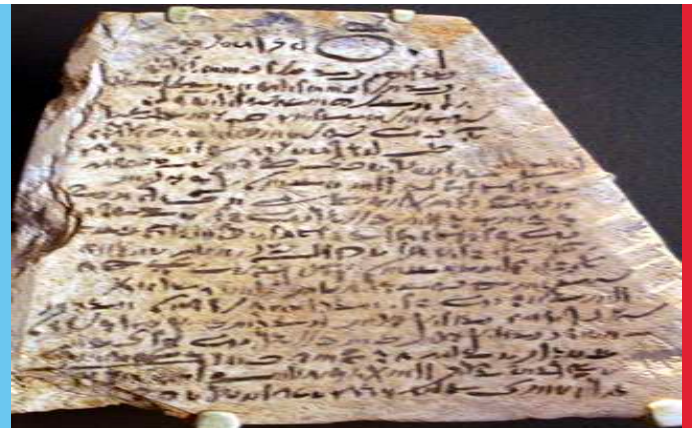
MTU
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

Computer Science Department

Natural Language Processing

Week2: Text Processing with Regular Expressions, *Unix tools and* Scikit-Learn



Dr Haithem Afli

@AfliHaithem

Haithem.afli@mtu.ie

Recap

Go to **www.menti.com** and use the code **9816 0270**

Or use the link/QR Code below

https://www.menti.com/wxhysyji96

Recap

MTU
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

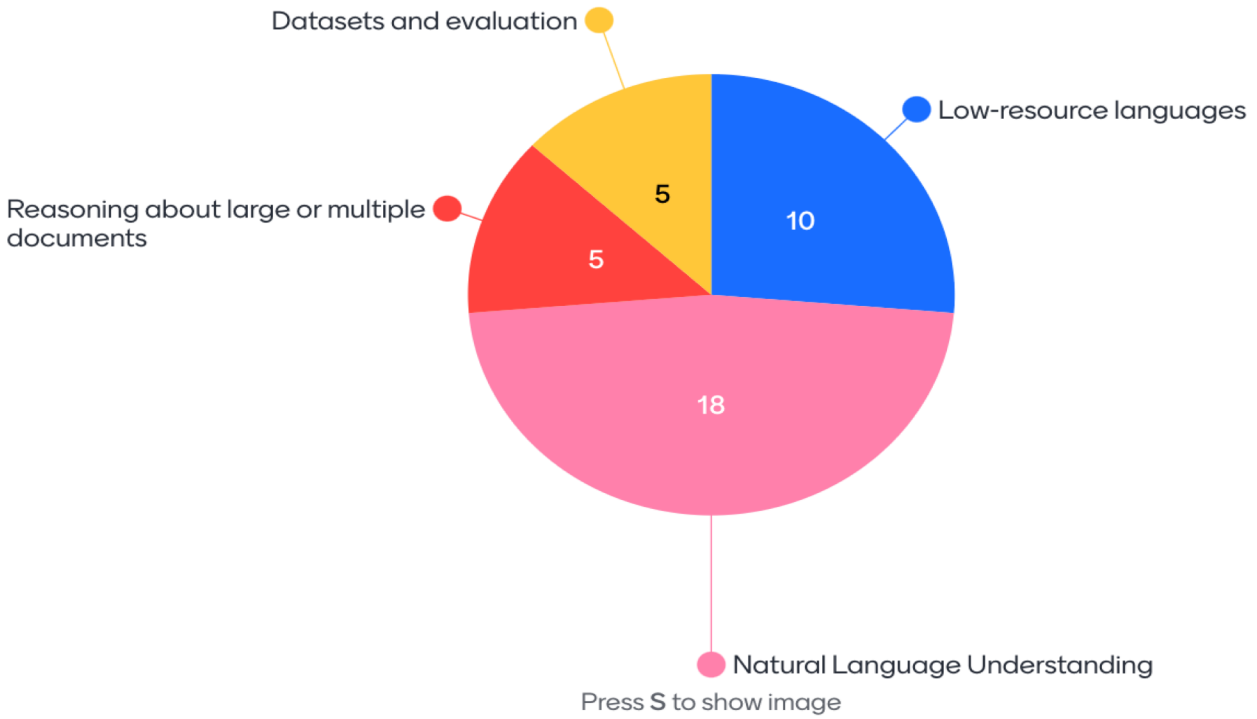
1) **What are the main Open-Problems in natural language processing?**

2) Describe NLP technologies with three words!

