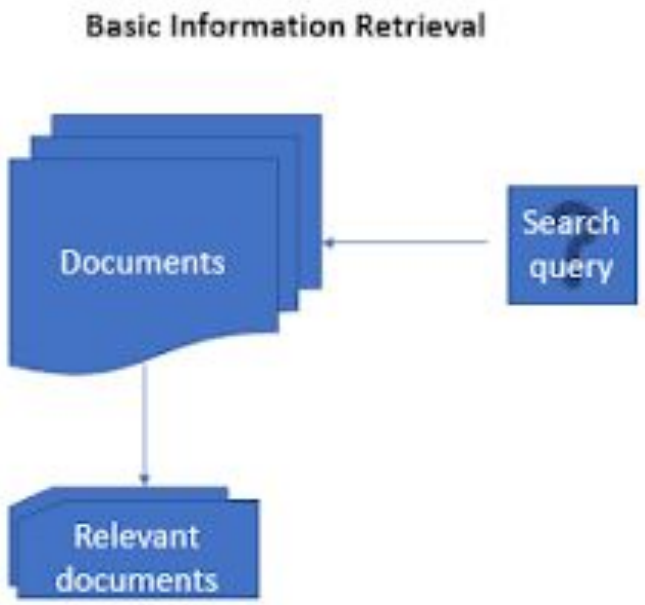
מבוא לאיחזור מידע

תרגול 1

מבוא לאיחזור מידע

גילוי והצגה אוטומטיים של עובדות, חוקים, וקשרים החבויים בתוכן טקסטואלי.



מעבדה 1 - הכר את הנתונים

בשבועות הקרובים נעבוד עם נתונים שנקצרו מטוויטר.

הנתונים הינם טוויטים שעברו סיווג לדבר נאצה (תוכן גזעני או סקסיסטי)

הקובץ הינו קובץ CSV - עמודה ראשונה הינה ID, שניה הינה סיווג לדברי נאצה (מלל גזעני או סקסיסטי) , ושלישית היא הטקסט עצמו.

מעבדה 1 - חלק א - פייתון

כיתבו קוד בפייתון, מומלץ בעזרת שימוש ב pandas , על הקוד לנתח את קובץ הנתונים, וליצור קובץ חדש גם הוא מופרד בפסיקים.
עבור כל טוויט יתווספו הפרטים הבאים:

1. מספר מילים
2. מספר אותיות
3. גודל מילה ממוצע
4. מיספור של stopwords בעזרת שימוש בחבילה NLTK
5. מספר תווים מספריים
6. ספירת כמות של תווים מיוחדים יכולה לתת עוד מידע על אופי הטקסט. עבור טוויטים לדוגמא מס ה - # . הוסיפו עמודה עם מס ה - # ובמידה ויש עוד סימנים יחודיים שכדאי למספר הוסיפו גם אותם.
7. מספר מילים שנכתבו באותיות גדולות (ביטוי לכעס)

מבוא לאיחזור מידע

היכרות עם R

R Reserved Words

Reserved words in R

if	else	repeat	while	function
for	in	next	break	TRUE
FALSE	NULL	Inf	NaN	NA
NA_integer_	NA_real_	NA_complex_	NA_character_	...

ניתן לראות גם בעזרת

```
> help(reserved)

> ?reserved
```

R Variables and Constants

Identifiers in R:

- Identifiers can be a combination of letters, digits, period (.) and underscore (_).
- Identifiers must start with a letter or a period. If it starts with a period, it cannot be followed by a digit.
- Reserved words in R cannot be used as identifiers.

Coding convention:

a.variable.name is preferred over a_variable_name or alternatively we could use camel case as aVariableName

Constants in R

Numeric Constants - integer, double, complex, hex representation

```
> typeof(5)
[1] "double"
> typeof(5L)
[1] "integer"
> typeof(5i)
[1] "complex"
> 0xff
[1] 255
```

Constants in R Cont’

Character Constants - can be represented using either single quotes (') or double quotes (") as delimiters.

```
> 'example'
[1] "example"
> typeof("5")
[1] "character"
```

Built-in Constants - there are some built in constants such as LETTERS, letters, pi, month.name (use it carefully it’s implemented as variables and can cause ambiguity)

R Assignment Operators

Assignment can be done with <- or with =

<- is more common, and = operator has some more meanings

We can use the function c() (as in concatenate) to make vectors in R

Assignment Operators in R

Operator	Description
<-, <<-, =	Leftwards assignment
->, ->>	Rightwards assignment

R Arithmetic Operator

Arithmetic Operators in R

Operator	Description
+	Addition
–	Subtraction
*	Multiplication
/	Division
^	Exponent
%%	Modulus (Remainder from division)
%/%	Integer Division

R Relational Operators

Relational Operators in R

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to

R Logical Operators

Logical Operators in R

Operator	Description
!	Logical NOT
&	Element-wise logical AND
&&	Logical AND
|	Element-wise logical OR
||	Logical OR

Precedence and Associativity of different operators in R from highest to lowest

Operator Precedence in R

Operator	Description	Associativity
^	Exponent	Right to Left
-x, +x	Unary minus, Unary plus	Left to Right
%%	Modulus	Left to Right
*, /	Multiplication, Division	Left to Right
+, –	Addition, Subtraction	Left to Right
<, >, <=, >=, ==, !=	Comparisions	Left to Right
!	Logical NOT	Left to Right
&, &&	Logical AND	Left to Right
|, ||	Logical OR	Left to Right
->, ->>	Rightward assignment	Left to Right
<-, <<-	Leftward assignment	Right to Left
=	Leftward assignment	Right to Left

