

Introduction

Introduction to Text Mining with R

Ayoub Bagheri

Contents

ChatGPT	3
.	4
.	4
.	5
Course Logistics	5
Course materials	5
Teachers	5
Program	6
Goal of the course	7
What is Text Mining?	8
Example	8
.	8
Example	9
Example	10
Language is hard!	11
Text Mining to the Rescue!	11
.	11
Applications	12
Text mining applications	12
Who wrote the Wilhelmus?	12
Text Classification	12
Which ICD-10 codes should I give this doctor's note?	13
Sentiment Analysis / Opinion Mining	13
Statistical Machine Translation	13
Dialog Systems	14

Question Answering Go beyond search	14
Which studies go in my systematic review?	15
.	15
And more	15
Process & Tasks	16
Text mining process	16
Pattern discovery tasks in text	16
And more in NLP	17
10-minute break	17
Regular Expressions	17
Regular expressions	17
Regular expressions	17
Regular expressions	17
Some simple regex searches	18
Disjunction	18
Brackets and dash	18
Negation	18
Question and period marks	19
Anchors	19
Common sets	19
Operators for counting	20
Other	20
Operator precedence hierarchy	20
Understanding Regular Expressions	20
In R	21
Example	21
Example	21
Errors	22
Errors cont.	22
Question	22
Question	22
Solution	22
Question	23
Question	23
Solution	23
.	24

Summary	24
Summary	24
Next: Text preprocessing	25
Practical 1	25
Are you curious about the end of the Example?	25
End	26

ChatGPT

A What is Introduction to Text Mining with R?

 Introduction to Text Mining with R is a course or a learning resource that provides an overview of text mining techniques using the R programming language. Text mining is the process of extracting meaningful information, patterns, and insights from unstructured textual data. It involves various tasks like text preprocessing, text classification, sentiment analysis, topic modeling, and more.

In the context of this course, the focus is on using R, a powerful and popular programming language for data analysis and statistical computing, to perform text mining tasks. R has numerous packages and libraries that facilitate text mining, making it an excellent choice for working with textual data.

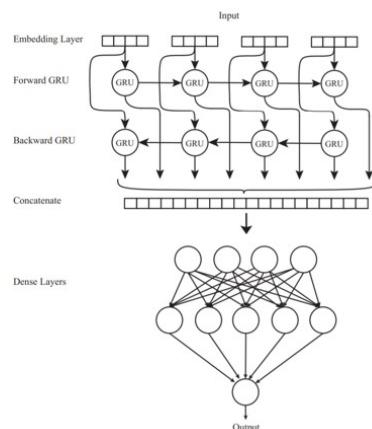
Did a poet with donkey ears write the oldest anthem in the world?

<https://dh2017.adho.org/abstracts/079/079.pdf>



Automatic detection of ICD10 codes in cardiology discharge letters

<https://www.nature.com/articles/s41746-021-00404-9>



Box 1: An example of a Dutch discharge letter from the dataset

Bovengenoemde patiënt was opgenomen op <DATUM-1> op de <PERSONA-1> voor het specialisme Cardiologie.

Reden van opname: STEMI inferior

Cardiale voorgeschiedenis: Blanco
Cardiovaskulair risicofactoren: Roken(-) Diabetes(-) Hypertensie(?) Hypercholesterolemie (?)

Presentatie: Om 19.30 pijn op de borst met ustraling naar de linkarm, zweten, misselijk. Ambulance gebeld en bij aansluiten monitor beeld van acuut ondervanndinfarct.

AMBU overdracht. 500 mg aspegic iv, ticagrelor 180 mg oraal, heparine, zofran eenmalig, 3x NTG spray. HD stabiel gebleven. Medicatie bij presentatie: Geen.

Lichamelijk onderzoek: Grauw, vegetatief. Halsvenen niet gestuwwd. Cor s1 s2 geen souffles. Pulm schoon. Extr warm en slank.

