CSE 556: Natural Language Processing
Assignment 2

Date: 21st Feb 2024
Due Date: 11:59:59 pm on 11th March 2024 Max Marks: 100

General Instructions:

- Every assignment has to be attempted by four people. At least one subtask has to be done by one team member. All members need to have a working understanding of the entire code
- Institute policies will apply in cases of plagiarism
- Create separate .ipynb files for each part. The file name should follow the format: "A2_<Part number>.ipynb"
- Create a single .ipynb file for generating the final outputs that are required for submittables. It should be named as "A2_<Group No>_infer.ipynb". Clearly indicate which cell corresponds to the output of which task/subtask. Outputs will be checked from this inference file only by TAs.
- Carefully read the deliverables section at the bottom of this document. Along with the code files, submit all the other files mentioned there, strictly following the naming convention instructed.
- Only one person has to submit the zip file containing all the above-mentioned files, saved models, and the report PDF. It will be named "A2_<Group No>.zip". The person having the alphabetically smallest name should submit.

Task 1:

This task corresponds to Legal Named Entities Recognition from the Indian court judgment text. Named entities refer to the key subjects (given as labels in the dataset) of a piece of text. Implement this task using the methods described below as subtasks (Part 2 and Part 3).

Example:
Text:
`True, our Constitution has no 'due process' clause or the VIII Amendment; but, in this branch of law, after R.C. Cooper v. Union of India, (1970) 1 SCC 248 and Maneka Gandhi v. Union of India, (1978) 1 SCC 248, the consequence is the same.`
Named entities:

1. Entity (Constitution) - label (STATUTE)
2. Entity (R.C. Cooper v. Union of India, (1970) 1 SCC 248) - label (PRECEDENT)
3. Entity (Maneka Gandhi v. Union of India, (1978) 1 SCC 248) - label (PRECEDENT)

Dataset 1: It can be downloaded from here. The folder consists of two JSON files as train and test data. The JSON files contain a list of dictionaries, each representing a judgment case. Inside the dictionaries, the judgment text is given inside the 'data' key, and the 'annotations' contain the named entities, their span, and corresponding labels present in that text segment. It is a subset of the OpenNyai legal-NER dataset available here. Refer to this paper to get insight into the dataset and explanations of different named entities. Of the 14 legal entities mentioned in the paper, 13 are available in this given subset of the dataset, so only those 13 will be used as labels for task evaluation.

Evaluation Metrics: Macro-F1, Accuracy

Task 2:

This task corresponds to Aspect Term Extraction from Laptop-review texts. Aspect extraction is the task of identifying and extracting terms relevant to opinion mining and sentiment analysis (e.g., terms for laptop features.Implement this task using the methods described below as subtasks (Part 2 and Part 3).

Example:
Text:
`One night I turned the freaking thing off after using it , the next day I turn it on , no GUI , screen all dark , power light steady , hard drive light steady and not flashing as it usually does .`
Aspect Terms:
“GUI”, “screen”, “power light”, “hard drive light”

Dataset 2: It can be downloaded from here. This dataset will be used for all the given subtasks. The folder contains three JSON files as train, validation and test data. The JSON files contain a list of dictionaries, and inside the dictionaries, the review text is given inside the ‘raw_words’ key, and the ‘aspects’ key contains all of the aspect terms of the corresponding text and their span indices.

Evaluation Metrics: Macro-F1, Accuracy

Part 1: Data Preparation [12 marks]

A) This part corresponds to the preparation of Dataset_1. (7 marks)

First, split the train data of Task_1 into training and validation sets with an 85:15 ratio (randomly stratified). **(1 mark)**
Implement a code for BIO (Beginning-Intermediate-Outside) chunking of the given dataset of Task_1 (for the three splits). Tokenization should be done based on space, and each token needs to be assigned a BIO label (in format B_label, I_label or O, where “label” refers to one of the 13 legal entities). Preprocessing can be done on the text if required. The output should have the following format described as following: **(5 marks)**

```
// it’s a nested dictionary where the outer dict has case_id as keys
// here ids refer to the unique case_id strings given in the dataset
// inner dict should have two keys named as follows:
        // “text”: judgement text of the case present as outer dict key
        // “labels”: list of BIO tags corresponding to each token of the judgement text
{
        “id1”:{
                “text”: "(See Principles of Statutory Interpretation by Justice G.P. Singh, 9th Edn.,
                2004 at p. \n\n 438.)."
                “labels”: [O, O, O, O, O, O, O, B_JUDGE, I_JUDGE, O, O, O, O, O, O, O]
        },
        …
}
```

This processed data should be used for all the tasks described below and saved separately in JSON files for train, validation, and test splits. **(1 mark)**

B) This part corresponds to the preparation of Dataset_2. (5 marks)

Write a code for BIO (Beginning-Intermediate-Outside) chunking of the given dataset for Tast_2 (for the three splits). Tokenization should be done based on space, and each token needs to be assigned a BIO label (in format B, I, O). The output should have the following format described as following: **(4 marks)**

```
// it's a nested dictionary where the outer dict has any integers as keys,
// inner dict should have two keys named as follows:
    // "text": text copied from the values of 'raw_words' in the dataset
    // "labels": list of BIO tags corresponding to each token of the text
    {
        "1":{
            "text": "This laptop meets every expectation and Windows 7 is great !"
            "labels": [O, O, O, O, O, O, B, I,  O, O, O] },
        …
        "1000":{
            "text": "I am using the external speaker -- sound is good."
            "labels": [O, O, O, O, B, I, O, B, O, O] }
    }
```

