COMP9318 (16S2) PROJECT 1

DUE DATE: 23:59 11 OCT 2016 (TUE)

1. Objective: Named Entity Recognition

In this project, you will use `scikit-learn` and `python 3` to engineer an effective classifier for an important Information Extraction task called *Named Entity Recognition*.

2. Getting Start

2.1. **Named Entity Recogonition.** The goal of Named Entity Recognition (NER) is to locate segments of text from input document and classify them in one of the pre-defined categories (e.g., `PERSON`, `LOCATION`, and `ORGNIZATION`). In this project, you only need to perform NER for a single category of `TITLE`. We define a `TITLE` as an appellation associated with a person by virtue of occupation, office, birth, or as an honorific. For example, in the following sentence, both `Prime Minister` and `MP` are `TITLE`s.

Prime Minister Malcolm Turnbull MP visited UNSW yesterday.

2.2. **Formulating NER as a Classification Problem.** We can cast the `TITLE` NER problem as a binary classification problem as follows: for each token w in the input document, we construct a feature vector $\vec{x}$, and classify it into two classes: `TITLE` or `O` (meaning *other*), using the logistic regression classifier.

The goal of the project is to achieve the best classfication accuracy by configuring your classifier and performing feature engineering.

In this project, you must use the `LogisticRegression` classifier in `scikit-learn ver 0.17.1` in the implementation.

3. Your Tasks

3.1. **Training and Testing Dataset.** We will publish a training dataset on the project page to help you build the classifier. The training dataset contains a set of sentences, where each sentence is already tokenized into *tokens*. We also provide the part-of-speech tags (e.g., , and the correct named entity class (i.e., "`TITLE`" or "`O`") for each token[1]. The data structure in `python` for the above sentence is:

[1]The POS tag is generated by `nltk.pos_tag()`. The set of POS tags is listed at `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`.

```
[[('Prime', 'NNP', 'TITLE'), ('Minister', 'NNP', 'TITLE'),
('Malcolm', 'NNP', 'O'), ('Turnbull', 'NNP', 'O'),
('MP', 'NNP', 'TITLE'),
 ('visited', 'VBD', 'O'), ('UNSW', 'NNP', 'O'),
('yesterday', 'NN', 'O'), ('.', '.', 'O')]]
```

Therefore, the training dataset is formed as a list of sentences. Each sentence is formed as a list of 3-tuple, which contains to the token itself, its POS tag, and the named entity class.

In python, you could just use the following code to get the training dataset (where `training_data` is the path of the provided training dataset file).

```
with open(training_data, 'rb') as f:
    training_set = pickle.load(f)
```

The final test dataset (which will **not** be provided to you) will be similarly formed as the training dataset, except each sentence will be formed as a list of 2-tuple (only the token itself and its PoS tag). For example, if the testing dataset only contains one sentence which is "*Prime Minister Malcolm Turnbull MP visited UNSW yesterday.*", then it will be format as:

```
[[('Prime', 'NNP'), ('Minister', 'NNP'), ('Malcolm', 'NNP'),
('Turnbull', 'NNP'), ('MP', 'NNP'), ('visit', 'NN'), ('UNSW', 'NNP'),
('yesterday', 'NN'), ('.', '.')]]
```

3.2. **Feature Engineering.** In order to build the classifier, you need to firstly extract features for each token. In this project, using token itself as a feature usually achieves a high accuracy on the training dataset, however it could result in low testing accuracy on the test dataset due to *overfitting*. For example, it is not uncommon that the test dataset contains titles that do not exist in the training dataset. Therefore, we encourage you to find/engineer meanningful/strong features and build a more robust classifier.

You will need to describe all the features that you have used in this project, and justify your choice in your report.

3.3. **Build logistic regression classifier.** In this project, you need to use the logistic regression classifier (i.e., `sklearn.linear_model.LogisticRegression`) from the scikit-learn package. For more details, you could refer to the scikit-learn page[2] and the relevant course materials.

You also need to dump the trained classifier using the following code (where `classifier` is the trained classifier and and `classifier_path` is the path of the output file):

```
with open(classifier_path, 'wb') as f:
    pickle.dump(classifier, f)
```

You are also required to submit the dumped classifier (which must be named as `classifier.dat`).

[2] `http://scikit-learn.org/stable/index.html`

Suggestion Steps. You may want to implement a basic, initial NER classifier. Then you can further improve its performance in multiple ways. For example, you can find best setting of hyper-parameters of the your model/features; you can design and test different sets of features. It is recommended that you use cross validation and construct your own testing datasets.

You need to describe how you have improved the performance of your classifier in your report.

