Ling 473 Project 2
Due 11:45pm on Tuesday, August 15, 2017

For this project you will write a program to clean a corpus and tally the words in it. The resulting output is called a unigram language model. This is one of the most common tasks in computational linguistics text processing, as it is a prerequisite for many of the more advanced NLP tasks. For this task you will use part of the AQUAINT corpus of English newswire. You will process all of the files in the following directory, which contains 249 files containing New York Times news articles in SGML format.

`/corpora/LDC/LDC02T31/nyt/2000`

1. First, you will clean the files of formatting. The only appearance of the less-than (<) and greater-than (>) symbols in these files is for delimiting tags that we wish to discard. Accordingly, you will ignore all SGML tags, retaining only content that is outside of any SGML tag.

2. We are only interested in words, not punctuation, symbols, or digits. The next step in cleaning will be to gather all of the words. For this project, a word is defined as a contiguous occurrence of one or more acceptable characters. The acceptable characters include only the following:

    a. Capital letters `A` through `Z`
    b. Lower-case letters `a` through `z`
    c. The straight apostrophe, `'` which is ASCII character 39, (\x27).

3. After gathering these words, perform the following operations:

    a. Trim any occurrence(s) of the straight apostrophe from the beginning and end of the word. Note that this preserves internal occurrences of this character, as in the words "Marvin's" or "O'Conner."

    b. Convert every word to lower-case the word

4. Finally, you will tally (count) the number of instances of each word, and write this information, sorted according to descending order of frequency, to the console (stdout). Each line will consist of the word, exactly one tab character (ASCII 9), and then a number representing the total number of instances of that word in the entire corpus.

Suggestions

For some programming languages, the most effective way to achieve step 2 is to first load the contents of the file into a string, replace all characters in this string that are *not* acceptable with a space character, and then use a string "split" function (specifying the space character for the character to split on) to obtain all of the words. Depending on language, you might have to then discard empty strings from this result.

For tallying, you will most certainly want to use a hash table that maps a string (the word) to an integer (the current tally for that word). Depending on programming language, this data structure is called a *dictionary*, *map*, or perhaps *associative array*.

Because the results must be presented in sorted order, you must maintain the tallies in memory and process all the files, prior to sorting the results and writing them to the console.

Instructions

You can use any programming language that is available on *patas*. Use absolute paths (path that starts with '/corpora/...') to reference the corpora. Use relative paths (paths that do not start with '/') to reference any additional files you are including with your submission.

Output Format

Your program will write the sorted unigram language model to the console (stdout). Depending on programming language, this is achieved by using a statement such as 'print,' 'printf,' 'WriteLine', etc. The results are written in descending order of their count. Here is an example of what the first few lines might look like:

```
the     26799464
of      11334126
to      10920886
and     10016354
a       9642340
in      9057801
that    4179900
for     4108357
on      3401376
etc...
```

Not visible in this example is the fact that there is just a single 'tab' character between the word and the tally. Do not use any space characters for this.

Submission

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 this file will be empty, or contain just the single line: `#!/bin/sh`
`run.sh`	The command(s) that run your program. If you use a compiled language, be sure to include compiled binaries (if any) in your submission so that this script will execute without first running compile.sh
`output`	This is the captured console output (stdout) from running your program, and must be formatted according to the "Output Format" instructions above. This file should be produced by: `./run.sh >output`
`readme.{pdf, txt}`	Your write-up of the project. 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., if any) required to run and compile your program

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 within the directory containing your files, issue the following command to package your files for submission. Replace the bracketed portion with your UW NetID.

```
tar -czf [your-uw-netid].tar.gz *
```

Notice that this command packages all files in the current directory; do not include any top-level directories. Upload the file to CollectIt.

Grading

Correct results	30
Follow submitting instructions	10
Clarity, elegance, and readability of code	25
Program efficiency	20
Write-up	15

Corpus Citation

David Graff. 2002. *The AQUAINT Corpus of English News Text*. LDC, Philadelphia.