Assignment 2

COMP7607: Natural Language Processing - University of Hong Kong

Fall 2022

In this assignment, you will explore how to use a language model to solve tagging problems. The tagging problem aims to predict the linguistic structures in natural language sentences. Generally, there are two typical problems in this direction, part-of-speech tagging and named entity recognition. In this assignment, we will focus on named entity recognition.

Submit: You should submit your assignment to the COMP7607 Moodle page. You will need to submit a PDF file **UniversityNumber.pdf** of your report and a zip file **UniversityNumber.zip**, which includes:

- `UniversityNumber.ipynb` with all log outputs **OR** source code with a README and all log files.
- UniversityNumber.lstm.test.txt and UniversityNumber.transformer.test.txt and (UniversityNumber.optional.test.txt if any)

Please make sure you include your log files containing your training and evaluation progress. Do not submit any data or model files.

1 Data

We will conduct experiments on Conll2003, which contains 14041 sentences, and each sentence is annotated with the corresponding named entity tags. You can download the dataset from the following link: `https://data.deepai.org/conll2003.zip`. We only focus on the **token and NER tags**, which are the **first and last columns** in the dataset. The dataset is in the IOB format, which is a common format for named entity recognition. The IOB format [1] is a simple way to represent the named entity tags. For example, the sentence "I went to New York City last week" is annotated as follows:

```
I O
went O
to O
New B-LOC
York I-LOC
City I-LOC
last O
week O
```

2 Tagger

You will train your tagger on the train set and evaluate it on the dev set. And then, you may tune the hyperparameters of your tagger to get the best performance on the dev set. Finally, you will evaluate your tagger on the test set to get the final performance.

[1] https://en.wikipedia.org/wiki/Inside–outside–beginning_(tagging)

There are some key points you should pay attention to:

- You will batch the sentences in the dataset to accelerate the training process. To batch the sentences, you may need to pad the sentences to the same length.
- You are free to design the model architecture with (Bi)LSTM or Transformer unit for each part, but please do not use any pretrained weights in your basic taggers.
- You will adjust the hyperparameters of your tagger to get the best performance on the dev set. The hyperparameters include the learning rate, batch size, the number of hidden units, the number of layers, the dropout rate, etc.
- You will use `seqeval` [2] to evaluate your tagger on the dev set and the test set.

2.1 LSTM Tagger

We will first use an LSTM tagger to solve the NER problem. There is a very simple implementation of the LSTM tagger on PyTorch website https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html. You can refer to this implementation to implement your LSTM tagger.

2.2 Transformer Tagger

We will also use Transformer to solve the NER problem. You can refer to the following link to implement your Transformer tagger: https://pytorch.org/tutorials/beginner/transformer_tutorial.html.

2.3 Results

Please report your best performance on the test set in the same format as the test.txt file, but replace the ground truth labels with your prediction.

Optional: After you finish the basic LSTM tagger and the Transformer tagger, you can try to improve the performance of your tagger with external resources. For example, you can use pre-trained word embeddings like GloVe [3] to initialize the word embeddings of your tagger. You can also use the pre-trained language model like BERT [4] to initialize the parameters of your tagger. Please describe your method in detail in your report and save your prediction in another file.

3 Report

You will write a report including the following parts:

- The description of your tagger, including the architecture, the hyperparameters, etc.
- The description of your settings, including the hyperparameters you have tried, the performance of your tagger on the dev set, etc.
- The **discussion** and **table** of your results, including the performance of your tagger on the test set, the analysis of the results, etc.

Optional: Visualizing the training process can help you analyze the training process and the performance of your tagger. You may use tensorboard [5] to visualize the training process of your tagger, including the loss curve, the accuracy curve, etc. You can refer to the following link for more details about tensorboard: https://pytorch.org/tutorials/intermediate/tensorboard_tutorial.html.

[2] https://github.com/chakki-works/seqeval
[3] https://nlp.stanford.edu/projects/glove/
[4] https://arxiv.org/abs/1810.04805
[5] https://github.com/tensorflow/tensorboard