LING 410X: Language as Data

Semester: Spring '18

Instructor: Sowmya Vajjala

Iowa State University, USA

18 January 2018

Class outline

- Some simple text processing functions in R: regular expressions
- Practice: Counting word frequencies in a text

Simple text processing with regular expressions - Overview

What are regular expressions?

- They are a special strings of text used to search and extract matching patterns from a large piece of text.
- Example: Let us say I want to extract all money expressions from a news article. I will create a regular expression to cover all patterns of describing money (25.99$, 25$, 25 Dollars etc) and use that to collect a list of all money expressions.

What are regular expressions?

- They are a special strings of text used to search and extract matching patterns from a large piece of text.
- Example: Let us say I want to extract all money expressions from a news article. I will create a regular expression to cover all patterns of describing money (25.99$, 25$, 25 Dollars etc) and use that to collect a list of all money expressions.
- Example 2: I may want to collect all phone numbers or email addresses on English department website.

What are regular expressions?

- They are a special strings of text used to search and extract matching patterns from a large piece of text.
- Example: Let us say I want to extract all money expressions from a news article. I will create a regular expression to cover all patterns of describing money (25.99$, 25$, 25 Dollars etc) and use that to collect a list of all money expressions.
- Example 2: I may want to collect all phone numbers or email addresses on English department website.
- There is a limit to what kind of things we can search for using regular expressions.
- There is a specific syntax to using them.
- Yet, they are one of the most basic and powerful text processing tools.

Why/When do we need them?

- As mentioned in the examples, for pattern extraction from text.
- If you know the kind of patterns you are looking for, go for regular expressions.
- If you know there are patterns but do not know what they are or how to get them, go to text classification, topic modeling and such advanced stuff.

Why/When do we need them?

- As mentioned in the examples, for pattern extraction from text.
- If you know the kind of patterns you are looking for, go for regular expressions.
- If you know there are patterns but do not know what they are or how to get them, go to text classification, topic modeling and such advanced stuff.
- They have a very restricted syntax, so once you know what your pattern is, the only task left is to use the syntax to match your pattern.

Why/When do we need them?

- As mentioned in the examples, for pattern extraction from text.
- If you know the kind of patterns you are looking for, go for regular expressions.
- If you know there are patterns but do not know what they are or how to get them, go to text classification, topic modeling and such advanced stuff.
- They have a very restricted syntax, so once you know what your pattern is, the only task left is to use the syntax to match your pattern.
- in R: grep() and grepl() functions.
- They are already implemented for your MacOS terminal, for any other programming language you may end up using at some point in future.

Basic syntax and vocabulary of regular expressions

Let us take this sentence: "This is a sentence to demonstrate regex usage."

- The expression "a" just matches all occurrences of "a" in this.

Basic syntax and vocabulary of regular expressions

Let us take this sentence: "This is a sentence to demonstrate regex usage."

- The expression "a" just matches all occurrences of "a" in this.
- [*abc*] matches a or b or c.

Basic syntax and vocabulary of regular expressions

Let us take this sentence: "This is a sentence to demonstrate regex usage."

- The expression "a" just matches all occurrences of "a" in this.
- [*abc*] matches a or b or c.
- "." (period) matches everything except newline. If you want to match a period, "\\ \\ ."

Basic syntax and vocabulary of regular expressions

Let us take this sentence: "This is a sentence to demonstrate regex usage."

- The expression "a" just matches all occurrences of "a" in this.
- $[abc]$ matches a or b or c.
- "." (period) matches everything except newline. If you want to match a period, "\\."
- "s.e" matches anything between s and e. (in the above example, "sentence").

Basic syntax and vocabulary of regular expressions

Let us take this sentence: "This is a sentence to demonstrate regex usage."

- The expression "a" just matches all occurrences of "a" in this.
- [*abc*] matches a or b or c.
- "." (period) matches everything except newline. If you want to match a period, "\ \ ."
- "s.e" matches anything between s and e. (in the above example, "sentence").
- "a+" matches wherever a occurs once or more times consecutively.
- "a*" matches zero or more occurrences of "a".

Basic syntax and vocabulary of regular expressions

Let us take this sentence: "This is a sentence to demonstrate regex usage."