Aanvullend onderzoek: AMBU ECG: Sinusritme, STEMI inferior III/II Cvermoedelijk RCA.
Coronair angiografie (...). Conclusie angio: 1-vatslijden. PCI

Conclusie en beleid:

Bovengenoemde <LEFTUID-1> jarige man, blanco cardiale voorgeschiedenis, werd gepresenteerd vanwege een STEMI inferior waarbij een spoed PCI werd verricht waarna de pijn snel verdween. Hij is nu de <INSTELLING-1>. De patienten werden overgeplaatst naar de CCU van het <INSTELLING-2>... Dank van de snelle overname... Medicatie bij overplaatsing. Acetylsalicyzuur dispersablet 90 mg: oraal: 1x per dag 80 milligram: <DATUM-1>. Ticagrelor tablet 90 mg: oraal: 2x per dag 25 milligram: <DATUM-1>. Metoprolol tablet 50 mg: oraal: 2x per dag 40 milligram: <DATUM-1>. Atorvastatine tablet 40 mg (als cd-zout-3-water); oraal: 1x per dag 40 milligram: <DATUM-1>

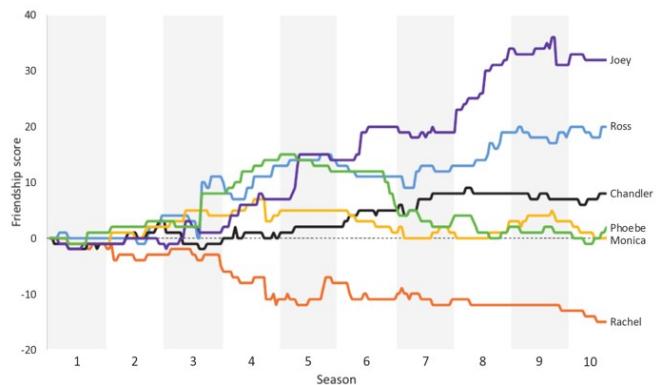
Samenvatting:

Hoofddiagnose: STEMI inferior vv PCI RCA. Geen nevenletsets. Nevendiagnoses: geen.

Complicities: geen Ontslag naar: CCU <INSTELLING-2>.

Who was the best Friend?

<https://rss.onlinelibrary.wiley.com/doi/epdf/10.1111/1740-9713.01574>



Course Logistics

Course materials

You can access the course materials quickly from

https://ayoubbagheri.nl/r_tm/

Teachers



Luka



Laurence



Daniel



Pablo

Program

```
## Warning in latex_new_row_builder(target_row, table_info, bold, italic,
## monospace, : Setting full_width = TRUE will turn the table into a tabu
## environment where colors are not really easily configurable with this package.
## Please consider turn off full_width.

## Warning in latex_new_row_builder(target_row, table_info, bold, italic,
## monospace, : Setting full_width = TRUE will turn the table into a tabu
## environment where colors are not really easily configurable with this package.
## Please consider turn off full_width.

## Warning in latex_new_row_builder(target_row, table_info, bold, italic,
## monospace, : Setting full_width = TRUE will turn the table into a tabu
## environment where colors are not really easily configurable with this package.
## Please consider turn off full_width.

## Warning in latex_new_row_builder(target_row, table_info, bold, italic,
## monospace, : Setting full_width = TRUE will turn the table into a tabu
```

```

## environment where colors are not really easily configurable with this package.
## Please consider turn off full_width.

## Warning in latex_new_row_builder(target_row, table_info, bold, italic,
## monospace, : Setting full_width = TRUE will turn the table into a tabu
## environment where colors are not really easily configurable with this package.
## Please consider turn off full_width.

## Warning in latex_new_row_builder(target_row, table_info, bold, italic,
## monospace, : Setting full_width = TRUE will turn the table into a tabu
## environment where colors are not really easily configurable with this package.
## Please consider turn off full_width.

## Warning in latex_new_row_builder(target_row, table_info, bold, italic,
## monospace, : Setting full_width = TRUE will turn the table into a tabu
## environment where colors are not really easily configurable with this package.
## Please consider turn off full_width.

## Warning in latex_new_row_builder(target_row, table_info, bold, italic,
## monospace, : Setting full_width = TRUE will turn the table into a tabu
## environment where colors are not really easily configurable with this package.
## Please consider turn off full_width.

```

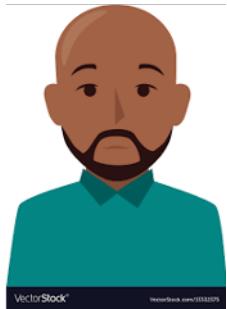
Time	Monday	Tuesday	Wednesday	Thursday
9:00 - 10:30	Lecture 1	Lecture 3	Lecture 5	Lecture 7
	Break	Break	Break	Break
10:45 – 11:45	Practical 1	Practical 3	Practical 5	Practical 7
11:45 – 12:30	Discussion 1	Discussion 3	Discussion 5	Discussion 7
	Lunch	Lunch	Lunch	Lunch
14:00 – 15:30	Lecture 2	Lecture 4	Lecture 6	Lecture 8
	Break	Break	Break	Break
15:45 – 16:30	Practical 2	Practical 4	Practical 6	Practical 8
16:30 – 17:00	Discussion 2	Discussion 4	Discussion 6	Discussion 8

Goal of the course

- Text data is everywhere!
- A lot of world's data is in unstructured text format
- The course teaches
 - text mining techniques
 - using R
 - on a variety of applications
 - in many domains.

