School of Computer Science

COMP30640

Lab 8
Regular Expressions

Teaching Assistant:	Thomas Laurent
Coordinator:	Anthony Ventresque
Date:	Friday 2nd November, 2018
Total Number of Pages:	2

1 Crossword Helper

In this exercise we will manipulate the `/usr/share/dict/words` file which is the basic vocabulary used in Unix-like systems (including Linux and MacOS X). The objective is to simulate a crossword helper - similar to the little devices you find in shops or the app you can download for your smartphones.

Write a script that asks the users to input an incomplete word - mixing letters and '?' characters - and gives all the possible words taken from the file that match it (replacing '?' by 1 letter). Example:

```
$> ./crosshelper.sh abbreviat???
abbreviating
abbreviation
```

Solution

```
#!/bin/bash

if [ $# -lt 1 ] ; then
echo "This script requires one parameter" >&2
echo "$0 incomplete_word" >&2
exit 1
fi

pattern=`echo $1 | sed -e s/?/./g` #Replace the ?s in the argument with .s
grep -w -i "$pattern$" /usr/share/dict/words #Search for lines with this pattern.
    #The last $ prevents pattern "test" from matching test's
```

2 Text Processing

Download a book from the Gutenberg Project, for instance this one: *James Joyce's Ulysses* (maybe something shorter if you do not want to wait too long for your scripts to execute ;-)).

1. parse the text to remove the punctuation and get one word per line. Save this new version of the document in a file. You can do this by using the `tr` function and replacing punctuation, spaces and carriage returns by new lines. You might get blank lines in your new file, `tr` has an option to deal with this.

   Solution We use `tr` to replace every punctuation sign, horizontal space and carriage return with a new line. We use the -s option to not have blank lines (squeeze repeated new lines).

   ```
   $> cat 4300-0.txt | tr -s '[:punct:][:blank:]\r' '\n' > list1
   ```

2. list the 20 most frequent words from your text

   Solution

   ```
   $> cat list1 | sort -f | uniq -c -i | sort -n | tail -n 20
   ```

3. remove the stop words (i.e., the words with little or no real meaning); use a list of stop words from the web (e.g., one of *these lists*).

   Solution

```
#!/bin/bash
# not efficient!

if [ $# -lt 2 ] ; then
echo "This script requires two parameters" >&2
echo "$0 original_file target_file" >&2
exit 1
fi
if [ ! -e "$1" ]; then
echo "The supplied parameter is not a file" >&2
exit 2
fi
while read word; do
        if grep -i -q -w "$word" terrier.txt; then
                echo $word in terrier
        else
            if [ ! -e $2 ]; then
                echo $word > $2
            else
                echo $word >> $2
            fi
        fi
done < "$1"
```

4. list the 20 most frequent words excluding stop words

   Solution

```
$> cat list2 | sort -f | uniq -c -i | sort -n | tail -n 20
```