grepWin: Regular expression search within large text corpora

Description

This excellent open-source tool allows parallel search within a large number of documents using regular expressions. It is very useful if you look for specific information withing, say, a large number of PDF, HTML, and/or Word documents. By using (Perl) regular expressions you can use highly specific Boolean search query combinations…

github.com/stefankueng/grepWin/releases/tag/1.9.0

```
// e1 is a case sensitive Perl regular expression:
// since Perl is the default option there's no need to explicitly specify the
boost::regex e1(my_expression);
// e2 a case insensitive Perl regular expression:
boost::regex e2(my_expression, boost::regex::perl|boost::regex::icase);
```

www.boost.org/doc/libs/1_57_0/libs/regex/doc/html/boost_regex/syntax/perl_syntax.html

“grep” is a command-line utility for searching text data sets for lines that match a regular expression. Its name comes from the ed command g/re/p, which has the same effect: doing a global search with the regular expression and printing all matching lines. Grep was originally developed for the Unix operating system, but later available for all Unix-like systems. More at Wikipedia

---

A regular expression (regex or regexp for short) is a special text string for describing a search pattern. You can think of regular expressions as wildcards on steroids. You are probably familiar with wildcard

notations such as *.txt to find all text files in a file manager. The regex equivalent is `^.*\.txt$`.

But you can do much more with regular expressions. You could use the regular expression `\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}\b` to search for an email address. *Any* email address, to be exact. A very similar regular expression (replace the first `\b` with `^` and the last one with `$`) can be used by a programmer to check whether the user entered a properly formatted email address. In just one line of code, whether that code is written in Perl, PHP, Java, a .NET language, or a multitude of other languages.

www.regular-expressions.info/

search basics

. (dot)
: a dot matches any character. Searching for `t.t` will match `tat` as well as `tut`.

+
: matches the previous expression one or more times, but at least once. Searching for `spel+ing` will find all words like `speling` or `spelling` but not `speing` since the `l` must be matched at least once.

*
: matches the previous expression zero or more times. Searching for `spel*ing` will find all words like `speling` or `spelling` and also `speing` since the `l` can be matched zero times, which means it doesn't have to be there.

\
: the backslash escapes special characters that would otherwise be treated specially. Searching for a double dot in your text with `..` would not work since the dot matches any character. To search for a double dot you have to escape the dot chars like this: `\.\.`.

\Q..\E
: in case you need to search for a literal string that has a lot of special characters in it, you can use the `\Q..\E` sequence. Searching for `*.*` would match everything unless you escape every single char like this: `\*\.\*`. For such search strings it's easier to just put them inside the `\Q..\E` sequence like this: `\Q*.*\E`.

[]
: With square brackets you can specify so called character classes. Such a class matches all chars that are specified between the brackets. Searching for `[-+0-9]+` will find any string that contains the chars '-', '+' and all chars between 0 and 9, but no other chars. It will match `-123`, `+123` or `123`, but not `testword`. There are a few default character classes defined so you don't have to create one yourself. You can find a list of those classes here. The most used ones are `\d` which matches all digits, `\w` which matches all word chars and `\s` which matches all whitespace chars.

^, $

the caret matches the beginning of a line, and the string char `$` matches the end of a line. Searching for `^title$` will only find lines that only consist of the word `title`, but no places where the word `title` is inside a line. Searching for `^//` will find all lines that start with two slashes, but not lines where two slashes are not at the very beginning of a line. Searching for `goodbye\.$` will find lines that end with `goodbye.`, but not if `goodbye.` is somewhere inside a line.

\b

`\b` matches word boundaries. Searching for `\bword\b` finds `word`, but not `subwords` or `words`.

()

parenthesis pairs define a group. Grouping is useful for more advanced regex searching, but also for use when replacing text. Each group that matches part of the full matching string can be referenced later in the replace string.

|

The `|` char is used as an OR operator. Searching for `cat|dog` will match either `cat` or `dog`. Note that the OR operator uses everything left and right of the operator. If you want to limit the reach of the operator, you have to use brackets to group them. Searching for `(cat|dog)food` finds `catfood` and `dogfood`.
tools.stefankueng.com/regexhelp.html

Category

1. Software

Date Created
November 2018
Author
web45