What is Text Mining?

Example



- This is **Garry**!
- **Garry** works at Bol.com (a webshop in the Netherlands)
- He works in the dep of **Customer relationship management**.
- He uses Excel to read and search customers' reviews, extract aspects they wrote their reviews on, and identify their sentiments.
- Curious about his job? See two examples!

This is a nice book for both young and old. It gives beautiful life lessons in a fun way. Definitely worth the money!

+ Educational

+ Funny

+ Price

Nice story for older children.

+ Funny

- Readability

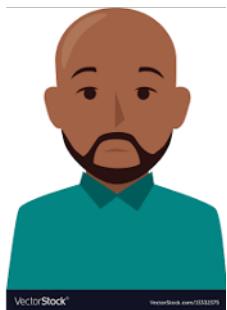
ANTOINE DE SAINT-EXUPÉRY

The Little Prince

The original
translation by
Katherine Woods,
with full-colour
illustrations



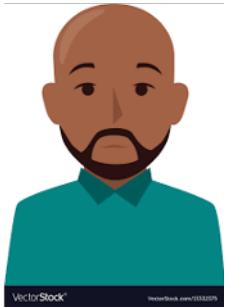
Example



- Garry likes his job a lot, but sometimes it is frustrating!
- This is mainly because their company is expanding quickly!
- Garry decides to hire **Larry** as his assistant.



Example



- Still, a lot to do for two people!
- Garry has some budget left to hire another assistant for couple of years!
- He decides to hire **Harry** too!
- Still, manual labeling using Excel is labor-intensive!



Language is hard!

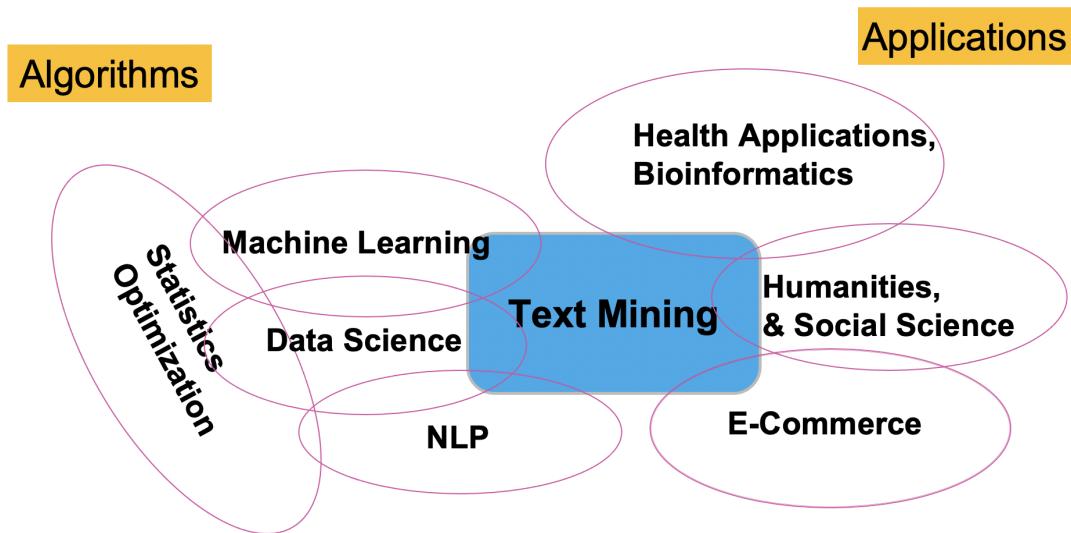
- Different things can mean more or less the same (“data science” vs. “statistics”)
- Context dependency (“You have very nice shoes”);
- Same words with different meanings (“to sanction”, “bank”);
- Lexical ambiguity (“we saw her duck”)
- Irony, sarcasm (“That’s just what I needed today!”, “Great!”, “Well, what a surprise.”)
- Figurative language (“He has a heart of stone”)
- Negation (“not good” vs. “good”), spelling variations, jargon, abbreviations
- All the above are different over languages, 99% of work is on English!

Text Mining to the Rescue!

