time, will be the slots for each token in the input utterance. For example:

show me movies directed by Woody Allen recently.

There are two slots in this utterance:

1. director = “Woody Allen”
2. release\_year = “recently”

The correct labels for this utterence would then be:

show me movies directed by Woody Allen recently.

O O O O O B\_director I\_director B\_release\_year

Note that the start of a slot has tags prefixed with B (e.g., B\_director) and the interior of slots has tags prefixed with I (e.g., I\_director).

The goal is to use PyTorch to develop and train your own deep neural network models and tag the sequential data using the training examples. You will have to try different techniques to improve upon a baseline model (which you need to define in your report). Here are some techniques you may try:

* Multilayer Perceptron (MLP)
* Recurrent networks (Elman RNN, LSTM, GRU)
* CNNs
* Different core features such as bag-of-words, word embeddings (random or pretrained)
* Additional information derived from the text
* Different loss functions
* Different optimizers such as SGD, AdaGrad, or Adam
* Preventing overfitting via dropout or weight-decay/regularization

To train a robust model, you will need to compare the performance of different models to identify their suitability. Test out key hyper-parameters and find what works best. Look at how similar problems are solved in research papers and try to implement those approaches. It may require some ingenuity in choosing features and developing/picking the right models.

Remember, each token in the input corresponds to one label in the output. Your model should be predicting exactly the right number of output labels, and if it isn’t, it may indicate a bug in your architecture.

Competition

The competition is hosted on [KaggleLinks to an external site.](https://www.kaggle.com/t/4305a2ddd58141ecad377d92b30acac6" \t "_blank).

You may use the training data to train a model. It’s recommended to use part of your training data as a validation set, but we’ve left that up to you. Using your model's predictions on the test set, generate a submission file in the format of sampleSubmission.csv. You can submit your predictions no more than 5 times every day before the competition deadline. However, there is no limit on the total number of submissions you can make. Kaggle will automatically evaluate your performance.

The leaderboard on Kaggle displays your model’s performance on the test set. The final results will be based on your highest scoring submission.

Dataset

This dataset is generated based on film schema of Freebase knowledge graph.

[Link to hw2 filesLinks to an external site.](https://drive.google.com/drive/folders/1pBXAY5qpKDbmdjDr4xUENPUpPlvVc9Xg?usp=sharing)

There are three CSV files.

hw2\_train.csv The file has three columns:

1. ID: the id for each row
2. UTTERANCE: the natural language text from you will extract relations
3. IOB Slot Tags: The targeted sequence of IOB tags corresponding to the utterance.

hw2\_test.csv

This file is similar to hw2\_train.csv, but only contains the ID and UTTERANCE columns. Your model will predict the IOB Slot Tags.

sampleSubmission.csv

This file is an example of what your submission file should look like. It contains only the ID and IOB Slot Tags columns.

You may submit your predictions no more than 5 times each day before the competition deadline. You will get an opportunity to review your scores in the output logs before making a submission to the leaderboard. We advise you to start making submissions as early and as frequently as possible!

Submission

You need to submit your predicted slot tags on Kaggle. The submission must be a CSV file named submission.csv. Additionally, you are also required to submit one report as well as your training and inference code on Canvas. The code package should include everything that is required to train and use your models at run-time. Please ensure you set up your code accordingly for us to be able to replicate your training and submissions — this involves setting a random seeds. You need to describe your models, approach, and parameters in your report.

Code Submission Details

**run.py**

Your submitted code package, when unzipped, must have a run.py file at the top level. run.py takes 3 command line arguments python run.py <train\_data> <test\_data> <output>

E.g. python run.py hw2\_train.csv hw2\_test.csv submission.csv

**requirements.txt**

Your submitted code package, when unzipped, must also have a requirements.txt file at the top level. This file specifies the dependencies of your code, and they will be installed with pip install -r requirements.txt

Your code package will be run by the following bash script. Make sure it can successfully run end to end.

#!/bin/bash  
  
{  
    unzip hw2.zip -d new\_dir  # unzip into a new directory  
  
    cd new\_dir  # go to the new directory  
  
    python -m venv venv  # create a virtual environment  
  
    source venv/bin/activate  # activate the virtual environment  
  
    pip install -r requirements.txt  # install dependencies  
  
    echo '========== start running =========='  # start running the main file  
    python run.py hw2\_train.csv hw2\_test.csv submission.csv  
  
} 2>&1 | tee record.txt  # record all outputs

Note

1. Your code will be run by python 3.11.

2. You don't need to include the training (hw2\_train.csv) and test (hw2\_test.csv) data in the zip file.

Evaluation

As before in Homework 1, your submissions will be evaluated using F1 score on the recognized entities. Since this is a sequence-to-sequence task, we will be using the [seqeval libraryLinks to an external site.](https://github.com/chakki-works/seqeval" \t "_blank) to calculate F1 score. I suggest you use the same in your code for validation (schema should be IOB2).

