Comp 6751 # Project 1 **Fall 2023**

Due date: September 22, 2023

Goal: running text preprocessing pipeline in NLTK in Linux and proofreading results

Data: NLTK version of Reuter's text `training/267`

> INDONESIA UNLIKELY TO IMPORT PHILIPPINES COPRA
>
> Indonesia is unlikely to import copra from the Philippines in 1987 after importing 30,000 tonnes in 1986, the U.S. Embassy's annual agriculture report said. The report said the 31 pct devaluation of the Indonesian rupiah, an increase in import duties on copra and increases in the price of Philippines copra have reduced the margin between prices in the two countries. Indonesia's copra production is forecast at 1.32 mln tonnes in calendar 1987, up from 1.30 mln tonnes in 1986.

Overview: Do the following project in NLTK. Get the text using the NLTK corpus access commands, do not cut and paste from the assignment. Develop a single script called *PreProcess* that executes the following pipeline in NLTK, taking the file to be preprocessed as a parameter, *PreProcess(*`training/267`*)*

1. tokenization (from NLTK)
2. sentence splitting (from NLTK)
3. POS tagging (from NLTK)
4. gazetteer annotation (country, units, currency)
5. named entity recognition, NER (mostly from NLTK)
6. measured entity recognition (*5 dogs, 2 days, 2am, $15, 5mins, ...*)

Proofread every step in your pipeline. To save time and gain the anticipated insight, run your script on other texts as well. Solutions that only work on this text and not on others may not get full marks.

Description: Each step in your pipeline has specific additional requirements in order to be considered satisfactory. It is important that you do not limit yourself to the requirements, but think beyond the minimum requirements for your solution.

tokenization start with a regular expression based tokenizer from NLTK. Note that you are free (and possibly required to) improve on that tokenizer for best results. Copious in-line comments in the code for your changes are essential. The enhancements have to also be summarized in the report.

sentence splitting as for tokenization, start with a NLTK module. Enhance if necessary.

POS tagging inspect your options in NLTK. Pick the best module. Do not spend time to improve the POS tagger at this time.

gazetteer annotation for country names, currencies, and units, develop a gazetteer each (this does not have to be comprehensive at this point, but think how you would populate a more complete list). The simplest form of a gazetteer is a word list, as in `country = {Canada Denmark England France Germany Hungary Indonesia ...}`. For all your gazetteer lists, find a way to annotate all tokens in the text that occur in a gazetteer with the name of that gazetteer (i.e. annotate *(Indonesia country)*). Note that all entities of a gazetteer list are named entities by default.

Named entity recognition is one of the most frequently implemented modules, without being a solved problem. Study NLTK Ch 7, especially the final sections. NLTK provides a classifier that has already been trained to recognize named entities, accessed with the function `nltk.ne_chunk()`. If we set the parameter binary=True, then named entities are just tagged as NE; otherwise, the classifier adds category labels such as PERSON, ORGANIZATION, and GPE. The binary tagging is sufficient for this project, but experiment with the labelled categories, also. How will you integrate your own gazetteer lists into the NE module? Do you have any improvements you can make to the NE module?

measured entity detection units of measurements should be grouped with the numbers they usually follow. Develop your own *MeasuredEntityDetection* module that groups a number with its unit, if available (i.e. 30,000 tonnes). For this step, you should compile a unit gazetteer (best solution is to find a resource on the web and adapt its format to a simple word list).

For all modules, create your own test cases for a more general solution. Experiment on other texts from the NLTK Reuter's corpus.

Deliverables: to be submitted in Moodle by September 22, 2023

- (2pts) tokenizer, sentence splitter, POS tagger used (enhancements clearly identified in inline comments. (Attrib 1,5)
- (2pts) gazetteer annotation and resulting data structure (motivate what is included in the gazetteers and why, they do not need to be complete but cannot be limited to entities from the text above). (Attrib 1,12)
- (2pts) named entitiy recognizer (with clear indications on additions to NLTK). (Attrib 1, 12)
- (2pts) measured entity recognizer and resulting datastructure. (Attrib 2, 12)
- (1pts) Report: a .pdf document that documents your work and submitted modules. Make sure that the grammars are rendered completely in human readable form and that they are well motivated, possibly with examples. (Attrib 6)
- (1pts) Demo File: a .pdf document that shows examples of input and output for all modules and all interesting cases in a single "demo" file. Organize your demonstration of your demo to be readable. Do not include repetitive example runs. You must select the most helpful runs to show strengths of your modules/grammars. Errors or limitations must be addressed openly in the Report. Note that there are no solutions without errors or limitations. (Attrib. 6)

Marking scheme: Grading is based on two components. The first component is adherence to the instructions. This is expected to be 100%. The second component is what you bring to the task. We can only appreciate this, if it is well documented in the report and in the code and illustrated with convincing examples in the Demo File.

Solutions that work on a wider variety of input data are preferred over narrow solutions (coverage). Solutions that produce fewer false positives are preferred (specificity). There is no optimum and you have to explain your choices.