Digital Humanities

Tutorial "Regular Expressions"

March 10, 2014

1 Introduction

In this tutorial we will look at a powerful technique, called 'regular expressions', to search for specific patterns in corpora. Regular expressions can be used in many different programs, including the programming language Perl (where they originate), Excell (with the right package installed), the text editor Emacs and several specific programs such as RegExCoach for Windows. On linux computers and linux emulators (such as cygwin on windows), there is the standardly installed program 'grep', which we will use today.

2 Getting started

In this tutorial, we will use a command-line interface known as a *shell*. On Windows, a shell is provided with the cygwin package. Double click on the 'cygwin terminal' icon on your desktop, or select 'cygwin' from the program menu, and you are ready to start the exercise.

> If cygwin is not installed, you can try to install it from `http://cygwin.org`: choose 'download' and download the program 'setup.exe' (using the right-mouse click). Go to your download folder, rename 'setup.exe' to 'cygwin-setup.exe' and double click it to run the program. Select a 'mirror' nearby (i.e. one of the mirrors that end in `.nl`) and install cygwin with all the default settings.
>
> Alternatively, if you have access to a unix/linux server, you can use a secure shell client such as `putty.exe` to log in remotely. if you have a Mac (running OSX), there is a built-in unix shell. If you have an android system, you can try installing the 'bash shell' app. If you have a linux system, you can simply open a terminal from the menu under 'system'.

3 Preview

To try the command interface, type:

```
echo "FINISHED FILES ARE THE RESULT OF YEARS OF SCIENTIFIC STUDY
COMBINED WITH THE EXPERIENCE OF YEARS"
```

Question 1 *How many occurrences of the letter F are there in that sentences?*

Using regular expressions, we can also get the computer to do such counting for us. With the up arrow key you can get the previous command back. Now add some further commands such that the whole line looks like the following and then hit enter:

```
echo "FINISHED FILES ARE THE RESULT OF YEARS OF SCIENTIFIC STUDY
COMBINED WITH THE EXPERIENCE OF YEARS" | sed 's/ /\n/g' | grep 'F'
```

If you have never used a linux shell before, try the commands `pwd`, `ls`, `echo "hello world"`, `echo "hello world">myfirstfile.txt` and `less myfirstfile.txt` (each followed by the enter key). They will show you where you are in the file system, tell you which files there are in the current folder, output "hello world" to the screen, to a file, and show you the contents of that file, respectively. Hit the up and down error keys to see (and reuse) the earlier commands you typed. When something unwanted happens, hitting `ctrl-c ctrl-c` is often the way to return to the familiar cygwin terminal.

4 Download some data

Make a folder `dighum14` in your MyDocuments folder. Try to move there within the cygwin environment by typing something like (pay attention to the slashes, backslashes and spaces):

```
cd /cygdrive/c/Documents\ and\ Settings/jelle/My\ Documents/dighum14
```

The exact 'path' to your own folder will differ from computer to computer. Start with typing just `cd /cygdrive/` and then hit the tab key to see all the possible completions of your command.

In your browser, go to `http://www.illc.uva.nl/laco/clas/dighum14/` and download the file `corpora.zip`. Move it to your working directory, unzip it (right-click on the zip-file and choose 'unpack all files') and make sure all corpora are in a folder 'corpora' inside your folder 'dighum14'.

Now, in cygwin, run your first command that involves a regular expression:

```
grep -E 'ZZZ' corpora/welcome
```

If you see a couple of lines that all start with 'ZZZ' everything is working well. You have just run the grep program with a regular expression `ZZZ` on the file `welcome`. The programs returns by default only the lines of the file that contain a *match* with your regular expression. With `less corpora/welcome` you can see the full contents of that file (use `q` to return).

You can also let grep count the number of lines that match it by adding `-c` to you command, or return the precise matches with `-o`, or return the lines that do not contain a match with `-v`. Try!

5 Regular expressions

5.1 Concatenation & alternation

With regular expressions, we can thus search for a specific word, but more interestingly, we can search for more abstract patterns. For instance, the file `corpora/ts-text-english.txt` contains the text of the novel Tom Sawyer; if we try to count the number of lines on which the name 'Tom' occurs, we quickly discover the name is also written as 'TOM'. The regular expression `'TOM|Tom'` matches both alternatives. Alternatively, we can write that as `T(O|o)(M|m)`.

