Conditional Random Fields for Sequential Tagging

1 Objectives and Material

The goal of this lab session is to implement linear-chain Conditional Random Fields (CRF) for the extraction of named entities in texts. You will first have to learn how to use an existing CRF software and apply it for a specific text tagging task, then to implement a part of the CRF inference engine in python.

The resources needed for this work can be downloaded from :
http://www.telecom-paristech.fr/~essid/teach/tp_crf.zip

If you use your own computer, for this lab, you will need the anaconda distribution of python and crf-suite. To install the latter with pip :

```
pip install python-crfsuite
```

1.1 Named Entity Recognition task

"Named Entity Recognition (NER) is a subtask of Information Extraction. The goal is to find the phrases that contain person, location and organization names, times and quantities. Each word is tagged with the type of the name as well as its position in the name phrase (i.e. whether it is the first item of the phrase or not) in order to represent the boundary information [1]."

1.2 Dataset

The dataset to be tagged is a Spanish-language corpus which was provided for the Special Session of CoNLL2002 on NER [Tjong Kim Sang, 2002]. The data is a collection of news wire articles and is labelled for person, organization, location and miscellaneous names. Thus, the micro label set consists of 9 labels, that are :

- the beginning and continuation of Person (B-PER, I-PER), Organization (B-ORG, I-ORG), Location (B-LOC, I-LOC), and Miscellaneous names (B-MISC, I-MISC)
- nonname tags (O).

By definition, the continuation of a name type has to be preceded by the beginning or the continuation of the same type. The training data consists of 7230 sentences of average length 36. Each word is tagged in Part of Speech (POS) : noun, verb, adjective, etc. POS are considered here as a part of the observations, for both NER and chunking (see last section).

The data can be obtained via the nltk package. You may need to download the data :

```
import nltk
nltk.download()
```

1. from *Altun, Yasemin. "Discriminative methods for label sequence learning." (2005).*

2 python-crfsuite

2.1 Getting started

You will use `python-crfsuite`, a python binding to crfsuite (originally written in C++). The first steps to take consist in :

1. getting familiar with the generation of features under CRF suite by reading the doc available here : http://www.chokkan.org/software/crfsuite/tutorial.html#id485365
2. studying the following python notebook presenting a tutorial on `python-crfsuite` applied to NER : http://nbviewer.ipython.org/github/tpeng/python-crfsuite/blob/master/examples/CoNLL%202002.ipynb

To work with the notebook file `CoNLL-2002.ipynb`, save it to your working directory and, in a terminal, run the following commands :

```
cd /cal/softs/anaconda/anaconda-2.0.1/bin
./ipython notebook CoNLL-2002.ipynb
```

Note that it is critical to run `ipython` from the above specified location, in order to use the right configuration for this lab session.

Observe the transitions and their probability and the features that have been used.

2.2 Experiments

Download the code of the notebook and start customising the script. We recommend using `Spyder` :

```
cd /cal/softs/anaconda/anaconda-2.0.1/bin
./spyder &
```

Study the behaviour of the estimated model and the results when considering different types of regularizations (use only ℓ_1, only ℓ_2 or both) and changing the values of the regularization parameters. Make sure to use l-bfgs for solving the optimisation problem.

What is the impact of the regularization parameter on the results ?

3 Coding your own CRF inference routines in python

For convenience, it is suggested that you dump your test dataset so you can easily reuse it for what follows. To do so, you can use :

```
import cPickle as pickle
pickle.dump({'X': X_test, 'y': y_test},
              open(CRFSUITE_TEST_DATA_FILE, 'wb'))
```

Now, study the structure of `tp_crf.py` and the routines defined in `flexcrf_tp.models.linear_chain.py`. Then,

— code the `viterbi_decoder()` function and compare the results obtained with those output by `pycrfsuite` ;

— add the code needed to compute the posterior probability of the decoded sequences, using routines defined in `flexcrf` ; compare to `crfsuite` results.

4 Going further

Try to improve the tagging results by designing new features.

Create your own features in order to build a chunker on the CoNLL 2000 data, relying on the information provided on http://www.chokkan.org/software/crfsuite/tutorial.html.

5 Documentation : Python and NLTK - Natural Language processing Toolkit

To start with python :

⋆⋆⋆ http://perso.telecom-paristech.fr/~gramfort/liesse_python/1-Intro-Python.html

⋆⋆⋆ http://perso.telecom-paristech.fr/~gramfort/liesse_python/2-Numpy.html

⋆⋆⋆ http://perso.telecom-paristech.fr/~gramfort/liesse_python/3-Scipy.html

⋆⋆⋆ http://scikit-learn.org/stable/index.html

⋆⋆ http://www.loria.fr/~rougier/teaching/matplotlib/matplotlib.html

⋆⋆ http://jrjohansson.github.io/

NLTK documentation : http://www.nltk.org/