CSC 247/477 Programming Assignment #2 **Due April 7th, 2015**

This assignment involves building part of a semantic parser that uses the structure of a verb sub-categorizations to simultaneously disambiguate the verb sense and the semantic roles for that verb.

Specifically, you should write a program that takes a dependency parse of a sentence, such as the following for the sentence "he pushed the box" (obtained from the Stanford dependency parser):

nsubj(push-2, he-1)
root(ROOT-0, push-2)
det(box-4, the-3)
dobj(push-2, box-4)

You then will use a simplified version of the TRIPS verb lexicon entries to identify the ontology type for the main verb and its semantic roles. For example, two senses of "push" are shown here in the sidebar. In its PROVOKE sense, "push" has a subject NP, and object NP and a complement that is a VP of type TO. In its PUSH sense, "push" has just a subject NP and an object NP, and no complement.

By matching the entries to the dependency parse, you would find the second entry as the best match, so would output

ONT::PUSH AGENT he-1 AFFECTED box-4

Lexical Entries for "Push"
1. ONT::PROVOKE
"He pushed him to do it"
LSUBJ (NP) -> AGENT
LOBJ (NP) -> AFFECTED
LCOMP (VP TO) -> FORMAL

2. ONT::PUSH
"He pushed the truck"
LSUBJ (NP) -> AGENT
LOBJ (NP) -> AFFECTED

As another example, the sentence "The event pushed me to resign", has the dependency parse

```
det(event-2, the-1)
nsubj(pushed-3, event-2)
root(ROOT-0, pushed-3)
dobj(pushed-3, me-4)
aux(resign-6, to-5)
xcomp(pushed-3, resign-6)
```

Which would match the first sense of "push" and produce the output

ONT::PROVOKE AGENT event-2 AFFECTED me-4 FORMAL resign-6

Part 1: Write a program that takes in a dependency parse representation (we will provide the files) and the TRIPS simplified lexicon (provided in YAML format), and produces a one line output that identifies the main verb type and its semantic roles. The Chart below summarizes all the different subcategorization forms you will see in the lexicon:

constit	*feature*	*defn*	*example*
NP		Any noun phrase	
PP	x	Any PP with preposition x	I went **to the store**. I came **from there**.
VP	TO	Infinitive VP	I like **to eat.**
VP	ing	Present participle VP	I like **eating**
VP	that	Sentential complement	I know **that I saw him**
VP		Any other VP	

CSC 247/477 Programming Assignment #2 **Due April 7th, 2015**

Part 2: Extend you system to handle relational roles, such as the result role in "He pushed the box **into the corner**"

nsubj(push-2, he-1)
root(ROOT-0, pushed-2)
det(box-4, the-3)
dobj(pushed-2, box-4)
prep(pushed-2, into-5)
det(corner-7, the-6)
pobj(into-5, corner-7)

To do this, you need a general rule matching preposition/adverbial modification. With your extension, and given the above lexicon entries for "push" , your program would output

ONT::PUSH AGENT he-1 AFFECTED box-4 RESULT into-5

Extend your program to identify at least three relational roles. You can get extra credit for handling additional cases.

Documentation

Documentation is as important as the code. Make sure your system is well described. Pay particular attention to describing your approach to handling relational roles. Identify the strengths and weaknesses of your algorithm.

As with assignment one, hand in your code with your results so we can test it on the test sentences we will send you, plus a range of additional sentences to test how general your approach is.

Questions

If you have questions on getting the project started, using the data, or the evaluation, please feel free to contact Omid (omidb@cs.rochester.edu)**.** Omid will be sending you out the lexicon file in YAML format, as well as some dependency parses to test your system. To make things easier for you, these files will have the verbs already stemmed. For example, instead of have *pushed-2* in the dependency parse, you will get *push-2*, so it is easier to look up the verb in your lexicon.

Technical descriptions:

You will have two YAML files for this assignment with a readable format.

1. Lexicon File: Contains all information you need to access any lexical data in TRIPS. It has the following structure (a list of YLexicalItem):

   class YLexicalItem(words:String, ontotypes:String, frames: List[YTemplate])

   class YTemplate(id:String,name:String, argments: List[YTemplateArg], example:String)

   class YTemplateArg(syntax:String, constituent:String, restrictions: List[String], semantic:String)

2. StanfordParser Data: It contains List of parsed sentences, I also put them in YAML format so you can easily parse them. It has the following structure (a list of ParseOutput):

   // words are index and value of words, and lemmas are index and value of stemmed version of words

   class ParseOutput(sentence:String, words:List[Element], lemma:List[Element], outgoingEdges:List[ParseObject] , incommingEdges:List[ParseObject],roots:List[String])

   //node:current word, target: target word (outgoing or incoming)

   class ParseObject(node:String, target:String,label:String)

   class Element(index:String,value:String)

Your program should take these two files and output the correct lexical item and frame as the output. Your code should be executable on department machines.