- “the discovery by computer of new, previously unknown information, by automatically extracting information from different written resources” Hearst (1999)
 - Text mining is about looking for patterns in text, in a similar way that data mining can be loosely described as looking for patterns in data.
 - Text mining describes a set of linguistic, statistical, and machine learning techniques that model and structure the information content of textual sources. (Wikipedia)
-
- We won’t solve linguistics ...
 - In spite of the problems, text mining can be quite effective!

Applications

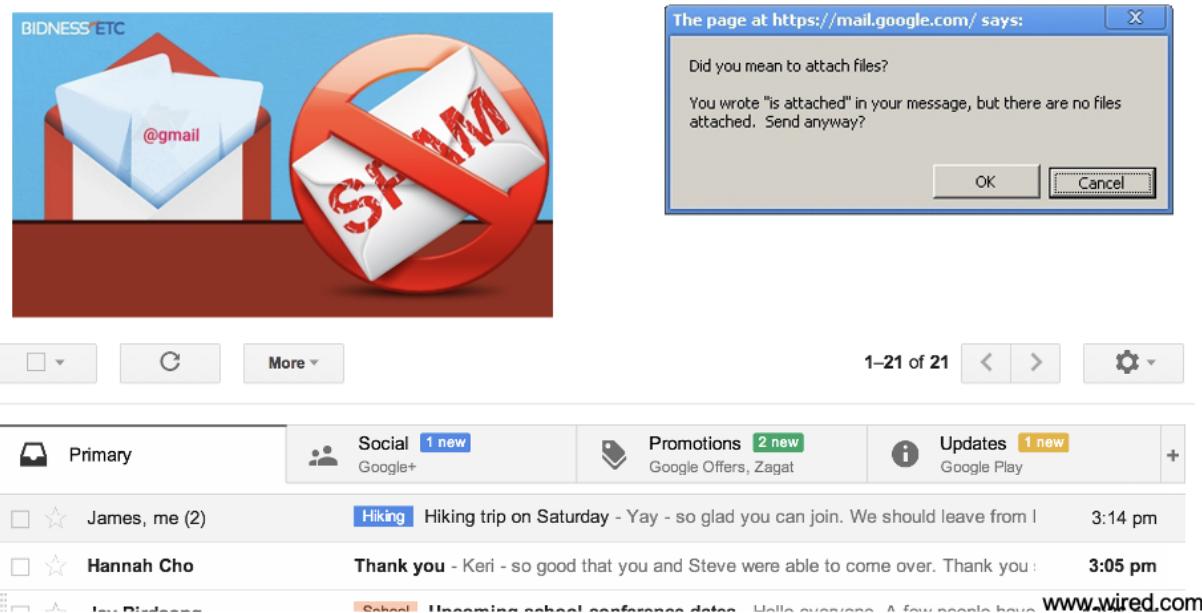
Text mining applications



Who wrote the Wilhelmus?

<https://dh2017.adho.org/abstracts/079/079.pdf>

Text Classification

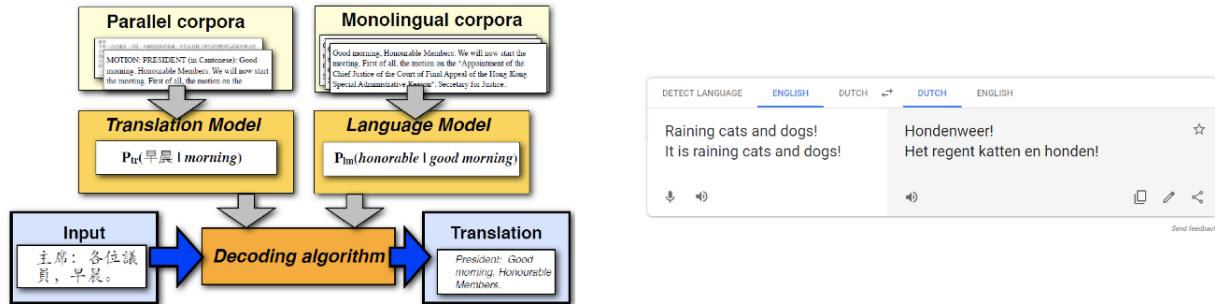


Which ICD-10 codes should I give this doctor's note?

