LING 406: Intro to Computational Linguistics

Spring 2020

Assignment #1: *Temporal Named Entity Recognition*

Issued: Jan. 28, 2020

Due: Feb 7, 2020 (by midnight)

Credits: 60 points

Objective

Searching large collections of unstructured data requires fast, convenient, and robust processing methods. Consider, for example, the amount of text data being generated on Social Media every day. Unstructured data is usually complex and noisy. Fortunately, there are techniques useful to extract important features from such type of data.

In this assignment, you will get hands-on experience with regular expressions that can be used in frequent pattern matching and searching as part of natural language processing applications. This combination of string manipulation functions and regular expressions will prepare you for more complex computational linguistics tasks later in the course. For example, some common text processing tasks where regular expressions can come handy are tokenization, chunking and stemming.

Description

For many text-based applications, date and time **named entity recognition** is a tremendously useful task, yet a challenging one to provide consistent results from application to application. The reason is that there are so many arbitrary dates, ranges, and holidays that can be expressed in so many ways and are context dependent.

In this assignment, however, you are going to build **a simple temporal named entity recognizer** using regular expressions. In particular, you are asked to design and write a program that recognizes simple date expressions of two types:

1. Fixed dates (including exact time if mentioned in the expression)

Examples: “January 15, 2014”, “the 21st of December”, “01/15/2014” (only the American notation), “Monday”, “Monday the 23rd”, “Monday, 2pm”, “Monday afternoon”

1. Holidays (only American holidays; to guide you, here is a list of Federal holidays: https://www.redcort.com/us-federal-bank-holidays/)

Examples: “Labor Day”, “Memorial Day”, “Christmas”.

Keep in mind that you have to identify and extract the longest expression mentioned to receive full credit. For example, if the text says

*“.. The mayor addressed the audience on Feb. 14th, 2012..*”,

you should extract "Feb. 14th, 2012" and not only "Feb. 14th".

Your program should recognize all such “absolute” dates, like those shown above and use representations that capture classes of words (e.g., holidays such as Labor Day, Memorial Day, Christmas, etc.). Your system should NOT extract dates relative to a particular day, like "the day before yesterday", “two years ago”, dates relative to temporal focus (“3 days later”), absolute dates with imprecise reference (“in the beginning of the 80s”), relative dates with special forms (seasons), basic duration ("during 3 years"), duration as interval (“from February, 11 to October, 27. . .”), relative duration, w.r.t utterance time (“for a year”), and temporal atom (“three days, four years, . . .”). Years before 1000 AD are not of interest here and should not be considered.

Grading

Your system takes as input a text file – e.g., a recent news article – and should identify and extract a list of dates as output (one per line) stored as output.txt. We will give you a sample of such input file, with all the correct expressions already manually identified. However, you are advised to test your program on other text data and make sure it works. We will run your program on the given input file as well as on another file of our choice and your grade will be proportional to the performance of the output.

You should also detail (in a file system-description.txt) the design decisions you made for your system (e.g., what regular expression types / data structures you decided to use to capture the different types of temporal named entities and why).

The assignment is worth 60 points: 10 points for the design and 50 points for the implementation of the system. Each type of expression considered (i.e., NOT individual instances) is 25 points (for a total of 50 points). For each type, we will deduct 5 points per missing date entity instance (if you miss more than one instance). We will also penalize for incomplete date instances.

Note: All programs have to be done in Python 3. It is your responsibility to make sure your code runs properly and accomplishes the task before you submit your work. You are advised to contact the TA (ahead of time) if you need assistance with the assignment.

Deliverables:

Using compass2g, upload the source code, a Readme file, the output.txt file, the system-description.txt file, as well as any files we need to access to grade the assignment correctly. The Readme file should indicate the name of the program, a short description of the problem solved and a short description on how to run the script(s). You must comment your code accordingly (i.e., following good coding practice guidelines). 5 points will be deducted if any of these deliverable files is missing. As for the code, delivering a jupyter notebook is highly encouraged (although providing a .py file is also ok).

--

Other good examples of temporal entities:

* He will arrive on **January 5th, 2015**.
* The king was born on **the 25th of February**.
* **Tuesday, March 15** is my birthday.
* Dateline: **04/01/1999**
* We fly to Denver on **Monday**.
* I think it happened **Aug. 23, 2000**.
* My birthday is **the third of Oct.**
* On **10/31** we dressed in costumes.
* My grandmother was born in **1935**.
* **Thursday, Nov. 28th, 1948** was an interesting day.