3.4. **Trainer.** You need to submit a python file named `trainer.py`. It receieves two command line arguments which specify the path to the training dataset and the path to a file where your trained classifier will be dumped to. Your program will be excuted as below:

```
python trainer.py <path_to_training_data> <path_to_classifier>
```

3.5. **Tester.** You need to submit a python file named `tester.py`. It recieves three command line arguments which specify the path to the testing dataset, the path to the dumped classifier, and the path to a file where your output results will be dumped to. Your program will be executed as below:

```
python tester.py <path_to_testing_data> <path_to_classifier> <path_to_results>
```

You should, for each token in the testing dataset, output the named entity class of it (i.e., `TITLE` or `O`). Your output, in python internally, is a list sentences, where each sentence is a list of `(TOKEN, CLASS)` tuples. For example, a *possible* output for the example in Section 3.1 is:

```
[[('Prime', 'TITLE'), ('Minister', 'TITLE'), ('Malcolm', 'O'),
('Turnbull', 'O'), ('MP', 'O'), ('visit', 'O'), ('UNSW', 'O'),
('yesterday', 'O'), ('.', 'O')]]
```

Then you should dump it to a file using the following code (where `result` is your result list and `path_to_results` is the path of the dumped file):

```
with open(path_to_results, 'wb') as f:
    pickle.dump(result, f)
```

3.6. **Report.** You need to submit a report (named `report.pdf`) which answers the following two questions:

- What feature do you use in your classifier? Why are they important and what information do you expect them to capture?
- How do you experiment and improve your classifier?

4. Execution

Your program will be tested automatically on a CSE Linux machine as follows:

- the following command will be used to build the classifier:

  `python trainer.py <path_to_training_data> <path_to_classifier>`

  where

  - `path_to_training_data` indicates the path to the training dataset
  - `path_to_classifier` indicates the path to the dumped classifier
- the following command will be used to test the classifier:

```
python tester.py <path_to_testing_data> <path_to_classifier> <path_to_results>
```

  where
  - `path_to_testing_data` indicates the path to the testing dataset
  - `path_to_classifier` indicates the path to the dumped classifier
  - `path_to_result` indicates the path to the dumped result

Your program will be executed using python 3 with only the following packages available:

- nltk 3.2.1
- numpy 1.11.1
- scipy 0.18.0
- scikit-learn 0.17.1
- pandas 0.18.1

5. Evaluation

We will use F_1-measure to evaluate the performance of your classifier. We will test your classifier using two test datasets,

- the first one is sampled from the **same** domain as the given training dataset, and
- the second one is sampled from a domain **different** from that of the given training dataset.

Each test will contribute 40 points to your final score. Your report contributes the rest 20 points.

In order to minimize the effect of randomness, we will execute `trainer.py` three times, and use the best performance achieved.

For each test dataset, your best performed classifier will be compared with a reference classifier C. While the detailed scheme will be published later on the project website, basically, the marking scheme will reward you with more points if your classifier works no worse than the reference classifier.

Neither the reference classifier C nor the two test datasets will be given to you. But you can create your own test dataset, submit it and get the performance of the reference classifier C on it. For more details, please refer to Section 6.

5.1. Bouns.

For the *second* test dataset (i.e., the one sampled from a *different* domain), Your classifier will be entered into a "contest" with all the classifiers from other students in the class.

For the top-20 classifiers, you will get *bouns points*. More specifically, the 1st place will get 20 bonus points, the 2nd place will get 19 points, and so on and so forth.

6. Customized Datasets

You are encouraged to construct your own test datasets to test your classifier and help improve its performance. You can submit your datasets to get the F_1-measure of the pre-trained classifier C on your datasets. Furthermore, all the submitted datasets is accessible by every student in the class. You should also benefit from such experience because this will give you many initial ideas about the meaningful features to use in your own classifier.

You can upload your test cases to through the data submission website (url will be sent via email). You will need to login before uploading your datasets or downloading other datasets. The id is your student number (e.g., $z1234567$) and your password will be sent to you by email.

Once you have logged into the system, you can:

- submit your dataset
- view the performance of all the datasets
- download a chosen dataset

Your submitted test datasets should be in the same format as the training dataset.[3] You can submit **at most 10 datasets** to the system. We will release more details about how to use the system and some tools/tips to help you visualize and tag `TITLE` occurrences later on the project website.

7. Submission

Please archive and compress your project files as `proj1.tar.gz`. Note that all the files **must** be in the same directory (i.e., no sub-directory allowed).[4]

Your submission should contain at least the following three files:

- `trainer.py`,
- `tester.py`,
- and `report.pdf`

Your file may include other files (e.g., source file or a file that stores all the features) as long as the `proj1.tar.gz` file is within 5MB.

You can submit your file by

```
give cs9318 proj1 proj1.tar.gz
```

Late Panelty. -10% per day for the first two days, and -20% per day for the following days.

[3] You can use `pickle.dump()` to dump the list to a file.

[4] Use the command line `tar cvfz proj1.tar.gz trainer.py tester.py report.pdf OTHERFILES`