Go to **www.menti.com** and use the code **9816 0270**

What are the main Open-Problems in natural language processing?

Mentimeter

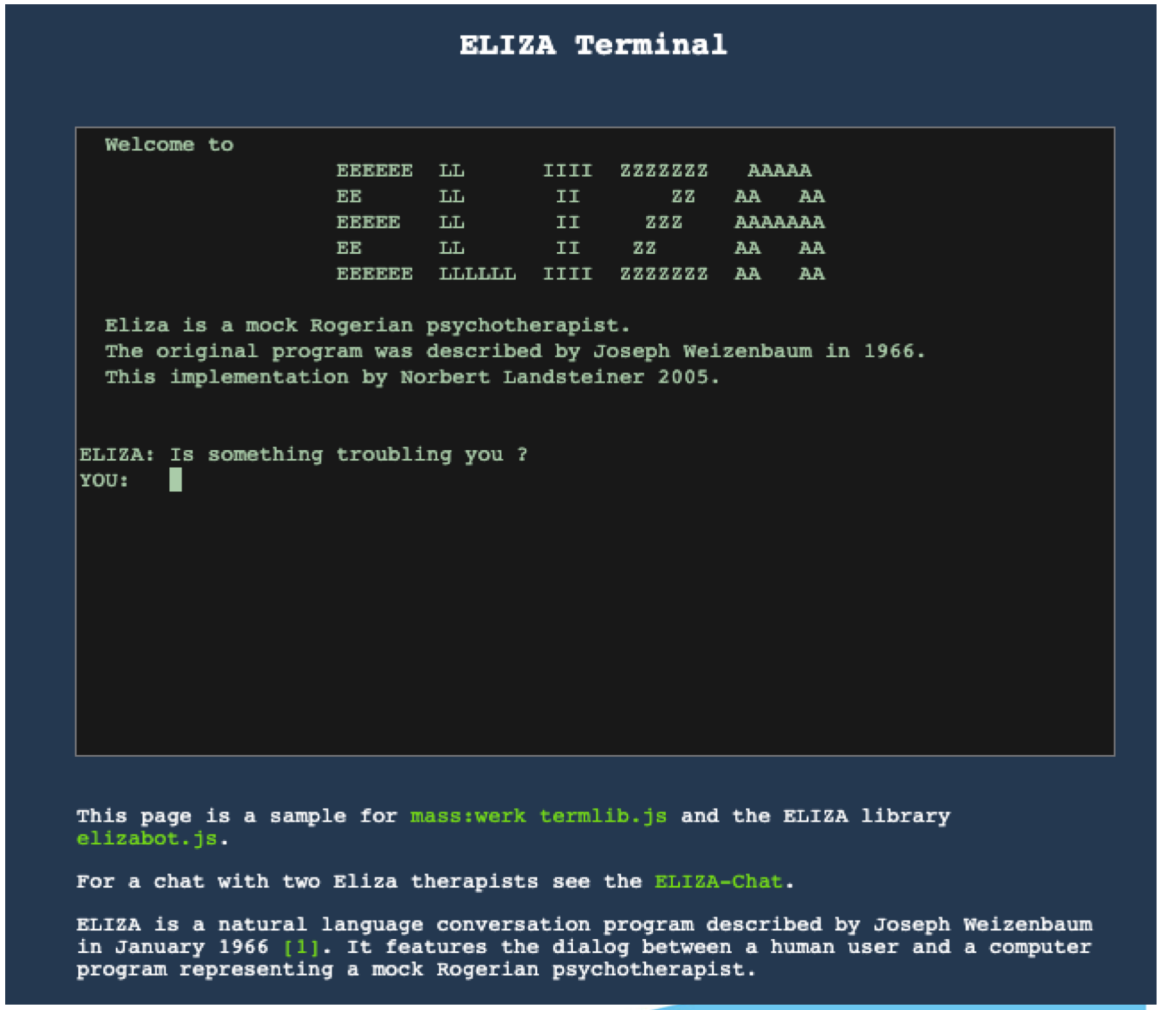


20

Outline

