# CS 419/519 Document Analysis
# Lab and Assignment 2: Information Extraction

October 15, 2015

This lab and assignment involves training a conditional random field (CRF) based named entity recognizer (NER), experimenting with various features and model configurations, and running the best NER system on automatically extracted key content from news websites (for which you have to write a very simple XML/XPath-based key content extractor).

- Maximum marks: 10
- Programing language: Java (only)
- Assignment questions: CS419/519 Canvas Site
- Deadline: Fri Oct 23 @ 2pm (scheduling a meeting 3-5pm mandatory for discussing answers)

**Marking scheme and requirements:** Full marks will be given for (1) working, readable, reasonably efficient, documented code that achieves the assignment goals and (2) for providing appropriate answers to Q1-Q4 in a file `ANSWERS_lastname.pdf` submitted with your solution.

Please adhere to the collaboration policy on the course website – people you discussed the assignment solution with, or websites with source code you used should be listed in `ANSWERS_lastname.pdf`.

To verify that your code runs end-to-end, you may be asked to make some changes to your search engine during the evaluation lab and then re-evaluate results.

**What/how to submit your work:** Please submit `ANSWERS_lastname.pdf` and all your source code, zipped into a single file called `Assignment2_lastname.zip` and submit to **Canvas**. **Please do not submit data or other large binary files — Canvas will not accept large submissions.**

## 1 Preparation for the Introductory Lab and Assignment

In the lab you will familiarize yourself with two pieces of software: CRF++, a popular CRF toolkit for supervised sequential labeling. You will use CRF++ for Named Entity Recognition (NER) by training and testing on CoNLL data (see Canvas course website), and OpenNLP, a Java library for variety Natural Language Processing tasks.

All code should be provided in the github repository that contains this assignment handout.

The code directory `code/NER` includes a README file with instructions for getting started. It is suggested to use an IDE such as Eclipse, IntelliJ, etc. Make sure you can run the classes listed in the README and obtain error-free output. You are free to borrow code from these classes for your assignment (indeed you'll probably want to).

The CRF directory `Tools/CRF++-0.58` also contains a README. This software is distributed as C++ source. To build an executable, you'll need "gcc" and "make" installed. (Feel free to grab a binary executable from a classmate or you can also download a Windows executable online.) Once you have a running executable, type `crf_learn` to verify it is correctly installed. More details of training model and testing are pointed to in the README file.

Sample data for CRF++ is as follows:

- `DATA/conll2002`: *Spanish* NER train and test data and model template
- `DATA/basenp`: Noun-phrase chunking train and test data and model template
- *Assignment 2 listing on Canvas*: CONLL English NER train and test data (please do not redistribute this data), but no model file

After a CRF evaluation run that outputs a `test_result` file, the results can be evaluated with the following Perl script (located in `DATA`):

```
perl conll-eval.pl -d '\t' < test\_result
```

Make sure you have Perl properly installed to run this script.

In the lab, you will be required to evaluate a CRF++ model on text data extracted from a web page (all within Java). An easy way to "call" the CRF++ binary from Java is to directly write the files required by CRF++ to disk and then invoke CRF++ on them from within Java and capture the output stream as follows:

```
Process proc = rt.exec("/path/crf_test -m model_file test_file");
proc.getOutputStream().close();
InputStream stdin = proc.getInputStream();
BufferedReader br = new BufferedReader(new InputStreamReader(stdin));
String currentLine = "";
while((currentLine = br.readLine()) != null ){
System.out.println(currentLine);
}
```

## 2 In the Introductory lab

In this introductory lab our goals are to (a) build a baseline NER system and understand the basics of model template specification and (b) understand how to load web pages into XML and and extract parts of this XML structure.

For part (a), if not already done above, please do the following:

- Download the CONLL English NER data from Canvas (three files: one train, two test).
- Load the following web page in the repository in a web browser: `CRF++-0.58/doc/index.html` — this provides instructions on training, testing, and model specification.
- Specify a model file with a single unigram feature for the current token (not provided).
- Follow the instructions to run `crf_learn` on the training data and `crf_test` on one of the test data files (a or b).
- Run `conll-eval.pl` described above to evaluation to determine the performance of your NER system on the test data.

Remember that it's always important to evaluate on the test data since it is easy to overfit (and do very well) on the training data, but not generalize well to previously unseen test data. Test data performance is an indicator of generalization.

Finally, note there are a large number of opportunities for modification and improvement of performance of this baseline by augmenting the features (columns in the train and test file) and by augmenting the model.

