Digital Methods: Learning Journal

Christoffer Mondrup Kramer

Autumn 2020

How to structure the journal:

1 Write current date

1.1 Write the assignment I was dealing with and my own thoughts

Write the time in Bold, if you remember it: Followed by text about my own thoughts

1.2 Actions and Results

- **Action:** Write your action

```
Write code here
```

- **Result:** write your result

1.3 Final thoughts

Write the time in bold, if you know it: Write your final thoughts.

2 11-09-2020

2.1 Regex (Regular expressions): Stop-word list from Voyant! to R

9.30: I'm supposed to make a list with stop-words for Voyant! into a list for R. The R list contains words which are enclosed in quotation marks and separated by a comma and a white-space. For example: "an", "og". In the Voyant! list, all words are separated by a line-break and nothing else such as:
Han
Hun
Den

2.2 Actions and Results

- **Action:** I started out by trying to just match the words. I did this with the following expression:

  ```
  ^\w+
  ```

  The circumflex indicates that the following expression should appear first in the string. That way I should not match any line-breaks. The w is used to match any word letter (including 0-9 and underscore). The + indicates that these character should appear one or more times.

- **Result:** This worked partially. However, it didn't match words containing æøå.

- **Action:** I, therefore, needed to account for Danish letters (æ-ø), which I did with the following expression:

  ```
  ^[\wæøå]+
  ```

  The square brackets was added to indicate, that I wanted ANY letter inside them. The æøå were added so I could match danish letters. The + was moved outside of the square brackets to indicate, that any of the letters inside of the brackets should appear at least one time.

- **Result:** I thought that this made it possible to match all words in the list. But, it did not take case into account. It did not appear to be a problem in this particular list since all words were written in non-capital letters. However, if another list contained capital letters, it would create problems.

- **Action:** I therefore decided to make the expression case-insensitive. The list did not contain capital letters, so I wrote two test words ("Ælling" and "legoLand") and entered the following regex:

```
^(?i)[\wæøå]+
```

The parenthesis, the ? and the i was something Vojtech taught us yesterday. It makes the rest of the regex case-insensitive.

- **Result:** Ælling was not being matched. I suspected, that this was because of the letter æ.
- **Action:** I read about w and realised that it also includes capital letters from A-Z. Therefore, the problem had to be, that æøå was not being parsed as letters, but as special characters. I, therefore, entered the following regex:

```
^[\wæøåÆØÅ]+
```

I removed the code for case-insensitive, since the w took care of it for the letters A-Z. I then used ÆØÅ and æøå in the regex. This should ensure, that capital ÆØÅ would be matched as well.

- **Result:** While scrolling through the results, I noticed that certain characters, such as apostrophes and periods, were not being matched.
- **Action:** I needed to account for more special characters, so I added a period to the regex inside of the brackets. This should ensure, that I would also match special characters except for white-spaces and line-breaks:

```
^[\wæøåÆØÅ.]+
```

- **Result:** This did not work. Nothing changed.
- **Action:** I read about the period, and apparently it will literally only match periods, when it is in square brackets. So I moved it outside the brackets.

```
^[\wæøåÆØÅ].+
```

- **Result:** It actually kinda worked. But now the regex were not matching digits:
- **Action:** I realised that the + needs to be after the square brackets, and that the period needs to be optional. If the + is after the period it means, that the pattern needs to contain one or more letter characters (A-Z, a-z, 0-9 and underscore) or ÆØÅ or æøå AND one or more of any character except for white-space and line-breaks. In short, it meant that all matches needed to contain at least two characters. That is why I were not matching digits with one character. Therefore, the period needed to be optional. I, therefore, typed in:

```
^[\wæøåÆØÅ]+.?
```

The ? indicates that any character after the brackets is optional.

- **Result:** It did not work. I were only matching the first special character in these words.
- **Action**: I read about the ? and apparently it means between zero and one time. I, therefore, needed to use the asterisk:

```
^[\wæøåÆØÅ]+.*
```

The asterisk means that any character after the square brackets will be matched if it is used between zero and unlimited times (effectively it means that these characters are optional).

- **Result:** It worked. But maybe I should include it at the begging as well. That way I can match words which starts with a special character.
- **Action**: I added .* after the circumflex.

```
^.*[\wæøåÆØÅ]+.*
```

I tested this expression by writing YOLO with a hashtag in front of it.

- **Result:** It worked, I got a match.
- **Action:** Now I needed to substitute the line-breaks. Therefore, I made a group, so I could create variables:

```
(^.*[\wæøåÆØÅ]+.*)
```

The parenthesis indicates, that this is a group, which can be used later. I tried to put quotation marks around the variable and follow it with a comma and a white-space:

```
"$1",
```

- **Result:** This worked partially, but it did not remove the line-break:
- **Action:** I tried to use "b" to mark word boundaries, since I suspected that I was matching line-breaks:

```
(\b.*[\wæøåÆØÅ]+.*\b)
```

The b indicates a word boundary (such as a white-space or a line-break):

- **Result:** This did not change anything, and I were not matching words which started with ÆØÅ. So I removed the boundary.

