Practical: Using Linux commands to analyze text documents & introduction to UNICODE

Inspired from "Unix for Poets" from Kenneth Ward Church, IBM Research, kwchurch (AT) us.ibm.com

Program

1. Count words in text
2. Order a list of words by
   - order "ASCII"
   - alphabetical order
3. Calculate occurrence and co-occurrence statistics
4. Extract useful information from a dictionary
5. Transform text
6. Calculate association measures between words

Tools

- `cat/head/tail`: display (the whole / the beginning / the end) of a file on the screen
- `less`: display a file with scrolling
- `grep`: search for a pattern (regular expression)
- `sort`: sort the lines
- `uniq -c`: eliminate or count duplicates
- `tr`: transpose or eliminate characters
- `wc`: count lines, words and characters
- `sed`: edit (replace, delete) a string with regular expressions
- `awk`: language for examining and processing patterns
- `cut`: extract columns
- `paste`: "paste" columns
- `join`: equivalent to SQL JOIN
- Web browser: Mozilla Firefox, Google Chrome or Chromium
- Hexedit text editor
  - Ubuntu et Debian-like: `sudo apt-get install hexedit`
  - Others: Download the executable
- Conversion of character encodings: `iconv`

Suggestion: always remember to read the tool's `man` page if you have any doubts about the tool's parameters and functionality. For example, the `man wc` command indicates how you can count only lines (`-l`) or only words (`-w`) in a file.

Input / output redirection

By default, most tools read the input (`stdin`) from the keyboard and write the output (`stdout`) to the screen. You can redirect inputs and outputs to files using the following commands:

<	Read from an input file
>	Write to an output file
|	Write the output of the previous command to the input of the next command (pipe)

Data

File for exercises: RADIOS.txt or RADIOS.txt.UTF-8 depending on the operating system available here. This file contains transcripts of radio recordings: Franceinter, RFI,...

X
For each exercise, in addition to answering the questions, also you need to explain the sequence of commands used and show an extract of the result using the commands `head` and/or `less`

Exercises

Exercise 0: Transform all RADIOS.txt text into capital letters

- **Q1**: Find another equivalent command
- **Q2**: Find an order that takes accented characters into account

Exercise 1: Find the sequence of instructions that allows you to count words in a text

- Input: RADIOS.txt text file
- Output: list of words with their number of appearances in the text (RADIOS.hist)

Help: use `tr`, `sort` and `uniq`, think of "piping" (|) the instructions

- **Q1**: What word appears exactly 1,732 times in this text?
- **Q2**: How many times does the word "orange" appear in this text?

Exercise 2: Sort the words (`sort`)

See page `man` of `fate`

Examples

`sort -d`	dictionary order
`sort -f`	ignore upper / lower case
`sort -n`	numerical order
`sort -nr`	reverse numerical order

`sort -k 1`	start at field 1 (the first is field 0)
`sort -k 0.50`	start at the 50th character
`sort -k 1.5`	start at 5th character of the field 1

- **Q1**: Sort the words of RADIO.txt by frequency of appearance
- **Q2**: Sort the words of RADIO.txt in alphabetical order
- **Q3**: Sort the words of RADIO.txt in "rhymic" order (example, put together all the words which end with "-ment". **Help**: use the command unix `rev`

Exercise 3: Co-occurrences of words or bigrams

Find and count all the bigrams text RADIOS.txt

- Input: RADIOS.txt text file
- Output: statistics file where each line is of the form "word1 word2 nb"

Do the same for n-grams (n = 2,3,4).

- **Q1**: How many times does the 2-gram "it is" appear in this text?
- **Q2**: How many separate 3-grams are found in this text?
- **Q3**: What is the most common 4-gram?

Help: use the `tail` and `paste` commands

Exercise 4: filters (`grep`)

See `man` page of `grep`.

Examples

`grep '[A-Z]'`	lines containing a capital letter
`grep '^[A-Z]'`	lines starting with a capital letter
`grep '[A-Z]$'`	lines ending with a capital letter
`grep '^[A-Z]*$'`	fully uppercase lines
`grep "[aeiouAEIOU]"`	lines containing a vowel
`grep "^[aeiouAEIOU]"`	lines starting with a vowel
`grep "[aeiouAEIOU]$"`	lines ending with a vowel
`grep -i '[aeiou]'`	lines containing a vowel
`grep -i '^[aeiou]'`	lines starting with a vowel
`grep -i '[aeiou]$'`	lines ending with a vowel
`grep -i '^[^aeiou]'`	lines starting with a non-vowel
`grep -i '[^aeiou]$'`	lines ending with a non-vowel
`grep -i '[aeiou].*[aeiou]'`	lines with at least two vowels
`grep -i '^[^aeiou]*[aeiou][^aeiou]*$'`	lines with exactly one vowel

With regular expressions

`a`	the letter "a"
`[a-z]`	a tiny letter
`[A-Z]`	a capital letter
`[0-9]`	a number
`[0123456789]`	a number
`[aeiouAEIOU]`	a vowel
`[^aeiouAEIOU]`	anything but a vowel
`.`	a character
`^`	start of line
`$`	end of line
`x*`	"x" repeated 0 or more times
`x+`	"x" repeated 1 or more times
`x|y`	"x" or "y" (only `egrep`)
`(x)`	impose rule previously (only `egrep`)

- **Q1**: How many has the words of 9 letters in RADIOS.txt?
- **Q2**: Are Has -it words without vowels in RADIOS.txt

Exercise 5: `Awk` language

`awk` is a language whose syntax is similar to C, and which allows operations to be performed on fields in a file where each line is of the type "field1 field2 field3 field4..."

