ASSIGNMENT 2: NAMED ENTITY RECOGNITION FOR REAL ESTATE TEXT

Motivation: The motivation of this assignment is to get practice with sequence labeling tasks such as Named Entity Recognition.

Scenario: Different real estate agents share noisy text messages on a real estate platform to inform buyers about new properties available for sale. We will call these text messages, *shouts*. As a company interested in automating real estate information, our first step is to perform NER on these shouts so that important and relevant information can be extracted downstream.

Problem Statement: The goal of the assignment is to build an NER system for shouts. The input of the code will be a set of tokenized shouts and the output will be a label for each token in the sentence. Labels will be from 8 classes:

- Locality (L)
- Total Price (P)
- Land Area (LA)
- Cost per land area (C)
- Contact name (N)
- Contact telephone (T)
- Attributes of the property (A)
- Other (O)

Labelled Training Data: We are sharing a training and a dev dataset of shouts, named `train.txt' and `dev.txt' respectively.. In train/dev file, each line contains one token of a sentence, followed by a space and its token label. Sentences are separated by the blank lines. There are 3466 labeled sentences (examples) in `train.txt' and 312 ones in `dev.txt'.

Unlabelled Training Data: Along with train-dev splits, we are releasing a supplementary data file named `unlabelled.json' containing 25000+ unannotated instances of short advertisements in jsonlines format, each line containing one advertisement. You may use this unlabelled corpus for fine-tuning BERT, word2vec embeddings etc. This may help fine-tune BERT or other pre-trained embeddings for the given domain and may potentially boost your performance on dev/test dataset. Again, this is optional and completely upto you whether you want to leverage this dataset for performance improvement or not. You may not like to use it at all.

The Task: You need to write a sequence tagger that labels the given shouts in a tokenized test file. The tokenized test file follows the same format as training data except that it contains only a sentence token in each line immediately followed by newline character `\n' (i.e. no token labels in test file will be given at inference time). You should label the test file in the same format as the training data. The format of your output file will be the label (of string type) for each corresponding token of the test file in one line

immediately followed by `\n' (do not output the token itself in output file but only the labels). So, the output will have same no. of lines as text file with matching blanks marking the end of sentence.

1. You may like to create features for each token, e.g. whether token is capitalized or not, whether it's a number or not etc. You may also try features from lower level syntactic processing like POS tagging or shallow chunking. You may need to use Twitter-trained chunkers/taggers. Resources: Twitter NLP at Noah's Ark, and Twitter NLP at Alan Ritter.
2. Define task-specific features such as specific regular expressions indicative of specific types.
3. Use existing gazetteers of locations or bootstrap one. We promise not to test you on locations outside the Delhi NCR area.
4. You may define word shape features or word substring features.
5. You can use any off-the-shelf tool/code for feature engineering (but not for making the tagging model itself).
6. You may try BiLSTM (with which you are familiar) and BiLSTM-CRF as well (read about this yourself). You are free to use BERT-based taggers etc. But again, make sure you do not copy the codebase from anywhere but write your own codes.

Submission Format:

The submission deadline for the assignment is 16th Jan 2021. There are no late submissions in this assignment.

We will follow a similar format as in A1.3, with changes in output format etc. for this problem. Please read below:

1. Submit a zip file on moodle wih name <kerberos_id.zip> (E.g. `csz198394.zip'). Unzipping this should generate a directory with name <kerberos_id> (E.g. `csz198394') having 3 files – `install_requirements.sh', `run_model.sh' and `writeup.txt'.
2. The command for training is - ` bash run_model.sh train <train_file_path> <val_file_path>' . This should generate a tagging model named `<kerberos_id>_model' (E.g. `csz198394_model') along with possibly other files needed for inference.
3. Command for inference - `bash run_model.sh test <test_file_path_1> outputfile.txt' . This should generate `outputfile.txt' having predicted label in string format (`L', `P' etc.) followed by `\n' in each line for the token in the corresponding line of test file. The `test_file_path_1' will only have an unlabelled token followed by `\n' in each line.
4. Command for evaluation - `python compute_accuracy.py <test_file_path_2> outputfile.txt' . We will release `compute_accuracy.py' shortly on Moodle. You must use that for validation purposes. The `test_file_path_2' will have a token followed by space followed by it's label followed by `\n' in each line.

We restate the complete instructions for training and inference as follows:

 # Running the model in inference

- unzip <kerboros_id.zip>
- cd <kerberos_id>
- conda create -n <kerboros_id> python=3.7
- conda activate <kerboros_id>
- bash install_requirements.sh

- cp -R /scratch/cse/phd/csz198394/A2/<kerboros_id>/* .
- bash run_model.sh test <test_file_path_1> outputfile.txt
- python compute_accuracy.py <test_file_path_2> outputfile.txt

 # Training the model

- bash run_model.sh train <train_file_path> <val_file_path>
- bash run_model.sh test <test_file_path_1> outputfile.txt

Evaluation Criteria

1. **20 points for performance of your code for each NER (including Other). A total of 160 points**. The metric used for evaluating each NER will be word-level F-score.
2. Bonus points awarded for outstanding performers

What is allowed? What is not?

1. The assignment is to be done individually.
2. You should use Python 3.7 and PyTorch for this assignment.
3. You must not discuss this assignment with anyone outside the class. **Make sure you mention the names in your write-up in case you discuss with anyone from within the class.** Please read academic integrity guidelines on the course home page and follow them carefully.
4. Feel free to search the Web for papers or other websites describing how to build named entity recognizers. Cite the references in your writeup.
5. We will run plagiarism detection software. Any team found guilty will be awarded a suitable penalty as per IIT rules.
6. Your code will be automatically evaluated. You will get a significant penalty if it does not conform to output guidelines.

Disclaimer: The dataset and problem is brought to you courtesy Plabro Networks, a former Delhi-based startup. So, this assignment gives you a taste of the real, real world. Kindly refrain from sharing the dataset outside the class.