Ling 473 Project 1
Due 11:45pm on Tuesday, August 6, 2013

For this project you will write a program to count the number of several syntactic constituent types that occur in an annotated corpus. Processing all of the files in the following directory, fill out the table below to indicate how many of the syntactic elements of each type are annotated in the entire corpus. Nested constituents of the same type are to be counted equally at any level where they appear.

`/corpora/LDC/LDC99T42/RAW/parsed/prd/wsj/14`

This is a portion of the Treebank-3 corpus from the Linguistic Data Consortium. You are not permitted to copy this corpus off the computational linguistics cluster. Note that these files have only the parse tree node labels and not the terminal POS tags[1].

Constituent	PTB symbol	Count
Sentence	`(S ...)`	
Noun Phrase	`(NP ...)`	
Verb Phrase	`(VP ...)`	
Ditransitive Verb Phrase	`(VP` *verb* `(NP ...) (NP ...) )`	
Intransitive Verb Phrase	`(VP` *verb* `)`	

For the last example, we are looking for VPs whose immediate (top-level) constituents include no NPs (or no immediate children at all). It turns out that this will actually give a lot of auxiliary verbs, and also the sentence complementizers such as "said."

For the Ditransitive case, we are looking for exactly two immediate constituents of type NP. Do not count NPs that are marked as, for example, NP-SBJ. Dealing with nesting and making sure that you only consider the immediate constituents will be tricky, especially if you are using RegEx, since RegEx does not easily handle matching of balanced parentheses.

You can use any programming language that is available on *patas*, including shell scripting, which may be adequate for this assignment. You do not need to use regular expressions in your program but you are welcome to use this method if you find it convenient. Well-written procedural code is often more self-documenting and maintainable than elaborate regular expressions.

Use absolute paths (path that starts with '`/corpora/...`') to reference the corpora. Use relative paths (paths that do not start with '`/`') to reference files you are including with your submission. Do not directly reference your home directory, since I may not have permissions for it when I'm running your program.

Output Format

In order for the automated checker to verify your program's output, it must include a line that matches this format exactly (not literally of course, this is all description):

```
<input file path>TAB<decimal count from first row>[SPACE or TAB<next row's count>]{4}
```

[1] IYI[2]: The `/corpora/LDC/LDC99T42/RAW/parsed/mrg/wsj/14` directory contains matching files with both sets of labels ("prd" is short for "parsed", "mrg" is for "merged").

[2] IYI: "If You are Interested."

Submission

For this project, you will also create a control file that submits your program as a batch job to the `condor` computing cluster. Include the following files in your submission:

`compile.sh`	Contains command(s) that compile your program. If you are using python, shell scripts, or any other interpreted language that does not require compiling, then omit this file, or use one that does nothing such as the single line: `#!/bin/sh`
`run.sh`	The command(s) that run your program, emitting required output to the console (stdout). Be sure to include compiled binaries in your submission so that this script will execute without first running compile.sh
`condor.cmd`	Condor control file, suitable for running your program as follows: `condor_submit condor.cmd`
`output.txt`	captured console output (stdout) from running your program, this should be produced by running your Condor job.
`readme.{pdf, txt}`	Your write-up of the project, including a table similar to the one above which reports your results. Describe your approach, any problems or special features, or anything else you'd like me to review. If you could not complete some or all of the project's goals, please explain what you were able to complete.
`(source code and binary files)`	All source code and binary files (jar, a.out, etc.) required to run and compile your program. Use subdirectory structure as appropriate.

Gather together all the required files, making sure that, for example, any PDF or other binary files are transferred from your local machine using a binary transmission format. Then, from the parent directory of the one containing your files, issue the following command to package them for submission.

```
tar czf project1.tar <your project dirname>
```

Upload this single tar file to CollectIt.

Grading

Correct results	30
Clarity, elegance, and readability of code	25
Followed submitting instructions	15
Write-up	15
Runs as-is	15

Notice that correct results gets a relatively small number of points compared to the sum of the rest of the elements of the rubric. I strongly recommend getting a complete submission working with simplest possible implementation (such as reporting all zeros) so that it can be turned in on time before spending much (or even any) time on implementing the specified algorithm.

Corpus Citation

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz and Ann Taylor. 1999. *Treebank-3*. Linguistic Data Consortium, Philadelphia