1. **Regular Expressions**
2. *Unix tools*
3. *Tokenization*
4. *Text processing with* Scikit-Learn

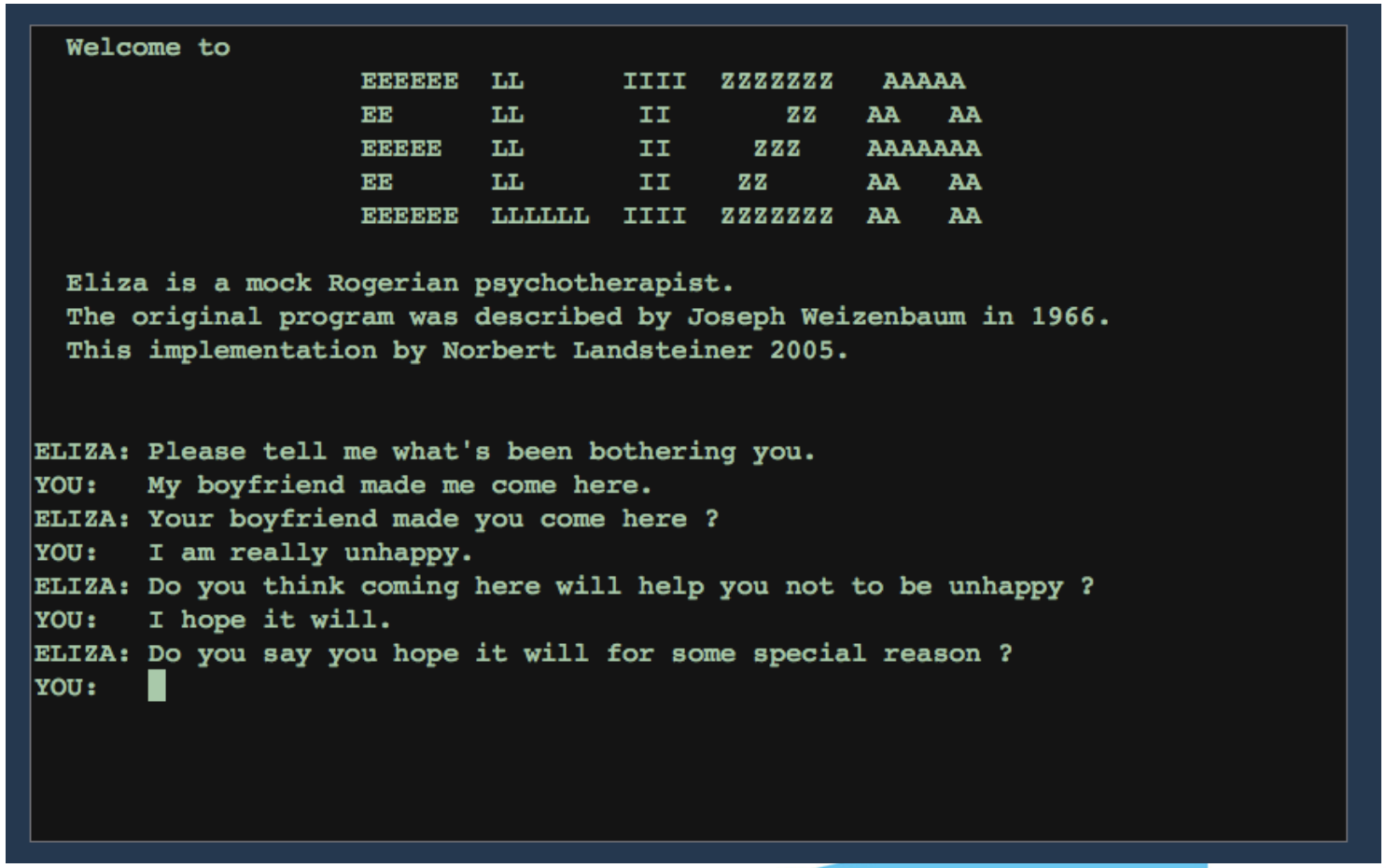
Please tell me what's been bothering you.