Sentiment Analysis / Opinion Mining



Statistical Machine Translation



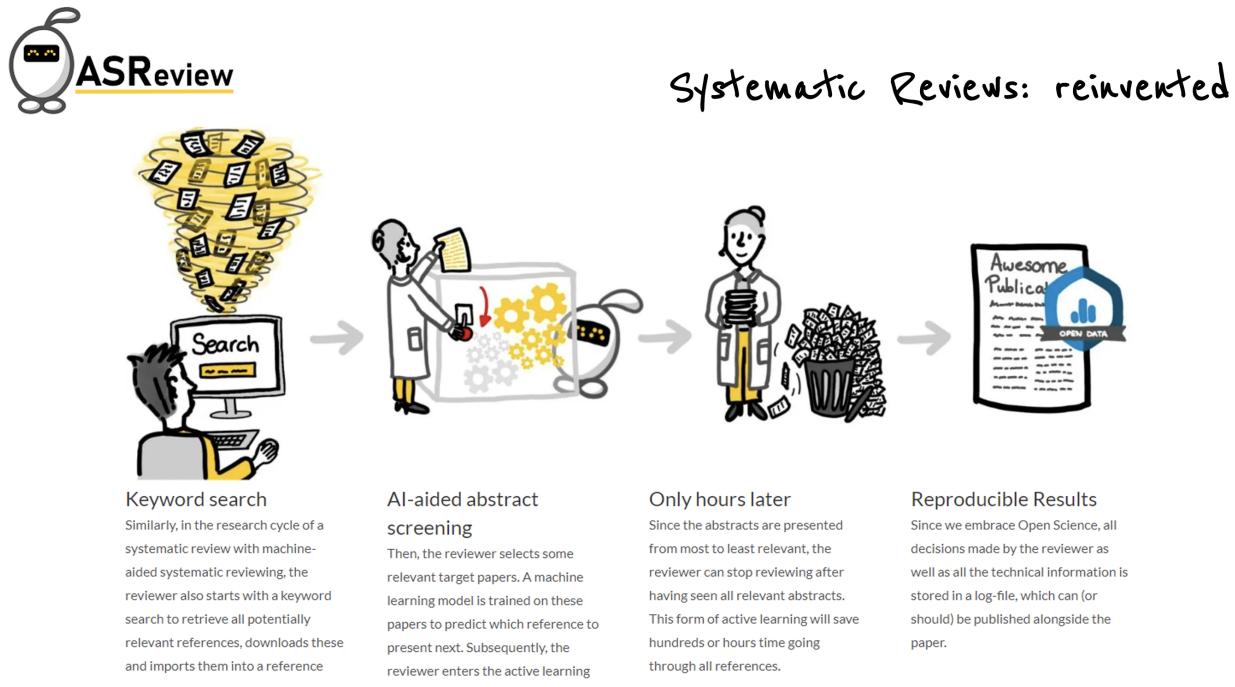
Dialog Systems



Question Answering | Go beyond search

The image shows two side-by-side search results. On the left, a Google search for "What is the capital of North Holland?" displays a thumbnail image of a church in Haarlem, a map of North Holland with Haarlem highlighted, and a text snippet stating that Haarlem is the capital and seat of the provincial government. On the right, a WolframAlpha search for "How old is Mark Rutte?" shows the input query, the result "52 years 5 months 28 days", and links to "Sources" and "Download Page".

Which studies go in my systematic review?



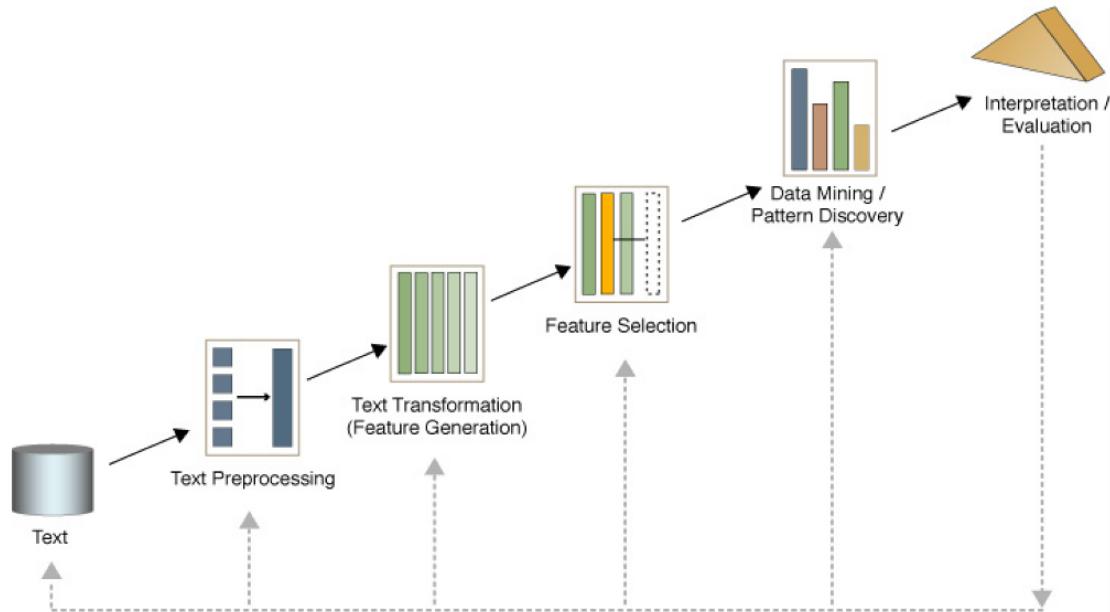
<https://asreview.nl/>

And more ...