- The expression "a" just matches all occurrences of "a" in this.
- [*abc*] matches a or b or c.
- "." (period) matches everything except newline. If you want to match a period, "\\ \\ ."
- "s.e" matches anything between s and e. (in the above example, "sentence").
- "a+" matches wherever a occurs once or more times consecutively.
- "a*" matches zero or more occurrences of "a".
- "\\ \\ *d*" matches one digit. """\\ \\ *D*" matches a non-digit.

Basic syntax and vocabulary of regular expressions

- $[a - z]$ matches any lower case letter.
- "$a|b$" matches "a" or "b".
- "^a" matches all strings in a vector that start with "a"
- "$a\$$" matches all strings in a vector that end with "a"
- $a\{2, 4\}$ matches if a occurs consecutively 2 to 4 times (aa, aaa, aaaa)

Basic syntax and vocabulary of regular expressions

- $[a-z]$ matches any lower case letter.
- "$a|b$" matches "a" or "b".
- "^a" matches all strings in a vector that start with "a"
- "$a\$$" matches all strings in a vector that end with "a"
- $a\{2,4\}$ matches if a occurs consecutively 2 to 4 times (aa, aaa, aaaa)

..... many more. Look at the tutorial document for Assignment 1 for more. For even more, search online. Lot of tutorials available for general regex and R specific regex usage.

Using Regex in R

```
> library(stringr)
> sentence = "This is a sentence to demonstrate regex usage
 and this contains a word Mississippi and another word Missouri."
> grepl("a",sentence) #returns TRUE if there is that pattern "a" in string sentence.
> grepl("s{2,3}",sentence) #What does this return?
> grepl("s{3}",sentence)
> library(stringr)
> str_count(sentence,"a") #returns the number of times pattern "a" occurs in the sentence.
> str_count(sentence,"s{2,3}") #Returns?
> sub("a","b",sentence) #substitutes first occurrence of a with b.
> gsub("a","b",sentence) #substitutes all occurrences of a with b.
> sentence #note that all this will not change sentence. Why?
```

Other useful functions: https://cran.r-project.org/web/packages/stringr/vignettes/stringr.html

Practice Exercise

Recap: R code for counting word frequencies

```
setwd("~/Dropbox/ClassroomSlides-BothCourses/LING410X/")
english <- scan("DollsHouse-Eng.txt", what = "character", sep = "\n")
english.start <- which(english == "DRAMATIS PERSONAE")
english.end <- which(english ==
      "(The sound of a door shutting is heard from below.)")
actual_english <- english[english.start:english.end]
actual_english_string <- paste(actual_english, collapse = " ")
english_lower <- tolower(actual_english_string)
english_words <- strsplit(english_lower, "\\W+")
sorted_freqs_english <- sort(table(english_words), decreasing = TRUE)
plot(sorted_freqs_english[1:10], type="b")
```

Today's handson exercise

- Take the word frequency counter code from Tuesday, and make it work on any given gutenberg.org book (in .txt format).
- One useful function: startsWith("Language as Data", "Lan") is useful to check if a given string/line of words in the first argument starts with the second argument.
- Note: You have to only change three lines from what I used on Tuesday.
- Note 2: You perhaps should look at 4-5 gutenberg text files to be able to "change" these lines.

Solution

- I am uploading the enhanced version of the tutorial file from previous class. That contains your solution.
- In short it is this:

```
english <- scan("DollsHouse-Eng.txt", what = "character", sep = "\n")
meta.top.end <- which(startsWith(english,"*** START OF THIS PROJECT GUTENBERG EBOOK"))
meta.bottom.start <- which(startsWith(english, "End of the Project Gutenberg EBook"))
actual_english <- english[meta.top.end+1:meta.bottom.start+1]
actual_english_string <- paste(actual_english, collapse = " ")
english_lower <- tolower(actual_english_string)
english_words <- strsplit(english_lower, "\\W+")
sorted_freqs_english <- sort(table(english_words), decreasing = TRUE)
plot(sorted_freqs_english[1:10], type="b")
```

Post your response in the discussion forum for today

- In Doll's House, how many times did: "you know" appear?
- How many times did numbers appear in the text (99712 is to be counted as 1 number. Not 5)?

Next Week

- processing different file formats (e.g., Doc files, Webpages, PDF files, news websites, tweets etc)
- Optional Reading: Chapter 10 in the textbook (you don't have to run those code examples)
- We will get back to processing of text files in the week after this (once you appreciate the complex issues that arise because we have so many different formats of textual data!)
- Note: I will talk about how to store these programs in the next 1-2 weeks.