Eliza

- Eliza is a mock Rogerian psychotherapist.
  - Rogerian psychotherapy, or person-centered therapy, is a form of talk-psychotherapy developed by psychologist Carl Rogers in the 1940s & 1950s.
- The original program was described by Joseph Weizenbaum in 1966.
  - Primitive NLP, using simple scripts and **regular expressions**
  - http://www.masswerk.at/elizabot/eliza.html
- Chatterbots or chatbots
- Eliza effect
- Turing test
- Regular expressions

Please tell me what's been bothering you.



Regular Expressions

- In computer science, RE (or regex, regexp) is a language used for **specifying text search string**.
- A *regular expression* is a ***formula* in a *special language*** that is used for specifying a simple class of *string*.
- Formally, a regular expression is an algebraic notation for characterizing a set of strings.
- RE search requires
  - a *pattern* that we want to search for, and
  - a *corpus* of texts to search through.
- http://www.regexr.com

Regular Expressions

- A RE search function will search through the corpus returning all texts that contain the pattern.
  - In a Web search engine, they might be the entire documents or Web pages.
  - In a word-processor, they might be individual words, or lines of a document. (We take this paradigm.)
    - E.g., the UNIX `grep` command

Regular Expressions
Basic Regular Expression Patterns

- The use of the brackets `[]` to specify a disjunction of characters.

RE	Match	Example Patterns
`/[wW]oodchuck/`	Woodchuck or woodchuck	"Woodchuck"
`/[abc]/`	'a', 'b', *or* 'c'	"In uomini, in soldati"
`/[1234567890]/`	any digit	"plenty of 7 to 5"

- The use of the brackets `[]` plus the dash − to specify a range.

RE	Match	Example Patterns Matched
`/[A-Z]/`	an uppercase letter	"we should call it 'Drenched Blossoms'"
`/[a-z]/`	a lowercase letter	"my beans were impatient to be hoed!"
`/[0-9]/`	a single digit	"Chapter 1: Down the Rabbit Hole"

Regular Expressions
Basic Regular Expression Patterns

- Uses of the caret ^ for negation or just to mean ^

RE	Match (single characters)	Example Patterns Matched
[^A-Z]	not an uppercase 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 a^ b now"

- The question-mark ? marks optionality of the previous expression.

RE	Match	Example Patterns Matched
woodchucks?	woodchuck or woodchucks	"woodchuck"
colou?r	color or colour	"colour"

- The use of period . to specify any character

RE	Match	Example Patterns
/beg.n/	any character between *beg* and *n*	begin, beg'n, begun

Regular Expressions: Anchors ^ $

Pattern	Matches
^[A-Z] Starting with uppercase letter	Palo Alto
^[^A-Za-z]	1 “Hello”
\.$	The end.
.$	The end? The end!

Regular Expressions
Disjunction, Grouping, and Precedence

- Disjunction

```
/cat|dog/
```

- Precedence

```
/gupp(y|ies)/
```

- Operator precedence hierarchy

```
()
+ ? { }
the   ^my   end$
|
```

Regular Expressions
A Simple Example

- To find the English article ***the***

```
/the/

/[tT]he/

/\b[tT]he\b/

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

/(^|[^a-zA-Z])[tT]he([^a-zA-Z]|$)/
```

Errors

- The process we just went through was based on fixing two kinds of errors
  - Matching strings that we should not have matched (there, then, other)
    - 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 two antagonistic efforts:
  - Increasing accuracy or precision (minimizing false positives)
  - Increasing coverage or recall (minimizing false negatives).