- Automatically classify political news from sports news
- Authorship identification
- Age/gender identification
- Language Identification
- ...

Process & Tasks

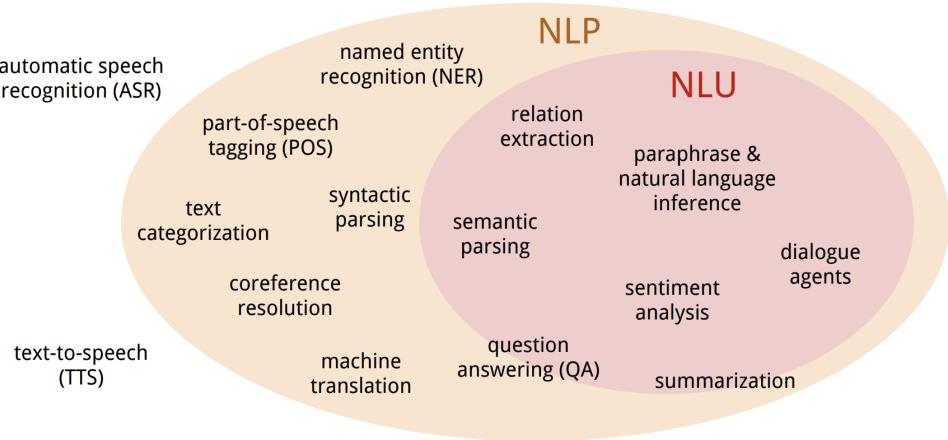
Text mining process



Pattern discovery tasks in text

- Text classification
- Text clustering
- Sentiment analysis
- Feature selection
- Topic modelling
- Responsible text mining
- Text summarization

And more in NLP



source: <https://nlp.stanford.edu/~wcmac/papers/20140716-UNLU.pdf>

10-minute break

Regular Expressions

Regular expressions

Really clever “wild card” expressions for matching and parsing strings.

http://en.wikipedia.org/wiki/Regular_expression

Regular expressions

In computing, a regular expression, also referred to as “regex” or “regexp”, provides a concise and flexible means for matching strings of text, such as particular characters, words, or patterns of characters. A regular expression is written in a formal language that can be interpreted by a regular expression processor.

http://en.wikipedia.org/wiki/Regular_expression

Regular expressions

- A formal language for specifying text strings
- How can we search for any of these?
 - netherland
 - netherlands
 - Nederland
 - Netherlands

Some simple regex searches

RE	Example Patterns Matched
/woodchucks/	“interesting links to <u>woodchucks</u> and lemurs”
/a/	“Mary Ann stopped by Mona’s”
!/ /	“You’ve left the burglar behind again!” said Nori

Disjunction

The use of the brackets [] to specify a disjunction of characters:

RE	Match	Example Patterns
/[wW]oodchuck/	Woodchuck or woodchuck	“ <u>Woodchuck</u> ”
/[abc]/	‘a’, ‘b’, or ‘c’	“In uomini, in soldati”
/[1234567890]/	any digit	“plenty of <u>7</u> to 5”

The pipe symbol | is also for disjunction:

Pattern	Matches
groundhog woodchuck	
yours mine	yours mine
a b c	= [abc]

Brackets and dash

The use of the brackets [] plus the dash - to specify a range:

RE	Match	Example Patterns Matched
/[A-Z]/	an upper case letter	“we should call it ‘ <u>Drenched Blossoms</u> ’ ”
/[a-z]/	a lower case letter	“ <u>my beans were impatient to be hoed!</u> ”
/[0-9]/	a single digit	“Chapter <u>1</u> : Down the Rabbit Hole”

Negation

The caret ^ for negation or just to mean ^:

RE	Match (single characters)	Example Patterns Matched
/[^A-Z]/	not an upper case letter	“Oyfn pripetchik”
/[^Ss]/	neither ‘S’ nor ‘s’	“I have no exquisite reason for’t”
/[^.]/	not a period	“our resident Djinn”
/[e^]/	either ‘e’ or ‘^’	“look up ^ now”
/a^b/	the pattern ‘a^b’	“look up <u>a^b</u> now”

