AHLT Laboratory Session NER.2

NER Feature Extraction

In this session we will build a basic machine learning based system for the first task in the DDI challenge.

We will split the required tasks in three parts:

- Process XML and convert text data (train and test) into feature vectors.
- Train a model using the feature vectors from the training data
- Apply the model to the test (or development) data to predict entities
- Evaluate results using the official scorer.

Extracting Features

The program `extract-features.py` will process all documents in the given directory and create a file with a feature vector for each word.

The features for a sequence (a whole sentence) are extracted by the function `extract_features`:

```
def extract_features(tokens) :

     #  for each token, generate list of features and add it to the
result
   result = []
   for k in range(0,len(tokens)):
      tokenFeatures = [];
      t = tokens[k][0]

      tokenFeatures.append("form="+t)
      tokenFeatures.append("formlower="+t.lower())
      tokenFeatures.append("suf3="+t[-3:])
      tokenFeatures.append("suf4="+t[-4:])
      if (t.isupper()) : tokenFeatures.append("isUpper")
      if (t.istitle()) : tokenFeatures.append("isTitle")
      if (t.isdigit()) : tokenFeatures.append("isDigit")

      if k>0 :
         tPrev = tokens[k-1][0]
         tokenFeatures.append("formPrev="+tPrev)
         tokenFeatures.append("formlowerPrev="+tPrev.lower())
         tokenFeatures.append("suf3Prev="+tPrev[-3:])
         tokenFeatures.append("suf4Prev="+tPrev[-4:])
         if (t.isupper()) : tokenFeatures.append("isUpperPrev")
         if (t.istitle()) : tokenFeatures.append("isTitlePrev")
         if (t.isdigit()) : tokenFeatures.append("isDigitPrev")
      else :
         tokenFeatures.append("BoS")

      if k<len(tokens)-1 :
         tNext = tokens[k+1][0]
         tokenFeatures.append("formNext="+tNext)
         tokenFeatures.append("formlowerNext="+tNext.lower())
         tokenFeatures.append("suf3Next="+tNext[-3:])
         tokenFeatures.append("suf4Next="+tNext[-4:])
         if (t.isupper()) : tokenFeatures.append("isUpperNext")
         if (t.istitle()) : tokenFeatures.append("isTitleNext")
         if (t.isdigit()) : tokenFeatures.append("isDigitNext")
      else:
         tokenFeatures.append("EoS")

      result.append(tokenFeatures)

   return result
```

The output of `extract-feature.py` is a file with all the sentences in the input corpus, each line encodes information about a token. Sentences are separated by blank lines.

For instance,

```
python3 extract-features.py data/Train/DrugBank > train.features
python3 extract-features.py data/Test-NER/DrugBank > test.features
```

Each token line in train.features has the following format. Fields are separated by TABs

```
SentenceID tokenForm startOffset endOffset class feature1 feature2 ...
```

Learning a model

The program `train-crf.py` will read a feature-encoded dataset and learn a CRF model.

The program will use only fields `class` and `feature`$_k$.

For instance:

```
python3 train-crf.py mymodel.crf < train.features
```

Using a model to make predictions

The program `predict-crf.py` will read a feature-encoded dataset and apply the given model to predict entities.

The program will ignore field `class`. It will use the `feature`$_k$ fields to predict a class, and the other fields (`SentenceID,tokenForm,startOffset,endOffset`) to format the predictions as requested by the task official scorer.

For instance:

```
python3 predict-crf.py mymodel.crf <test.features >task9.1_CRF1_01.txt
```

Evaluating predictions

The file produced by `predict-crf.py` can be input to the official task scorer.

Make sure to name the file with a name matching the pattern required by the scorer: `task9.1_XXXX_NNN.txt`, where the string `XXXX` and the number `NNN` identify your experiment.

For instance:

```
java -jar eval/evaluateNER.jar data/Test-NER/DrugBank task9.1_CRF1_01.txt
```

Session tasks

The goals of this session are:

- Understand how the provided programs work and interact.
- Learn about CRFsuite and how to call it from python
- Apply ML experimental methodology to improve the results of the provided prototype

Specifically, the tasks to perform are:

- Create a proper experimental environment:
  - Split the Training data set in training/development (or use k-fold cross validation if you prefer to). **Do NOT** use the Test section to optimize the parameters
  - Create programs/scripts that allow you to easily encode the corpus using different feature sets
  - Create programs/scripts that allow you to run the training/testing/evaluation cycle with different parameters (feature sets, feature frequency filters, regularization coefficients, etc.)
- Once you have this organized, you can perform experiments to find out the best performing parameters. You can also experiment using separate training for DrugBank and MedLine or to fuse them in a mixed dataset.

- It is specially important to extend the used feature set, which is now very simple. The extension should use a NLP tool (FreeLing, TextServer, NLTK, etc) to enrich the text with linguistic information. Some possible features to be added are:
  - Part-of-speech of the tokens (requires calling a PoS tagger)
  - Lemma of the tokens (requires calling a PoS tagger/lemmatizer)
  - Features indicating whether a token (or a n-gram of tokens) belongs to a list of known drugs [of a certain class] (requires external lists to be consulted)
  - Features indicating whether the tokens contain digits, dashes, greek letters, …

You can use this prototype as a starting point for your lab project.

You can replace the CRF learner/predictor with any other ML method(s) of your choice (SVMs, ANN, …) .

If you stick to CRFs, you will be expected to obtain competitive results, thus you’ll need to work harder on the feature extraction and parameter exploration.