COMP 150-06 Natural Language Processing
Spring 2016

Problem Set 1: Regular expressions and finite-state automata

Natural language text is full of date expressions. Detecting and extracting these is necessary not only in text preprocessing steps such as tokenization, but also in information extraction systems that utilize dates of important events etc. Dates come in a variety of formats (*January 1st 2000, tomorrow, next Tuesday*) however for the purposes of this problem set we will focus on a standard format such as MM/DD/YYYY, e.g., *12/31/2000*.

1. Create regular expressions for each component of the date expression. Make sure that your RE are as succinct as possible by using the operators that we discussed in class. Explain in a sentence or two how you built each one of your RE.
    1.1. MM component: Assume that months are in [01, 02, …, 12] inclusive.
    1.2. DD component: Assume that days are in [01, 02, …, 31] inclusive, irrespective of which month/year it is. (This is a simplification)
    1.3. YYYY component: Assume that years are in [1900, 1901, …, 2099] inclusive.
    1.4. Separator component: Assume that valid separators are space, dash and forward slash. You do not have to force the two separators in MM/DD/YYYY to match, that is, we will assume that *12-31/2000* and *12/31-2000* are also valid dates.
2. Draw FSA corresponding to each component and describe how you would build an FSA for the entire expression out of these individual components (No need to draw the final FSA as it might be too large). When you draw each FSA, include a paragraph explaining your design choices. Number your states with integers so that 0 is always the initial state. Clearly mark your initial and final states.
3. Implement an FSA class in Python. In **pset1_template.py** we provide you with a recommended prototype that you can follow. You can use your own design if you prefer but make sure to explain your choices. Create FSA objects for each of the components above.
4. Implement the D-RECOGNIZE algorithm of SLP Figure 2.12 using your FSA class. The signature of this function should be such that we can call it as `DRecognize("01", fsa)` and it returns `True` or `False`.
5. Extend D-RECOGNIZE into `DRecognizeMulti` so that it takes as input a list of FSA instead of a single one. This algorithm should accept/reject input strings such as *12/31/2000* based on whether or not the string is in the language defined by the FSA that is the concatenation of the input list of FSA. Use the `Test` function provided in pset1_template.py to demonstrate your algorithm on recognizing date expressions.
6. **Extra credit:** Introduce nondeterminism in your MM and DD components so that dates such as *1/2/2000* as well as *01/02/2000* can both be recognized (single-digit and double-digit months and days): Extend your FSA class so that it supports nondeterministic FSA. Implement ND-RECOGNIZE of SLP Figure 2.19 and demonstrate that it accepts/rejects valid/invalid date expressions. You can do this in one of two ways:

6.1. Implement `NDRecognize` and `NDRecognizeMulti` (similar to steps 4 and 5).

6.2. Build one large nondeterministic automaton for date expressions MM/DD/YYYY and use `NDRecognize`.

Deliver the following by Friday 2/12/2016 11:59 PM.

1. A PDF write-up. All questions 1-6 attempted should have at least a paragraph describing your reasoning and final solution, explaining your decisions in detail. The write-up should include code blurbs if necessary.
2. Your code. This is a .py file that compiles and runs. It should demo the answers that you produced in your work.

How to submit your homework:

1. Log onto the Tufts server:
   ```
   ssh your_username@homework.cs.tufts.edu
   ```
2. Navigate to the directory of your submission files and type the following command to submit all of your files together:
   ```
   provide comp150nlp pset1 [file1 file2 ...]
   ```