Question 2 *How often does the name appear in Tom Sawyer? How often in its Dutch translation? How often with an exclamation mark added?*

But there is a catch, because there might be occurrences of the sequence T-O-M that do not correspond to the name, but are part of larger words. For instance try:

```
grep -E 'TOM' corpora/ts-text-english.txt
```

We can make sure that our regular expression only matches complete words, by using the special character for 'beginning of word' `\<` and the special character for 'end of word' `\>`.

```
grep -E '\<T(O|o)(M|m)\>' corpora/ts-text-english.txt -c
```

5.2 Sets & Repetitions

Dutch spelling involves many cases where two characters are used to represent one sound. For instance, 'oo', 'aa', 'ie' are very frequent. To get an idea which characters from the set 'aeuioj' can follow an 'i', we can use the regular expression notation `[aeuioj]` to indicate 'any member of that set'. Try the following command, which uses grep's option `-m` to restrict the number of matching lines it considers[1]:

```
grep -E 'i[aeuioj]' corpora/ts-text-dutch.txt -o -m 10
```

We can use this way of writing down sets in combination with special symbols for repetition to look for repetitions of characters from a specific set. For instance, `e[aeuio]+` matches every 'e' followed by any number of repetitions of another vowel; `e[aeuio]{2}` matches every 'e' followed by *exactly* two repetitions of another vowel; `e[aeuio]{2,}` matches every 'e' followed by *at least* two repetitions of another vowel.

In the grep documentation (available with the command `info grep`; quit with `q`) you can find the meaning of remaining repetition operators `?`, `*`, and `{n,m}`.

Question 3 *Which triplets of vowel characters exist in Dutch? And in English?*

Question 4 *What is the longest consonant cluster in these languages?*

To specify sets, we can also make use of ranges: e.g., `[A-Z]` means 'all capitals' and `[0-9a-z]` means 'the set of all single digits or lower case characters'. Furthermore, some sets are frequently needed and therefore defined with a special name; e.g., `[[:punct:]]` matches all punctuation characters (see documentation). Finally, two important special symbols are the dot `.`, which represents the set of all possible characters, and `^`, which is used to indicate everything that is *not* in the set. That is, `(.+)` matches anything between parentheses (the brackets included), while `[^()]` matches any string that doesnot contain brackets.

5.3 Backreference (optional)

If we mark a specific part of a regular expression with brackets `()`, we can refer back to it with `\`n, where n is 1,2,3,... referring to the first, second, third, ... part between brackets. For instance, in `(.+) \<\1\>`, the first part matches any substring; the second part must match the *same* substring and be surrounded by word boundaries. Try:

```
grep -E '(.+) \<\1\>' corpora/ts-text-english.txt -m 20 -o
```

Question 5 *Which word repetitions do you find in Tom Sawyer? Are there words repeated thrice?*

[1] Type `export LC_ALL='C'` if you get an error message when opening the Dutch text.

6 Replacement and piping

`grep` is a program for *finding* lines in a file that match a regular expression. The *shell* we work with also comes with a standard program for *replacing* matches with something else: `sed`.

`sed` is most often used in combination with another trick that the shell offers: *piping*. Piping allows us to create a pipeline of commands, where the output of one tool is sent directly to the next with the vertical bar symbol: `|`.

For instance, we can take the output of a command `echo 'hello world'`, which would normally print "hello world" to the screen, and pipe it to `sed` with the instructions to replace "hello" with "goodbye":

```
echo 'hello world' | sed 's/hello/goodbye/g'
```

Try! The instructions to `sed` have the form `s/`expression1`/`expression2`/g`, where the `s` means 'substitute', expression1 is a regular expression to be replaced by expression2, and the `g` means 'do it for every match of expression1 on a line'.