Flow control

```
if (test_expression) {
statement
}

if (test_expression) {
statement1
} else {
statement2
}
```

If else statement can assign the value to a var:

```
y <- if(x > 0) 5 else 6
```

Can be used also with the function ifelse() :
ifelse(test_expression, x, y)

```
for (val in sequence)
{
statement
}

while (test_expression)
{
statement
}

repeat {
statement
}
```

Break and Next statements can be used inside loops (must be used in repeat)

R Functions

```
func_name <- function (argument) {
statement
}
```

Example:

```
pow <- function(x, y) {
# function to print x raised to the power y
result <- x^y
print(paste(x,"raised to the power", y, "is", result))
}
```

And calling the pow() function

```
>pow(8, 2)
```

R Functions Cont’ - named args

Named Arguments - we can state the value of an argument using its name.

The order of the inserted args makes no difference in such case

```
> pow(8, 2)
[1] "8 raised to the power 2 is 64"
> pow(x = 8, y = 2)
[1] "8 raised to the power 2 is 64"
> pow(y = 2, x = 8)
[1] "8 raised to the power 2 is 64"
> pow(x=8, 2)
[1] "8 raised to the power 2 is 64"
> pow(2, x=8)
[1] "8 raised to the power 2 is 64"
```

R Functions Cont’ - Default Values for Arguments

```
pow <- function(x, y = 2) {
# function to print x raised to the power y
result <- x^y
print(paste(x,"raised to the power", y, "is", result))
}
```

R Functions Cont’ - Syntax of return()

Return a single or list value from a function

```
multi_return <- function() {
my_list <- list("color" = "red", "size" = 20, "shape" = "round")
return(my_list)
}
```

R Vector

- Vectors are generally created using the c() function.

- Since, a vector must have elements of the same type, this function will try and coerce elements to the same type, if they are different.

- Coercion is from lower to higher types from logical to integer to double to character.

```
> x <- c(1, 5, 4, 9, 0)
> typeof(x)
[1] "double"
> length(x)
[1] 5
> x <- c(1, 5.4, TRUE, "hello")
> x
[1] "1"     "5.4"   "TRUE"  "hello"
> typeof(x)
[1] "character"
```

to double to character.

R Vector - Cont’

Creating a vector using : operator

```
> x <- 1:7; x

[1] 1 2 3 4 5 6 7

> y <- 2:-2; y

[1]  2  1  0 -1 -2
```

Creating a vector using seq() function

```
> seq(1, 3, by=0.2)          # specify step size
[1] 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0
> seq(1, 5, length.out=4)    # specify length
of the vector
[1] 1.000000 2.333333 3.666667 5.000000
```

R Vector - Cont’

Using integer vector as index

```
> x
[1]  0  2  4  6  8 10
> x[3]          # access 3rd element
[1] 4
> x[c(2, 4)]     # access 2nd and 4th element
[1] 2 6
> x[-1]          # access all but 1st element
[1]  2  4  6  8 10
> x[c(2, -4)]    # cannot mix positive and negative integers
Error in x[c(2, -4)] : only 0's may be mixed with negative subscripts
> x[c(2.4, 3.54)]    # real numbers are truncated to integers
[1] 2 4
```

R Vector - Cont’

Using logical vector as index

```
> x[c(TRUE, FALSE, FALSE, TRUE)]
[1] -3  3
> x[x < 0]  # filtering vectors based on conditions
[1] -3 -1
> x[x > 0]
[1] 3
```

Using character vector as index

```
> x <- c("first"=3, "second"=0, "third"=9)
> names(x)
[1] "first"  "second" "third"
> x["second"]
second
0
> x[c("first", "third")]
first third
3     9
```

R Vector - Cont’

modify a vector

```
> x
[1] -3 -2 -1  0  1  2
> x[2] <- 0; x    # modify 2nd element
[1] -3  0 -1  0  1  2
> x[x<0] <- 5; x  # modify elements less
than 0
[1] 5 0 5 0 1 2
> x <- x[1:4]; x  # truncate x to first 4
elements
[1] 5 0 5 0
```

delete a Vector

```
> x
[1] -3 -2 -1  0  1  2
> x <- NULL
> x
NULL
> x[4]
NULL
 9
```

Tidy text

עבודה עצמית - הכר את החבילה Tidy text

https://www.tidytextmining.com/tidytext.html

ספריות נוספות לעיון

https://cran.r-project.org/web/views/NaturalLanguageProcessing.html

מעבדה 1 - חלק ב - R

כיתבו קוד בR , מומלץ בעזרת שימוש ב Tidy tex, על הקוד לנתח את קובץ הנתונים, וליצור קובץ חדש גם הוא מופרד בפסיקים.

עבור כל טוויט יתווספו הפרטים הבאים:

1. מספר מילים
2. מספר אותיות
3. גודל מילה ממוצע
4. מיספור של stopwords בעזרת שימוש בחבילה NLTK
5. מספר תווים מספריים
6. ספירת כמות של תווים מיוחדים יכולה לתת עוד מידע על אופי הטקסט. עבור טוויטים לדוגמא מס ה - # . הוסיפו עמודה עם מס ה - # ובמידה ויש עוד סימנים יחודיים שכדאי למספר הוסיפו גם אותם.
7. מספר מילים שנכתבו באותיות גדולות (ביטוי לכעס)