Ling 473 Project 1
Due 11:59pm on Thursday, August 5, 2021

For this project you will write a program to count the number of syntactic constituent types that occur in an annotated corpus. Processing all of the files in the following directory, fill out the table below (i.e., Table 1) 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.

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

Table 1: The count of different constituent types in **`/corpora/LDC/LDC99T42/RAW/parsed/prd/wsj/14`**

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* (see example codes under **`/dropbox/20-21/473/code-samples/`**), including shell scripting, which may be adequate for this assignment. You are not required 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.

Running the Program

The program should accept exactly one argument which is the absolute path to the directory containing the files to be processed. Do not hardcode the path in your code, which will disallow us from testing the code on new data. For other path, use the 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.

The program prints the required output to the standard output (e.g., use 'print' function is python). The format of the output should follow "Constituent type + a tab + the count". That is, the output should match the format in **`/dropbox/20-21/473/project1/example.output`**. This is important because we are going to use an auto-grader to compare your output to the gold standard; following the same format will make the grading easier.

Run your code directly on Patas (not using Condor) with the following command line

```
./run.sh /corpora/LDC/LDC99T42/RAW/parsed/prd/wsj/14 >output
```

and obtain the "output" file. You can test your code on a small set of data which is offered to you at `/dropbox/20-21/473/project1/example_input`. Ideally, if you run your code with

```
./run.sh /dropbox/20-21/473/project1/example_input >ex.output
```

your output should match the one at `/dropbox/20-21/473/project1/example.output`.

Running the Program with Condor

Create a Condor control file and name it as "condor.cmd". This file contains the configurations to run your code on Condor. Use "condor_submit condor.cmd" to submit and run your code on condor. You can find a simple example under **`/dropbox/20-21/473/condor-samples`**. For more information, visit https://wiki.ling.washington.edu/bin/view.cgi/Main/HowToUseCondor. Hint: you may want to use "arguments" in "condor.cmd" to specify the arguments that you want to pass to the code (e.g., arguments `= "/corpora/LDC/LDC99T42/RAW/parsed/prd/wsj/14"`).

Submission

Submit "**hw.tar.gz**" and "**readme.{pdf, txt}**" to Canvas.

The "hw.tar.gz" is a package that contains all the code and required output files. It should contain the following files:

`run.sh`	The command(s) that run your program, emitting required output to the console (stdout).
`condor.cmd`	Condor control file, suitable for running your program as follows: `condor_submit condor.cmd`
`output`	captured console output (stdout) from running your program, this should be produced by: `./run.sh /corpora/LDC/LDC99T42/RAW/parsed/prd/wsj/14 >output`
`(source code and binary files)`	All source code and binary files (jar, a.out, etc.) required to run and compile your program

To get "hw.tar.gz", gather together all the required files, and then, from within the directory containing your files, issue the following command to package your files for submission.

```
tar -czf hw.tar.gz ./*
```

Notice that this command packages all files in the current directory; do not include any top-level directories. Hint: it is recommended to submit the output file you obtained on Patas.

Check whether your submission contains all required file using

```
/dropbox/20-21/473/project1/check_project1.sh
```

The **readme** file is your write-up of the project, including Table 1 which reports your results. Describe your approach, any problems or special features, or anything else you'd like the grader to review. If you could not complete some or all of the project's goals, please explain what you were able to complete.

Corpus Citation

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