We can use piping, `sed` and `grep` in all kinds of combinations. For instance, while `grep` can only count the number of matching *lines* (it ignores multiple matches on a line), we can use `sed` to put every word on a separate line (by replacing spaces with hard returns: `\n`) so that we can use `grep` to count the number of matching *words*. To count the number of occurences of "the" in Tom Sawyer, we thus use the command:

```
less corpora/ts-text-english.txt | sed 's/ /\n/g' | grep -E 'the|THE|The' -c
```

Question 6 *Calculate the frequency of the most frequent words in Tom Sawyer:* the, and, a, to *and* of. *Also calculate the number of lines that contain each of these words (i.e., without using* `sed`*).*

In combination with built-in shell commands such as `sort` (which sorts), `uniq -c` (which counts unique lines in a file) and `head -`n (which shows the first n lines of a given file), `sed` and `grep` can do surprisingly complex jobs for us. For instance, to get a table of the 20 most frequent words and their frequencies in Tom Sawyer, all we need to do is type (all on one line):

```
less corpora/ts-text-english.txt | sed 's/ /\n/g' |
   grep -E '[A-Za-z]' | sort | uniq -c | sort -r -g | head -20
```

6.1 Verb frequencies (optional)

There are now also many large corpora available where the sentences are annotated with important syntactic information, for instance about the categories of words. An example is the file `corpora/BNCmini.txt`, which is extracted from the much larger *British National Corpus*. Have a look at it using the command `less` (and quit it with `q`).

We can use corpora like this to start addressing some interesting linguistic questions. For instance, it has often been observed that there is a relation between frequency and (ir)regularity. Irregular verbs are almost always very frequent.

The following steps will calculate, fully automatically, the top 30 most frequent past tense verbs from this corpus:

Take a large tagged corpus	less BNCmini.txt
Put every word on separate line	sed 's/</\n/g'
Select all past/perfect tense verbs	grep -o 'VV[DN]>[a-zA-Z]\+'
Remove the tag	sed 's/VV[ND]>//'
Sort them alphabetically	sort
Calculate frequency of each verb	uniq -c
Sort the list on frequency	sort -g -r
Output the top 30	head -30

The command is (all on one line):

```
less ~/research/corpora/BNCmini.txt |
   grep 'VV[DN]>[a-zA-Z]\+ ' -o |
   sed 's/VV[ND]>//' |
   sort |
   uniq -c |
   sort -g -r -k 1 |
   head -30
```

Question 7 *Run this command to find the top 30 most frequent verbs in the corpus. Which of these are irregular?*

7 Optional: Authorship attribution

In this final part of the tutorial, you will carry out a cluster analysis on data extracted from 10 electronic books. For 9 of these books we know the author; your task is to determine the most likely author of the 10th book, and to consider how useful and reliable the used clustering technique is for solving that puzzle.

7.1 Getting started

You need a unix/linux-like terminal (as we used above) and the program R. Download `dighum-authorattribution.zip` from `http://www.illc.uva.nl/laco/clas/dighum14/`, move it to your working directory and unpack the file. Have a look at the 10 `.txt` files which are classic pieces of British literature.

7.2 Gathering statistics

You will need to compute statistics on the 10 files for your cluster analysis. To make things really easy for you, all commands are put together in a little script `frequent-words-script`. Have a look at this file with `less`. The first step is to make a list of the 100 most frequent words. The second step is to count the frequencies of these 100 words in fragments of each of the 10 texts.

Question 8 *How often does this script run the* `grep` *program?*

To run the script you need to make it *executable*. You do that with the command
`chmod u+x frequent-words-script`. After that, you can run the script with `./frequent-words-script`

Try! If things work well you should see a message about what the script is doing. When it's finished, which might take a minute, there should be a file `top100matrix.csv` in your folder. Check!

Question 9 *Which of the 10 books have male and which have female protagonists? How can you tell from the big matrix with numbers you just created?*

7.3 Clustering

Now start up `R`. R comes with clustering tools preinstalled, which you can make available by typing `library(cluster)`. Change R's working directory (using the option under File) to the folder where the `top100matrix.csv` file you just created is. Load the file into R by typing:

```
data = read.csv("top100matrix.csv",row.names=1)
```

Have a look at the data by typing `data`.

Now perform a cluster analysis by typing:

```
analysis = agnes(data,FALSE,stan=TRUE)
```

Finally, plot the dendrogram of the hierarchical cluster analysis by typing `plot(analysis)`, clicking on the graphical window and hitting `ENTER` twice.

Question 10 *Describe the results from the cluster analysis. Who is the most likely author of the unknown text? How confident are you of that conclusion, and why?*