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. More precisely you will experiment with the CRF model and various features on social media text with a natural language processing package called MALLET.

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)

Training Data: We are sharing an unlabeled dataset of shouts. The first line is the whole shout. Following it each line is one token, followed by a space and its token label. Blank lines indicate the end of a sentence. Each student is being given 100 shouts to label in Part I. This will create a dataset of 4500+ labeled shouts, which will be given to everyone as training data in Part II.

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 except that it does not have the final label in the input. Your output should label the test file in the same format as the training data.

To accomplish this, first download and install MALLET. Get Familiar with the "sequence tagging" part of MALLET by reading about command line interface for sequence tagging and the developer's guide for sequence tagging. This documentation is very short and incomplete, but that's all there is.

Run Mallet's SimpleTagger on training.txt with the following command:

- java -cp "/home/username/mallet-2.0.7/class:/home/username/mallet-2.0.7/lib/mallet-deps.jar" cc.mallet.fst.SimpleTagger --train true --test lab --threads 2 training.txt

The above command is meant for linux so you may need to adapt the syntax for other operating systems. Also, make sure to correct the path. Mallet seems to be buggy if you use a single thread so make sure to set the --threads option to at least 2. Mallet will optimize a CRF model on half of the data and test it on the other half. If MALLET takes too long, increase the number of threads based on the number of cores that your computer has. Also, use the --iterations option to reduce the number of iterations from the 500 default to something smaller like 50.

In order to improve the tagging accuracy create additional features that might be useful for the task. For example, if you wish to add features of capitalization and whether the current token is a number, it may look like this for the phrase “sector 35 Gurgaon”:

sector L

35 NUMBER L

Gurgaon CAPITAL L

…

You may have multiple features space separated before the final label of the token. Insert only the features that are on before the label. The word itself is treated as a feature. The order of the features does not matter.

Here are some suggestions on features:

1. 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 can promise not to test you on locations outside Delhi NCR area.
4. We are working on providing a larger unlabeled corpus of shouts. You can train a word2vec model and use embeddings as features (if we are successful).
5. You may define word shape features or word substring features.
6. Your idea here…

You may also experiment with the order of the Markov Chain in CRF model by using the --orders option. If you are dissatisfied with Mallet, you are welcome to write all your code (or part of your code) from scratch. But, please do not use any other existing codebase without permission.

What to submit?

1. Submit your tagging for Part I by Monday March 28th. Submit in a text file titled tagging-Entryno.txt. Late submission is not allowed for this.

2. Submit your best code (best if trained on all training data and not just on a subset) by Monday, 18th April 2016, 11:55 PM. The code should **not** need to train again. You should submit only the testing code, after the models have been trained. That is, you should not need to access the training data anymore.
   Submit your code is in a .zip file named in the format **<EntryNo>.zip.** Make sure that when we run "unzip yourfile.zip" the following files are produced in the present working directory:
   compile.sh
   run.sh
   Writeup.pdf (and not writeup.pdf, Writeup.doc, etc)

   You will be penalized if your submission does not conform to this requirement.

   Your code will be run as ./run.sh inputfile.txt outputfile.txt. The outputfile.txt should have the same number of lines as inputfile.txt. And it should have two additional characters per token line (space and labeling). Here is a format checker. Make sure your code passes format checker before final submission.

   Your code should work on our Baadal servers with 2GB memory. You will be penalized for any submissions that do not conform to this requirement.

3. Your writeup (at most 1 page, 10 pt font) should describe how you created your best NER system. Focus on any innovations in the system.

Evaluation Criteria

(1) 20 points are for completion of tagging for Part I.
(2) 12.5 points for performance of your code for each NER (including Other). A total of 100.
(3) Bonus points awarded for outstanding performers

What is allowed? What is not?

1. The assignment is to be done individually.
2. You may use Java, or Python 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 get significant penalty if it is does not conform to output guidelines. Make sure it satisfies the format checker before you submit.