University of Macau

CISC3025 – Natural Language Processing

Project#2, 2020/2021

(Due date: **4th May**)

Person Name ('Named Entity') Recognition

This is a group project with **two** students at most. You need to enroll a group here. In this project, you will be building a maximum entropy model (MEM) for identifying person names in newswire texts (Label=PERSON or Label=O). We have provided all of the machinery for training and testing your MEM, but we have left the feature set woefully inadequate. Your job is to modify the code for generating features so that it produces a much more sensible, complete, and higher-performing set of features.

NOTE: In this project, we expect you to design a web application for demonstrating your final model. You need to design a web page that provides at least such a simple function: 1) User inputs sentence; 2) Output the named entity recognition results. Of course, more functionalities in your web application are highly encouraged. For example, you can integrate the previous project's work, i.e., text classification, in your project (It would be very cool!).

You NEED to submit:

- **Runnable program**
  - You need to implement a Named Entity Recognition model based on the given starter codes
- **Model file**
  - Once you have finished the designing of your features and made it functions well, it will dump a model file ('`model.pkl`') automatically. We will use it to evaluate your model.
- **Web application**
  - You also need to develop a web application (freestyle, no restriction on programming languages) to demonstrate your NER model or even more NLP functions.
  - Obviously, you need to learn how to call your python project when building the web application.
- **Report**
  - You should finish a report to introduce your work on this project. Your report should contain the following content:
    - Introduction;
    - Description of the methods, implementation, and additional consideration to optimize your model;
    - Evaluations and discussions about your findings;

    - Conclusion and future work suggestions.

- **Presentation**
    - You need to give a **10-minute** presentation in the class to introduce your work followed by a **3-minute** Q&A section. The content of the presentation may refer to the report.

Starter Code

In the starter code, we have provided you with three simple starter features, but you should be able to improve substantially on them. We recommend experimenting with orthographic information, gazetteers, and the surrounding words, and we also encourage you to think beyond these suggestions.

The file you will be modifying is MEM.py

```
def features(self, words, previous_label, position):
        features = {}
        """ Baseline Features """
        current_word = words[position]
        features['has_(%s)' % current_word] = 1
        features['prev_label'] = previous_label
        if current_word[0].isupper():
            features['Titlecase'] = 1
        #===== TODO: Add your features here =======#
        #...
        #=============== TODO: Done ===============#
        return features
```

Adding Features to the Code

You will create the features for the word at the given position, with the given previous label. You may condition on any word in the sequence (and its relative position), not just the current word because they are all observed. You may not condition on any labels other than the previous one.

You need to give a unique name for each feature. The system will use this unique name in training to set the weight for that feature. At the testing time, the system will use the name of this feature and its weight to make a classification decision.

Types of features to include

Your features should not just be the words themselves. The features can represent any property of the word, context, or additional knowledge.

For example, the case of a word is a good predictor for a person's name, so you might want to add a feature to capture whether a given word was lowercase, Titlecase, CamelCase, ALLCAP, etc.

Imagine you saw the word “Jenny”. In addition to the feature for the word itself (as above), you could add a feature to indicate it was in Title case, like:

```
if current_word[0].isupper():
        features['Titlecase'] = 1
```

You might encounter an unknown word in the test set, but if you know it begins with a capital letter then this might be evidence that helps with the correct prediction.

Choosing the correct features is an important part of natural language processing. It is as much art as science: some trial and error is inevitable, but you should see your accuracy increasing as you add new types of features.

The name of a feature is not different from an ID number. You can use assign any name for a feature as long as it is unique. For example, you can use “case=Title” instead of “Titlecase”.

Running the Program

We have provided you with a training set and a development set. We will be running your programs on an unseen test set, so you should try to make your features as general as possible. Your goal should be to increase F1 on the dev set, which is the harmonic mean of the precision and the recall. You can use three different command flags (‘-t’, ‘-d’, ‘-s’) to train, test, and show respectively. These flags can be used independently or jointly. If you run the program as it is, you should see the following training process:

```
$ cd NER
$ python run.py -t
Training classifier...
  ==> Training (5 iterations)

      Iteration    Log-Likelihood    Accuracy
      ---------------------------------------
             1          -0.69315        0.055
             2          -0.09383        0.946
             3          -0.08134        0.968
             4          -0.07136        0.969
         Final          -0.06330        0.969
```

Afterward, it can print out your score on the dev set.

```
$ python run.py  -d
Testing classifier...
f_score =       0.8715
accuracy =      0.9641
recall =        0.7143
precision =     0.9642
```

You can also give it an additional flag, -s, and have it show verbose sample results. The first column is the word, the last two columns are your program's prediction of the word's probability to be PERSON or O. The star '*' indicates the gold result. This should help you do error analysis and properly target your features.

```
$ python run.py  -s
  Words          P(PERSON)  P(O)
----------------------------------------
  EU              0.0544  *0.9456
  rejects         0.0286  *0.9714
  German          0.0544  *0.9456
  call            0.0286  *0.9714
  to              0.0284  *0.9716
  boycott         0.0286  *0.9714
  British         0.0544  *0.9456
  lamb            0.0286  *0.9714
  .               0.0281  *0.9719
  Peter          *0.4059   0.5941
  Blackburn      *0.5057   0.4943
  BRUSSELS        0.4977  *0.5023
  1996-08-22      0.0286  *0.9714
  The             0.0544  *0.9456
  European        0.0544  *0.9456
  Commission      0.0544  *0.9456
  said            0.0258  *0.9742
  on              0.0283  *0.9717
  Thursday        0.0544  *0.9456
  it              0.0286  *0.9714
```

1. Function 'features()' in MEM.py
2. You can modify the "Customization" part in `run.py` in order to debug more efficiently and properly. It should be noted that your final submitted model should be trained under at least 20 iterations.

```
#====== Customization ======
BETA = 0.5
MAX_ITER = 5     # max training iteration
BOUND = (0, 20)  # the desired position bound of samples
#===========================
```

3. You may need to add a function "predict_sentence( )" in class MEM( ) to output predictions and integrate with your web applications.

Changes beyond these, if you choose to make any, should be done with caution.

Grading

The assignment will be graded based on your codes, reports, and most importantly final presentation.

Tips

- Start early. This project may take longer than the previous assignments if you're aiming for the perfect score.
- Generalize your features. For example, if you're adding the above "case=Title" feature, think about whether there is any pattern that is not captured by the feature. Would the "case=Title" feature capture "O'Gorman"?
- When you add a new feature, think about whether it would have a positive or negative weight for PERSON and O tags (these are the only tags for this assignment).