Answers and Explanations for Lab 1

Author: Sharon Goldwater
Date: 2014-09-01, updated 2015-09-15, 2016-09-23

This document is available as a web page or pdf document.

Downloading the data

ls

If you have downloaded and unzipped the corpus as instructed, you should see several subdirectories inside the `Providence` directory, each named after a child: Alex, Ethan, Lily, Naima, Violet, and William. There is also a file called `0metadata.cdc`. Inside each child's directory there are a large number of `.cha` files, numbered in order.

Looking at the data

less

Each of the `.cha` files starts with metadata listing the language(s) being used; the participants and their three-letter codes (used to identify them in the transcription below); the age, sex, and birthdate of the child; and the age and birthdates of any other participants. There are sometimes also additional comments about the language/dialect of the participants, the situation being recorded, the transcriber(s), etc. (see William's files for example).

The remainder of each file contains a transcription of the participant's interactions, with each line indicating who spoke that line, the words being spoken, and the timestamp.

The main difference you will see between files with lower numbers and those with higher numbers is that in the lower-numbered files, the child hardly speaks, whereas the later files contain quite a few child utterances. This is because the files are numbered in order of the child's age (which can be verified by checking the ages in the metadata), and in the earlier files the child is too young to speak.

head, tail

The `head` command by default prints out the first 10 lines of a file; the `tail` command prints out the last 10 lines.

The `*` is a wildcard, so `head *.cha` prints the first 10 lines of each file ending in `.cha`.

Finding and counting things in files

wc

`wc` stands for *word count*, although in fact it returns three numbers: the number of lines, words, and bytes (usually, characters) in a file.

`wc eth0*.cha` will give you the counts for the first nine files, from which you should be able to see that `eth08.cha` has the most lines. (If you have already done the rest of the lab and are familiar with the pipe operator, try using `wc* | sort` to make it easier to see which file has the most lines. We didn't cover the `sort` command in the lab, but it's worth knowing!)

grep

`grep 'MOT' eth01.cha` prints out all the lines in `eth01.cha` that contain the string `MOT`.

To count instead of print matching lines, use the `-c` option.

`grep 'MOT.*the' eth01.cha` prints out lines containing the string `MOT` followed by the string `the`, possibly with some other characters in between.

pipes and redirection

To count the number of lines with `'MOT'` without using `-c`, we can do:

```
grep 'MOT' eth01.cha | wc
```

grep again

In the first command, the regular expression matches lines with `MOT` followed at some point by `the`, the string `the` does not need to be a complete word, so it could match, e.g., the word `there` or `father`. In the second command, the regular expression will only match `the` if it is surrounded by word boundaries, i.e., whitespace. So we could actually use the second version to find or count the number of lines in which the mother used the word `the`, whereas the first version would match more than those.

There are a few lines at the top of each file in the metadata that include the string `'MOT'` but are not utterances spoken by the mother.

Computing MLU

To find Ethan's MLU, first we do

```
grep '^\*CHI' eth50.cha | wc
```

which gives us

```
521    3466   23250
```

However, we know that these word counts include 1 timestamp at the end of each line and 1 speaker ID at the beginning of each line, so to find the true number of words (as defined in the question) we need to subtract 2 from each line, giving 3466 - 2*521 = 2424. The MLU is then 2424/521 = 4.6. This is of course still an overestimate of a realistic MLU, since it includes punctuation and possibly other things that wouldn't be considered words by a child language researcher.

Going Further

1. You could use `grep -ow 'the' | wc`. The `-w` means that `the` has to be a word by itself, i.e., equivalent to `\bthe\b`. The `-o` prints out each match by itself on a separate line. But as a result it's hard to know if you are accidentally matching things you don't want to: the output of `grep -o 'the'` and `grep -ow 'the'` *looks* the same. So it is really good to test your grep command, for example enter `grep -ow` (or `grep -w` or `grep -o`) by itself so it reads from standard input, and then try typing some test lines to see what it is doing. This sort of testing is always a good idea, especially if you're not sure you understand how the command line options work.

2. One way to do it is to replace all spaces and tabs by newline characters to get all the words on separate lines, then grep as usual: `tr ' \t' '\n' < eth01.cha | grep -c '\bthe\b'` (or you could replace the `grep -c '\bthe\b'` with `grep -cw 'the'`).