- **Action:** I then tried to match and group line-breaks separately. However the line-break needed to be optional. Otherwise it would not match the last word on the list, which was not followed by a line-break:

  ```
  (^.*[\wæøåÆØÅ]+.*)(\r*)
  ```

- **Result:** This worked. It is now a stop-word list for R. It means the following: Find a pattern where THE FIRST CHARACTERS, WHICH CAN BE ANYTHING EXCEPT FOR WHITE-SPACES AND LINE-BREAKS, APPEARS BETWEEN ZERO AND UNLIMITED TIMES. Followed by ONE OR MORE WORD LETTERS FROM A-Z, a-z, 0-9, ÆØÅ, æøå. Followed by ZERO OR MORE CHARACTERS, WHICH CAN BE ANYTHING EXCEPT FOR WHITE-SPACES OR LINE-BREAKS. Followed by ZERO OR MORE LINE-BREAKS.
  However, after writing this explanation, I realised that I could just use the following code, since my code is way too over-engineered:

  ```
  (.+)(\r*)
  ```

  The period captures all characters regardless of their capitalization including æøå, digits and special characters. It just means the following: Find a pattern WHICH CONTAINS ONE OR MORE CHARACTERS EXCEPT FOR WHITE-SPACES AND LINE-BREAKS followed by ZERO OR MORE LINE-BREAKS.

2.3 Final thoughts

It was actually quite fun, but also very frustrating, to work with regexes. Moreover, I realised, that it is very helpfull to write exactly what the regex is matching, since it made me realize, that there was a way more elegant solution.

3 11-09-2020

3.1 Regex (Regular expressions): Stop-word list from R to Voyant!

13.30: Same task as before but in reverse.

3.2 Actions and Results

- **Action:** I started with my previous and over-engineered regex:

  ```
  (^.*[\wæøåÆØÅ]+.*)(\r)
  ```

- **Result:** This did not work. The whole list was displayed as one match rather than multiple matches.

- **Action:** I tried to but a word boundary around the regex

  `(\b.*[\wæøåÆØÅ]+.*\b)(\r)`

- **Result:** No changes.
- **Action:** I suspected that this was because of the periods in the regex. I, therefore, removed them.

  `(\b[\wæøåÆØÅ]+\b)(\r)`

- **Result:** Now I was not matching words with special characters such as umlauts or periods. But I could not use the period since it would match the quotation marks and commas. Moreover, I could not specify each special character, since I did not know which where used.
- **Action:** After a break I realized that I should be trying to match the commas, the spaces and the quotation marks, since they are the ones I need to replace. So I made the following expression:

  `(", )`

  This regex matches a string which contains a quotation mark followed by a comma and then a white-space. This is how all stop-words on the R list are separated.

- **Result:** This almost worked, however, the quotation mark at the start of each word was not captured.
- **Action:** I, therefore, added another quotation mark after the white-space, that way I should be able to match the commas at the start of each word as well:

  `(", ")`

- **Result**: The first and last quotation mark in the list were not being matched. Moreover, a few words did not fit this pattern (probably an entry mistake).
- **Action:** I made different groups to match sequences, that did not fit the standard pattern (entry mistakes).

  `(", ")|(")|(,)|( )`

  The vertical line is a logic operator, which means OR. So this regex should catch either the quotation – comma – white-space sequence OR quotation marks OR commas OR white-spaces that do not follow this sequence (effectively all signs that are misplaced).

- **Result**: This worked, but left a line-break at the beginning and at the end of the list and in places which did not follow the sequence of quotation – comma – space.

3.3 Final Thoughts

18.00: I tried for hours to find a way to removed the extra line-breaks, but I could not. So for now, I will leave it as it is.

4 13-09-2020

4.1 Regex (Regular expressions): Stop-word list from R to Voyant! - Part 2

12.00: I will proofread my assignment and submit it. I do not think, that I can crack the last part of the assignment.

13.00: While proofreading my assignment, I suddenly realised the solution. I could just use a circumflex inside the brackets to indicate, which patterns I do not want to match.

4.2 Actions and Results

- **Action:** I started by specifying that, I do not want any patterns containing white-spaces, commas or periods.

  ```
  ([^" ,]+)
  ```

  The + at the end ensures, that the characters are matched as a word, rather than individual letters.

- **Result:** It worked, now I am matching all words in the list without matching commas, white-spaces or quotation marks

- **Action:** Now I just need to match the commas, the white-spaces and the quotation marks. So I wrote the following regex:

  ```
  ([^" ,]+)([" ,]*)
  ```

  This should match either a quotation mark, a white-space or a comma, if it is used between zero and unlimited times after the first group.

- **Result:** It almost worked. However, I'm not matching the first quotation mark on the list.

- **Action:** However, I can just set the regex to match zero or more quotation marks at the start of the list by adding a quotation mark and an asterisk at the start of the regex

  ```
  ("*)([^" ,]+)([" ,]*)
  ```

  This should match the first quotation mark on the list.

- **Result:** It almost worked. There are two places, where there is a blank line-break. But that is probably because there are two line-breaks in the list. So I just need to match the line-breaks, as I did in the previous task.
- **Action:** I added two groups for matching line-breaks in the regex. One at the beginning, and one at the end. A line-break at the end should be enough. But, by having one at the beginning as well, this regex should catch all line-breaks. Lastly, I made these line-breaks optional using the asterisk.

```
(\r*)("*)([^" ,]+)([" ,]*)(\r*)
```

- **Result:** It worked! This regex means the following: Find a pattern that starts with ZERO OR MORE LINE-BREAKS followed by ZERO OR MORE QUOTATION MARKS and ONE OR MORE CHARACTERS THAT DOES NOT INCLUDE A QUOTATION MARK, A COMMA OR A WHITE-SPACE followed by ZERO OR MORE QUOTATION MARKS, COMMAS OR WHITE-SPACES followed by ZERO OR MORE LINE-BREAKS.

4.3 Final Thoughts

I learned that sometimes it helps to just take a step back from the project and return later to review it. It was while re-reading my journal, that I discovered how to solve this problem.