Project Warmup

First Attempt on Slot Filling

Due: Sunday 5pm 1/20

Read the EC2 tutorial (Link) before working on this assignment. Don't forget to redeem the $100 code (instructions in the tutorial).

In this assignment, you'll be asked to write a simple extractor based on regular expressions. It might not work very well but the process should hopefully get you familiar with the data and the task.

We prepared a prototype system for you written in Java. It implements a simple pipeline: reading queries, scanning sentences[1], filling the slot for each query, and then evaluates the results against gold standards (*Assignment1.java; hereafter the italicized font indicates a class name*). Please read the data format document so that you will know how to process the data.

The code is located at <volume-mounted-location>/assignments/a1[2]. Change your current directory to there. To begin with, you need change the root path (*rootPath* at src/tackbp/*KbpConstants.java*) if you mount the volume not in "/kbp". Run "java -cp bin Assignment1"[3] and you will see some results for the "per:date_of_birth" slot. You task is to build an extractor using any methods you can think of, and using as much of the preprocessed data as you want. (Documentation on the data formats is HERE and we'll cover it to you in class. But for this warmup part of the project, you don't need to use anything besides the token file iif you don't want to or don't understand it). Specifically, you could mimic *RegexPerDateOfBirthFiller* by writing simple regular expressions and matching them on the sentences of the token file.

Specifically, create a class named "<SlotName>Filler" extending *sf.filler.Filler*. Remove the punctuations in the slot name and make it camel. e.g. for "per:date_of_birth", the class name is "PerDateOfBirthFiller".

You assigned slot will be one of the following. Determine it by taking your student id mod 17.

- per:date_of_birth * done in the example code
0. per:age
1. per:country_of_birth
2. per:stateorprovince_of_birth

---

[1] To shorten the running time, we prepared a special and much smaller data set where only documents that have answers are provided. In the real task, many more sentences will need to be processed.

[2] Individual jar files of the assignments will be available for download.

[3] Java 7 is required.

3. per:city_of_birth
4. per:date_of_death
5. per:country_of_death
6. per:stateorprovince_of_death
7. per:city_of_death
8. per:cause_of_death
9. per:religion
10. org:number_of_employees/members
11. org:founded
12. org:dissolved
13. org:country_of_headquarters
14. org:stateorprovince_of_headquarters
15. org:city_of_headquarters
16. org:website

Turn in the following to https://catalyst.uw.edu/collectit/dropbox/xling/25619:

1. The code you implemented for the slot filling.
2. A report that explains what relation you tackled and what you did in the code. For example, if you created a bunch of regular expressions, explain them in English. If you tried something more complicated, please also explain that. Report the performance. It's ***absolutely*** ok to have poor performance but some intuitions and analysis of the errors would be ideal (*sf.ErrorAnalysis* can help you identify the error for a given slot name.).
3. Good luck! And remember, we'll be discussing more advanced extraction methods in class, so our performance will improve.