COMP 150-04 Natural Language Processing
Fall 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*.

Note: Throughout this pset, FSA will refer to nondeterministic-FSA (NFSA).

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. Make the initial zero optional so that 01 and 1 both are valid and refer to the same month).
   1.2. DD component: Assume that days are in [01, 02, …, 31] inclusive, irrespective of which month/year it is (This is a simplification). Make the initial zero optional so that 01 and 1 both are valid and refer to the same day).
   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 individual FSA corresponding to each component. 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. Build a nondeterministic recognizer for each component:
   3.1. 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.
   3.2. Implement the ND-RECOGNIZE algorithm of SLP Figure 2.19 using your FSA class. The signature of this function should be such that we can call it as `NDRecognize(<input string>, fsa)` and it returns `True` or `False`.
   3.3. Run the `TestComponents` function provided in pset1_template.py to demonstrate your system accepting/rejecting the right expressions, include the output in your write-up.
4. Build a nondeterministic recognizer for entire dates:
   4.1. Implement a function `Concatenate(fsa1, fsa2)` that will input two FSA and return the concatenated FSA so that if w_1 is a string that fsa_1 accepts and w_2 is a string that fsa_2 accepts then w_1w_2 will be accepted by the concatenated FSA.

4.2. Using your concatenate function and your individual component FSAs, create FSA for the entire date expression *MM/DD/YYYY*.

4.3. Run the `TestDates` function provided in pset1_template.py to demonstrate your system accepting/rejecting the right date expressions, include the output in your write-up.

Deliver the following by Friday 9/23/2016 after midnight 02:59 AM.

1. A PDF write-up. All questions 1-4 attempted should have at least a paragraph describing your reasoning and final solution, explaining any decisions you had to make in detail (include code blurbs if necessary). <u>The write-up is where your outputs (answers) should go, since we cannot guarantee to search the output of your submitted code to find your answers.</u>
2. Your code. This is a .py file that compiles and runs. It should demo the answers that you produced in your work. <u>But again, include these answers in your write-up, just printing on the console output will not get you the right grade.</u>

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 ...]
   ```