Augmented feature columns could consist of

- a boolean feature for capital letters (allowing you to lowercase the main column),
- a part-of-speech tag,
- a stem or lemmatization of a word,
- a canonical synonym,
- a boolean for whether or not that word corresponds to an article name in Wikipedia, or
- *anything else* you can derive from a word or its context that you think is useful for NER extraction.

Augmented models could consist of

- adding single observation features occurring in the context of the current token (current POS tag, token before, next POS tag, etc.),
- adding joint observation features consisting of tuples of the above (e.g., current token *and* next POS tag, current POS tag *and* token before *and* next POS tag), or
- combining the above observation features with both the current and next state (oddly called *bigrams* in CRF++ and indicated by `B`).

We'll explore augmented feature and models in the main assignment.

For part (b) of the assignment, you'll need to design a very simple *key content extractor* for a target news website, e.g., `washingtonpost.com`, where you can rely on all articles to be formatted consistently. To prepare for this, run the following classes in `code/NER/src`:

1. `web.reader.WebPageReader` to see the output and understand how a document is read into XHTML and the XML DOM — try changing the web page to a news article. Can you find the key content?

2. `util.XMLUtils PrintNode(...)` method to understand how all node contents and child nodes are accessed to achieve the XML printout of `web.reader.WebPageReader`.

3. `web.reader.XMLReader` to see the XML output from an really simple syndication (RSS) file for the SMH news website and then the results an XPath query on this XML.

4. `web.reader.XMLReader getXPathQueryResults(...)` method to understand how to make an XPath query on an XML document and process the results list.

There are various tutorials on XPath on the web and it will be covered briefly in lecture — you should only need to use *very* simple XPath queries for the assignment.

## 3 Main Assignment (Assessed in Evaluation Meeting)

### Q1. Baseline

(a) Report the accuracy of the baseline system. Look at some of the output from the test evaluation... do you think this would be acceptable in practice? [1 sentence]

### Q2. Improving performance

CRF performance is strongly influenced by the features used by the model, which includes both the raw features available to the system and how it combines those features in a model relative to the current token being evaluated as discussed in the Introductory lab.

In this question, we'll work on improvements to the baseline CRF model built in the introductory lab. All results will be reported as *accuracy*.

(a) Try training the model by augmenting the token context with different window sizes of single unigram features. Report results for windows of size 1, 2, 3 vs. the baseline (window size 0). Show a model for window size 2.

(b) Try augmenting the baseline model with two conjoint features of (i) the previous and current token and (ii) the current and next token. Report results for this model vs. the baseline and show the augmented model.

(c) Report results for a model that adds bigram features over states to the baseline vs. the baseline itself. What model file modification did you use for this?

(d) Select two different feature types and augment your training data with two columns for those feature types. Report results for the baseline model vs. (i) the model augmented with feature type 1 at the current token (window size 0), (ii) the model augmented with feature type 2 at the current token (window size 0), and (iii) the model augmented with feature types 1 and 2 at the current token (window size 0).

(e) Find a combination of features and/or models that improves on the results reported above and report the improvement over the baseline model. This best NER model is the one you will use for the next part.

The best overall result for the course is entered into the illustrious *OSU Document Analysis IE Assignment Hall of Fame.*

## Q3. Key Content Extractor for a News Website

Choose a news website, e.g., `washingpost.com`. Building on the introductory lab, write a small piece of code that loads an article (given the article URL) from that website and extracts the key article content into a Java String (some extraneous content like advertising within an article is permissible, but make sure that menu bars and side bars are not considered key content).

(a) Provide the source code of your XPath query and post-processing to build the String. [hopefully just a few lines]

(b) Provide a URL (not the content, the actual URL) and the Java String contents that your extractor finds for that URL.

## Q4. NER in the Wild (Web News)

Write Java code that takes a new web page (provided as an article URL) and extracts the NERs in a list (one NER per line) from the key content of that news web page, e.g.

```
[1] Barack Obama
[2] White House
[3] San Francisco
...
```

Run it on a few web pages and compare the output to the original key content (String) in order to assess errors. Note that the CONLL training and test data are generated from Reuters news articles from about 15 years ago.

(a) Provide a URL (not the content, the actual URL), the Java String key contents for that URL, and the NER list for that URL using (i) your baseline NER system and (ii) your best NER system.

(b) If the NER system makes more errors than you expected given it's accuracy on the CONLL test data, can you explain this? (If not, nothing to explain.) [1 sentence]

Note that to complete this task, you'll need to convert the key content first to a list of sentences and then to a list of tokens per sentence. The code in the `nlp` package provides tools for doing this (and more).