Homework Report

The homework report must be a detailed summary of the approach you followed to complete your work. We highly recommend that you use a [LaTeX templateLinks to an external site.](https://www.overleaf.com/latex/templates). for your report since for your proposal and final project, you will need to prepare those using the [ACL ProceedingsLinks to an external site.](https://www.overleaf.com/latex/templates/acl-2020-proceedings-template/zsrkcwjptpcd) format. You will be required to provide the following high level sections as part of your report with additional subsections as described:

**Introduction**

* Provide a formal statement of the problem you are trying to solve— whether it is a supervised or unsupervised problem, what specific task it is.
* Describe the dataset that was provided to you — background information, descriptive statistics of the dataset, what the input and output of the dataset are. Provide examples from the dataset inline or in tables.

**Models**

* Give a description of what embedding methods have used for the models that you are training and why you pick them.
* Include a subsection for each model that you are training. Give a brief summary of how the models are implemented, trained, and the tuneable hyperparameters of the model. Additionally, provide citations to the original work that implemented these methods.

**Experiments**

* Provide a description of the data-set split, the method for selecting hyper- parameters of your final models, any approaches you used for handling data sparsity/imbalance.
* Include a subsection for each model to describe the different values for the hyper-parameters tested, any special configurations for your model such as solvers or algorithms used.
* Describe the methods you used to evaluate how good your models were and what criterion you used to select the models for generating your test set submissions.

**Results**

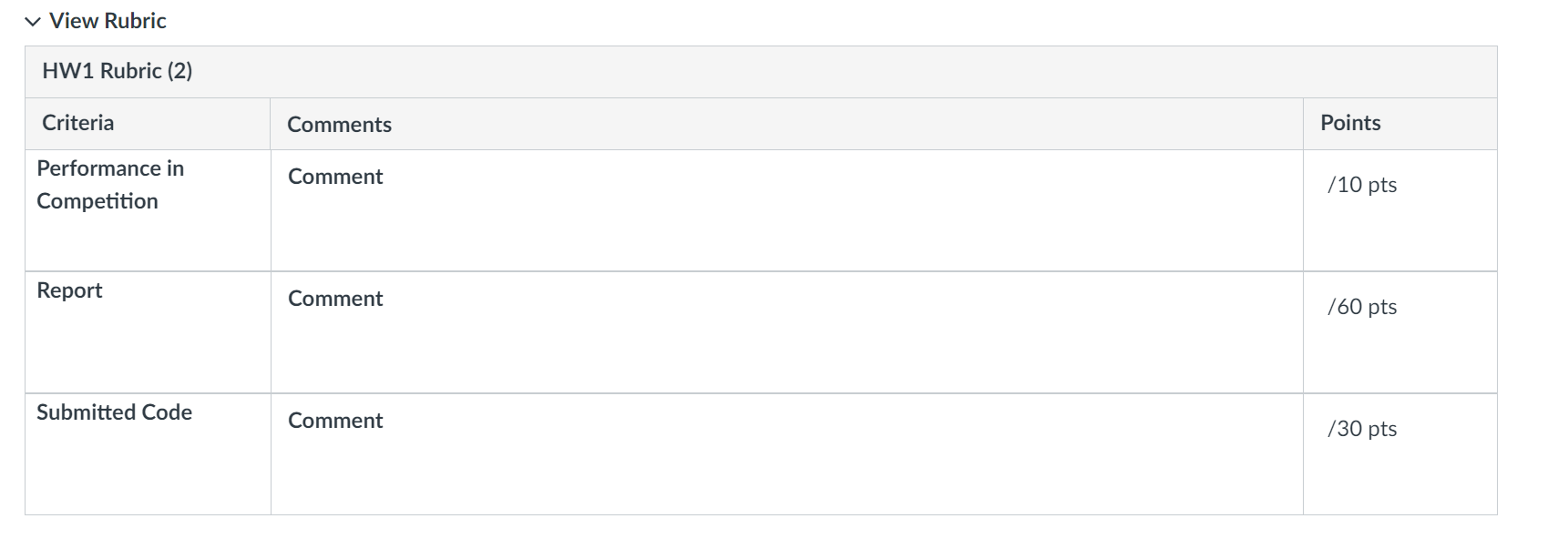
* Describe how well each model performed on your train, validation and test sets. Describe how the performance varied with different choice of hyperparameter values. Include the requisite tables, plots and figures to illustrate your point.
* Identify the best performing approach(es) and validate why they performed well. Try to bolster your conclusions by finding and citing work which arrive at similar results.

**References**

* Provide a bibliography for the literature that you cited. You can make use of bibtex or natbib packages to automatically generate the bibliography section.

**Appendix**

* Include an appendix for more detailed table, plots and figures, if needed.

****