Natural Language Processing Final: Day 2

Date: 23rd Dec 2014

Time: 8:30 am to 1 pm IST

Problem # 1: Develop a Relationship Extractor for the specified relations

In this project you will build a relationship extractor using a combination of supervised learning techniques and rule based models. At a high level, the steps involved in building this module using the supervised learning are:

1. Choosing the relations that we need to extract
2. Choosing the named entities that we need for our application
3. Build a labelled corpus
4. Train a classifier
5. Test and fine tune

For a subset of relations you are also required to implement rule based models as specified below.

You are required to perform the following:

1. You will need to support the following relations:
    a. Price Query
    b. Feature Query
    c. Comparison
    d. Interest_intent
    e. Irrelevant
    f. Disagreement
    g. Greeting
    h. Agreement
    i. Acknowledgement
2. Use the supervised learning techniques to extract relations for relations 1(a) to 1 (e) as above. Implement the remaining relations using a rule based approach.

    Note: Though it is possible to use machine learning to tag all the relations, we have chosen to also use rule based models due to 2 reasons:

    - The dataset created by us is not generated with a dialogue/chat process. Hence we will not have adequate data to build an effective machine learning system.

   - There may be situations where a hybrid approach (ie, rule based plus machine learning) may produce better performance compared to pure machine learning techniques.

3. Use the tool http://jnresearch.com/edit_rer_corpus to tag relations in a sentence. Please refer the document from Shruti for examples. Please note that a given sentence may contain more than 1 relation. If the sentence doesn't have any relation that we can process tag it as "irrelevant" relation. When you tag these keep in mind how these will be used in the downstream processes and use the judgment accordingly. Please note that the accuracy of your relationship extractor heavily depends upon the quality of the labelled data. The tool groups sequence of words that have same tag in to one single tag. For example: I/Other, need/Other, a/Other, phone/Phone is tagged as: (I need a)/Other, phone/Phone.
4. Once the steps above is completed, go to url: http://jnresearch.com/show You will see a json formatted list of records. You are required to do the following:
    a. Copy this data in to a text file using a text editor
    b. Embed this data (list) as: {"root": data_received_from_url}
    c. Save this as a JSON file: all_data.json
5. The file, all_data.json contains the Mobile Corpus in the format below (example):

```
{"root":[
          {"data": [
             { "rels": [{"interest_intent": ["Phone", "Other", "Price"]}],
             {"updates": [
               ],
              "sentence": "…"
             },
             {"updates": […], "sentence": "… "}
             …
             ]
             "short_name": "Meghana",
             "user_id": "1PI11CS096"
             },
             …
             ]}
```

The number of elements under root key is the number of people who contributed to the corpus. For 75 students and 4 faculty this should be 79 (assuming everyone contributed). Each of these elements is a dictionary having 3 keys: data, user_id and short_name. The element data is a list of dictionaries, each dictionary element having keys: **rels**, updates, sentence. The rels key maps to a list of values where each element is a dictionary of the form: {“relation_type”: [entity_types]}. The key updates is mapped to a list of elements of the form: {“tag”:…, “word”: …}.

6. You are required to use the MaxEnt classifier to extract the relations.
7. Features are at the heart of the MaxEnt classification process. For each of the relation to be supported you are required to identify one or more features.
8. There are about 1600 sentences in the corpus. For this exercise we will select 1200 sentences for training and another 400 sentences for testing.
9. You can use the reference implementation for MaxEnt that we provided.
10. Run the classifier classify() method with the test samples and record the result
11. Evaluate the accuracy of the implementation. You need to calculate precision, recall and F1 score. Precision refers to the ratio of number of relations correctly classified to the total number of relations of that type. Recall refers to the number of relations of a given type discovered divided by the total number of relations of that type.
12. Once you are done with the development upload your files using the upload web service provided in the ner_client.py module.

NOTE

You should implement all these using the files as below and upload them for evaluation. Please make the file names all lowercase letters as in Linux file names are case sensitive.

1. mymaxent.py : Implements MyMaxEnt class
2. rer_feature_functions.py : Implements all feature functions as FeatureFunctions class for relationship extraction
3. rules.py: Implements the rule based model for the specified relations
4. rer_main.py : the main program that:
    a. Prepares the input data for MaxEnt
    b. Trains the MaxEnt using train() method
    c. Evaluates the performance and reports
5. After training the classifier create a pickle file with the name: rer.p

Upload all the above files using ner_client.upload()

<u>Deliverables</u>:

1. Source code of all your py modules (use ner_client.upload())
2. Pickle file of your classifier (use ner_client.upload())

Best wishes from your faculty ☺