Question and period marks

The question mark ? marks optionality of the previous expression:

RE	Match	Example Patterns Matched
/woodchucks?/	woodchuck or woodchucks	" <u>woodchuck</u> "
/colou?r/	color or colour	" <u>color</u> "

The use of the period . to specify any character:

RE	Match	Example Matches
/beg.n/	any character between <i>beg</i> and <i>n</i>	<u>begin</u> , <u>beg'n</u> , <u>begun</u>

Anchors

RE	Match
^	start of line
\\$	end of line
\b	word boundary
\B	non-word boundary

Common sets

Aliases for common sets of characters:

RE	Expansion	Match	First Matches
\d	[0-9]	any digit	Party_of_5
\D	[^0-9]	any non-digit	Blue_moon
\w	[a-zA-Z0-9_]	any alphanumeric/underscore	Daiyu
\W	[^\w]	a non-alphanumeric	!!!
\s	[\r\t\n\f]	whitespace (space, tab)	
\S	[^\s]	Non-whitespace	in_Concord

The backslash for escaping!

Operators for counting

RE	Match
*	zero or more occurrences of the previous char or expression
+	one or more occurrences of the previous char or expression
?	exactly zero or one occurrence of the previous char or expression
{n}	n occurrences of the previous char or expression
{n,m}	from n to m occurrences of the previous char or expression
{n,}	at least n occurrences of the previous char or expression
{,m}	up to m occurrences of the previous char or expression

- Patterns are **greedy**: In these cases regular expressions always match the largest string they can, expanding to cover as much of a string as they can.
- Enforce non-greedy matching, using another meaning of the ? qualifier.
 - The operator *? is a Kleene star that matches as little text as possible.
 - The operator +? is a Kleene plus that matches as little text as possible.

Other

Some characters that need to be backslashed:

RE	Match	First Patterns Matched
*	an asterisk “*”	“K_A_P_L_A_N”
\.	a period “.”	“Dr._Livingston, I presume”
\?	a question mark	“Why don’t they come and lend a hand_?”
\n	a newline	
\t	a tab	

Operator precedence hierarchy

Parenthesis	()
Counters	* + ? { }
Sequences and anchors	the ^my end\$
Disjunction	

Understanding Regular Expressions

- Very powerful and quite cryptic
- Fun once you understand them

- Regular expressions are a programming language with characters
- It is kind of an “old school” language

In R

The primary R functions for dealing with regular expressions are:

- `grep()`, `grep1()`: Search for matches of a regular expression/pattern in a character vector
- `regexpr()`, `gregexpr()`: Search a character vector for regular expression matches and return the indices where the match begins; useful in conjunction with `regmatches()`
- `sub()`, `gsub()`: Search a character vector for regular expression matches and replace that match with another string
- The `stringr` package provides a series of functions implementing much of the regular expression functionality in R but with a more consistent and rationalized interface.

Example

- Find all instances of the word “the” in a text.

the

Misses capitalized examples

[tT]he

Incorrectly returns words such as other or Netherlands

[^a-zA-Z][tT]he[^a-zA-Z]

Still not completely correct! What is missing?

Example

```
txt <- "The other the Netherlands will then be without the"
r <- gregexpr("[^a-zA-Z][tT]he[^a-zA-Z]", txt)
print(regmatches(txt, r))
```

```
## [[1]]
## [1] " the "
```

```
r <- gregexpr("(^|[^a-zA-Z])[tT]he($|[^a-zA-Z])", txt)
print(regmatches(txt, r))
```

```
## [[1]]
## [1] "The " " the " " the"
```

Errors

- The process we just went through was based on fixing two kinds of errors
 - Matching strings that we should not have matched (**there**, **Netherlands**)
 - * False positives (Type I)
 - Not matching things that we should have matched (**The**)
 - * False negatives (Type II)

Errors cont.

- In NLP we are always dealing with these kinds of errors.
- Reducing the error rate for an application often involves:
 - Increasing precision (minimizing false positives)
 - Increasing recall (minimizing false negatives).

Question

Given the text “this Summer School is Utrecht summer school”, which RE finds all the “summer school” mentions?