This processed data should be used for all the tasks described below and saved separately in JSON files for train, validation, and test splits. **(1 mark)**

Part 2: Baseline Models Implementation [45 marks]

- Implement RNN-based sequence tagging models using the above datasets (Part 1A and 1B) in the following setups:
    1. **Model 1:** Using vanilla RNN layer (may refer to Figure_1 of this paper)
    2. **Model 2:** Using LSTM network (may refer to Figure_3 of this paper)
    3. **Model 3:** Using GRU network

  In each of these setups, use three different pre-trained word embeddings: word2vec, GloVe, and fasttext in the embedding layer of the networks and train a total of nine models for each dataset in this subtask. You can also use one pre-trained contextualized initialised embedding (e.g., BERT, LegalBERT) instead of any one of the above mentioned word embedding models.
  You are free to use any optimiser, learning rate, activation functions, dropout, and additional linear layers. Macro-F1 should be used as the validation metric. All of the models should be saved for future inference. **(18 x 1 = 18 marks)**

- Generate the following plots for every model: **(18 x 1 = 18 marks)**
    - Loss Plot: Training Loss and Validation Loss V/s Epochs
    - F1 Plot: Training Macro-F1-score and Validation Macro-F1-score V/s Epochs
    - Analyze and Explain the plots obtained

- In the final output file, load the trained models, extract the named entities (for Dataset_1) or aspect terms (for Dataset_2) from the test data, and report their overall accuracy and macro-F1 scores using nine print statements for each dataset. The TAs should be able to run this independently and generate these scores during evaluation. **(18 x 0.5 = 9 marks)**

Part 3: BiLSTM-CRF Model Implementation [43 marks]

- Implement a BiLSTM-CRF model **(Model 4)** for token classification using the above dataset (formed in Part 1A). You need to implement the architecture given in Figure_7 of this paper for this model. The concepts of LSTM and CRF are given in this paper. You may refer to this implementation code of BiLSTM-CRF. Use three different pre-trained word embeddings: word2vec, GloVe, and fasttext in the embedding layer of the networks and train a total of three models for each dataset in this subtask. You can also use one pre-trained contextualized initialised embedding (e.g., BERT, LegalBERT) instead of any one of the above mentioned word embedding models. You are free to use any optimiser, learning rate, dropout and activation functions. Macro-F1 should be used as the validation metric. The model should be saved for future inference. **(6 x 3 = 18 marks)**

- Generate the following plots for every model: **(6 x 1.5 = 9 marks)**
    - Loss Plot: Training Loss and Validation Loss V/s Epochs
    - F1 Plot: Training Macro-F1-score and Validation Macro-F1-score V/s Epochs
    - Analyze and Explain the plots obtained

- In the final output file, load the trained models, extract the named entities (for Dataset_1) or aspect terms (for Dataset_2) from the test data, and report their overall accuracy and macro-F1 scores using three print statements for each dataset.
    - The TAs should be able to run this independently and generate these scores during evaluation. **(6 x 1.5 = 9 marks)**
    - Calculate the label-wise F1 scores (for 13 labels) on test data of Dataset_1 using the best-performing model. Plot a bar graph or pie chart for this using code. **(2 marks)**
    - The test evaluation scores of variants of Model_4 should surpass that of all variants of Model_1, Model_2, and Model_3. **(5 marks)**

Deliverables:

Along with the code files, the following needs to be included in the zip.

- 3 JSON files for preprocessed dataset representing train, validation and test split generated in the subtask Part_1A. Name them as 'NER_train.json', 'NER_val.json' and 'NER_test.json'.
- 3 JSON files for preprocessed dataset representing train, validation and test split generated in the subtask Part_1B. Name them as 'ATE_train.json', 'ATE_val.json' and 'ATE_test.json'.
- 12 saved models for task_1 with the name "t1_<model>_<embedding>.pkl" (for Keras) or "t1_<model>_<embedding>.pt" (for PyTorch).
  (For example, 't1_model1_word2vec.pt' as the first saved model of Part2)
- 12 saved models for task_2 with the name "t2_<model>_<embedding>.pkl" (for Keras) or "t2_<model>_<embedding>.pt" (for PyTorch).
  (For example, 't2_model4_fasttext.pt' as the last saved model of Part3)
- The report pdf must contain the following:
    - Snippet of one or two data samples prepared in Part 1A and Part 1B. Describe if any additional preprocessing is performed on the text.
    - All 48 plots generated in the subtasks, along with their analysis.
    - Two tables containing the performance of all the 12 models on test data of Dataset_1 and Dataset_2, respectively, in this format:
      <Model_No, Embedding_used, Accuracy, Macro_F1>
    - Plot of the label-wise F1 scores (13 labels) on test data of Dataset_1 using best model.
    - Contribution of each team member to this assignment