Example

- `awk '{print $1}'` selects the first field in a file, equivalent to `cut -f 1`
- `awk '$1 > 100 { print $0}' RADIOS.hist` displays the words whose number of occurrences is greater than 100 in RADIOS.txt

It is possible to write the program in a file and after calling it, for example, by typing `SelectPremierChamp.awk <file`

The available predicates are:>, <, > =, <=, ==, !=, &&, ||

- **Q1:** Find the bi-grams that appear exactly 13 times.
- **Q2:** Find the palindromes in RADIOS.txt (words written in the same way from right to left or from left to right).

Exercise 6: Replacement with `sed`

See page `man` of `sed`.

The `sed` tool allows you to replace text using regular expressions. Thus, the command `sed 's/exp1/exp2/[options]'` will replace (`s`) the expression `exp1` by the expression `exp2`. The option is often the letter `g` to say that all occurrences on the same line should be replaced. For example:

- `sed 's/[éèêë]/e/g'` replaces all accented letters "e" with an unaccented letter
- `sed 's/ \([^]*\)ation / \U\1ATION/g'` transforms the words which end with the suffix "ation" into capital letters, for example, "habitation" becomes "HABITATION"
- **Q1:** Add an end point to each line and put the first letter in capital letters.
- **Q2:** Replace any occurrence of two consecutive identical words with a single occurrence of that word, for example, "de de" becomes "de".

Exercise 7: Union of files with `join`

A measure of association is a numerical value which estimates the degree of association between two words. This measure can be used to automatically detect "collocations", ie sequences of words which appear more often than one would expect by chance. A measure of association often used is the measure *PMI* (pointwise mutual information), which is defined as follows for a bigram formed by the words w_1 and w_2:

$$\text{PMI} = c(w_1\ w_2)\ .\ \log [\ c(w_1\ w_2) / E(w_1\ w_2)\ ]$$

Where $c\ (w_1, w_2)$ is the number of occurrences of the bigram $w_1\ w_2$, as calculated in exercise 3, and $E\ (w_1, w_2)$ is the expected number of occurrences of this bigram defined as:

$$E(w_1\ w_2) = \frac{c(w_1)\ c(w_2)}{N}$$

Where $c\ (w_1)$ is the number of occurrences of the first word, $c\ (w_2)$ is the number of occurrences of the second word and N is the total number of words in this text.

For each bigram in the text, calculate its *PMI* association value according to the given formula. Each line of the output file must have the form "w_1 w_2 $c(w_1,w_2)$. $c(w_1)$ $c(w_2)$ PMI"

- **Q1:** What are the five most strongly associated bigrams according to this measure?
- **Q2:** What are the five most strongly associated bigrams that appear less than 1000 times in the text?

Help: Use the `join` command to combine the information from the files in Exercises 2 and 3, but be careful to sort the files by the "junction" field (s). To avoid warnings, before any `sort` and `join` command, you must modify the `LC_ALL=C` variable, for example: `LC_ALL=C sort RADIOS.hist`

Exercise 8: observing the encodings
Open the files `testFR.txt.UTF-8`, `testRU.txt`, `testGK.txt` and `testAR.txt` with a web

browser and modify the character encoding for display (UTF-8, UTF-16, Western ISO-8859 -1, Arabic Windows-1256, etc.):
The files can be found here.

- Mozilla Firefox: Menu `View > Character Encoding > More Encodings`
- Chrome et Chromium: Bouton `Customize > Tools > Encoding`

Questions:

- **Q1:** Why do some files appear as strange symbols instead of text?
- **Q2:** How do you think the browser can detect the character encoding of the file? `Is` he still `getting` there (testing with `testAR.win`)?
- **Q3:** Why do some texts display correctly even when we choose a character encoding different from that in which the text is saved (for example, try to display `testFR.txt` with the encoding " Turkish Windows-1254")?

Open the `testFR.txt`, `testRU.txt`, `testGK.txt` and `testAR.txt files` with office software such as MS-Word or Libre Office. Same questions.

Exercise 9: analysis of encodings
Compare the different Unicode layers for the `testFR.txt.UTF-8` file, in particular:

- Character or glyph displayed
- Point code (find the relationship with the character in Unicode tables)
- Representation on disk according to the encoding format (`man UTF-8`, use `hexedit` to see the representation in hexadecimal)

Questions:

- **Q1:** How many bytes are needed to represent the word " below " in UTF- 8? What is the representation of this word in hexadecimal?
- **Q2:** In UTF-8, what are the glyphs represented by the hexadecimal sequence "47 72 C3 BC C3 9F 65"? How many bytes are required to represent each glyph?
  Help: To find out the glyph from the point code, type `/ usr / bin / printf "\ uXXXX \ n"` replacing XXXX with the point code of the glyph

Exercise 10: encoding and place occupied
Compare the files `testFR.txt.UTF-8` and `testFR.txt.iso8859-1`

- **Q1:** What is the most efficient representation in terms of bytes used?
- **Q2:** What is the sequence of bytes used in each encoding to represent the first character?

Exercise 11: conversion to UTF-8

- **Q1:** Convert the testAR.win file to UTF-8 with the Linux `iconv` command. What can we say about the display in `vim`, once the file has been converted?

Help: for Arabic, the character encoding of the file is in windows format, called `cp1256`.

Exercise 12: conversion from UTF-8

Convert the file `testFR.txt.UTF-8` to UTF-16 with `iconv`. Visualize the change with `hexedit`.

- **Q1:** What are the first two bytes in the file converted to UTF-16?
- **Q2:** What is the correspondence between Unicode point codes and their representation in UTF-16?
- **Q3:** Now try to convert to UTF-16BE. What is the difference between this encoding and UTF-16?