- $(^|[^a-zA-Z])[sS]ummer * [sS]chool(^|a-zA-Z)$
- $(^|[^a-zA-Z])[sS]ummer\s[sS]chool($|^a-zA-Z)$
- $(^|[^a-zA-Z])[sS]ummer *[sS]chool($|^a-zA-Z)$
- $[^a-zA-Z][sS]ummer\s[sS]chool($|^a-zA-Z)$

Question



1 Connect to www.wooclap.com/JYFUMG

2 You can participate



1 Not yet connected? Send **@JYFUMG** to **0970 1420 2908**

2 Send your answer to the same number

Solution

```

txt <- "this Summer School is utrecht summerschool"
r <- gregexpr("(^|[^a-zA-Z])[sS]ummer *[sS]chool($|[^a-zA-Z])", txt)
#r <- gregexpr("(^|[^a-zA-Z])[sS]ummer\\s*[sS]chool($|[^a-zA-Z])", txt)

print(regmatches(txt, r))

## [[1]]
## [1] " Summer School " " summerschool"

```

Question

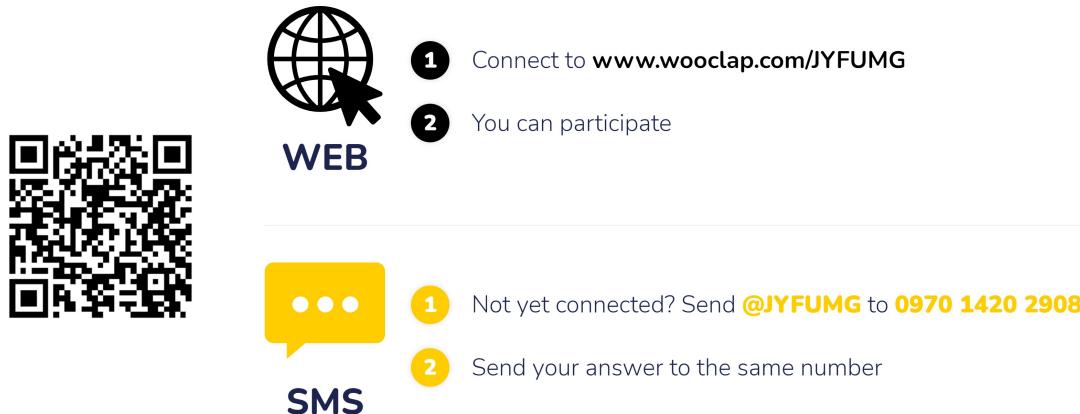
Suppose we want to build an application to help a user buy a car from textual catalogues. The user looks for any car cheaper than \$10,000.00.

Assume we are using the following data: `txt <- c("Price of Tesla S is $8599.99.", "Audi Q4 is $7000.", "BMW X5 costs $900")`

Which RE will help us to do this?

- $(^|\W)\$[0-9]\{0,4\}(\.[0-9][0-9])^*$
- $(^|\W)\$[0-9]\{0,3\}(\.[0-9][0-9])^+$
- $(^|\W)\$[0-9]\{0,4\}(\.[0-9][0-9])?$
- $(^|\W)\$[0-9][0-9][0-9][0-9](\.[0-9][0-9])^*$

Question



Solution

```

txt <- c("Price of Tesla S is $8599.99.",
       "Audi Q4 is $7000.",
       "BMW X5 costs $900")
r <- gregexpr("(^|\W)\\$[0-9]\{0,4\}(\\. [0-9] [0-9])?", txt)
print(regmatches(txt, r))

```

```

## [[1]]
## [1] " $8599.99"
##
## [[2]]
## [1] " $7000"
##
## [[3]]
## [1] " $900"

```



<http://xkcd.com/208/>

Summary

Summary

- Text data is everywhere!
- Language is hard!
- Sophisticated sequences of regular expressions are often the first model for any text processing tool
- Regular expressions are a cryptic but powerful language for matching strings and extracting elements from those strings
- The basic problem of text mining is that text is not a neat data set
- One solution: text pre-processing

Next: Text preprocessing

- is an approach for cleaning and noise removal of text data.
- brings your text into a form that is analyzable for your task.
- transforms text into a more digestible form so that machine learning algorithms can perform better.

Practical 1

Are you curious about the end of the Example?



- During one of the coffee moments at the company, **Garry** was talking about their situation at the department of Customer relationship management.
- When **Carrie**, her colleague from the **Data Science department**, hears the situation, she offers Garry to use Text Mining!!
- She says: “Text mining is your friend; it can help you to make the process way faster than Excel by filtering words and recommending labels.”
- She continues : “Text mining is a subfield of AI and NLP and is related to data science, data mining and machine learning.”
- After consulting with Larry and Harry, they decide to give text mining a try!

End