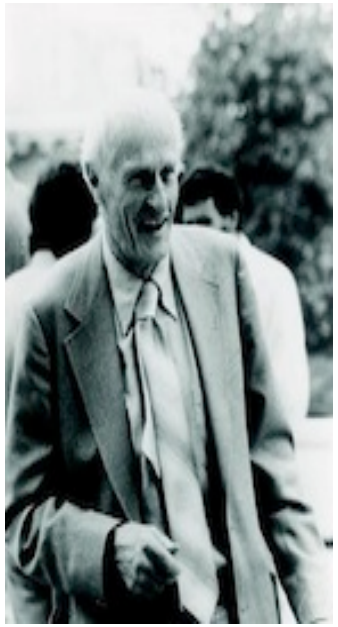
Regular Expressions
Advanced Operators

Aliases for common sets of characters

RE	Expansion	Match	Example Patterns
\d	[0-9]	any digit	Party␣of␣5
\D	[^0-9]	any non-digit	Blue␣moon
\w	[a-zA-Z0-9␣]	any alphanumeric or space	Daiyu
\W	[^\w]	a non-alphanumeric	!!!!
\s	[␣\r\t\n\f]	whitespace (space, tab)	
\S	[^\s]	Non-whitespace	in␣Concord

Regular Expressions: ? * + .

Pattern	Matches	
colou?r	Optional previous char	color colour
oo*h!	0 or more of previous char	oh! ooh! oooh! ooooh!
o+h!	1 or more of previous char	oh! ooh! oooh! ooooh!
baa+		baa baaa baaaa baaaaa
beg.n	any char	begin begun begun beg3n



Stephen C Kleene

Kleene *, Kleene +

Regular Expressions
Advanced Operators

Regular expression 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

Regular Expressions
Advanced Operators

Some characters that need to be backslashed

RE	Match	Example Patterns Matched
*	an asterisk "*"	"K*A*P*L*A*N"
\.	a period "."	"Dr. Livingston, I presume"
\?	a question mark	"Would you light my candle?"
\n	a newline	
\t	a tab	

Regular Expressions
Regular Expression Substitution, Memory, and ELIZA

```
s/regexp1/regexp2/
```

- E.g. *the 35 boxes → the <35> boxes*

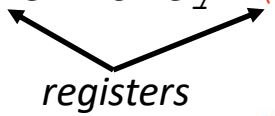
```
s/([0-9]+)/<\1>/
```

- The following pattern matches "*The bigger they were, the bigger they will be*", not *"The bigger they were, the faster they will be"*

```
/the (.*)er they were, the\1er they will be/
```

- The following pattern matches "*The bigger they were, the bigger they were*", not *"The bigger they were, the bigger they will be"*

```
/the (.*)er they (.*), the\1er they \2/
```



Regular Expressions
Regular Expressions Substitution, Memory, and ELIZA

- Eliza worked by having a cascade of regular expression substitutions that each match some part of the input lines and changed them
  - *my → YOUR, I'm → YOU ARE …*

```
s/.* YOU ARE (depressed|sad) .*/I AM SORRY TO HEAR YOU ARE \1/
s/.* YOU ARE (depressed|sad) .*/WHY DO YOU THINK YOU ARE \1/
s/.* all .*/IN WHAT WAY/
s/.* always .*/CAN YOU THINK OF A SPECIFIC EXAMPLE/
```

$User_1$: Men are all alike.
$ELIZA_1$: IN WHAT WAY
$User_2$: They're always bugging us about something or other.
$ELIZA_2$: CAN YOU THINK OF A SPECIFIC EXAMPLE
$User_3$: Well, my boyfriend made me come here.
$ELIZA_3$: YOUR BOYBRIEND MADE YOU COME HERE
$User_4$: He says I'm depressed much of the time.
$ELIZA_4$: I AM SORRY TO HEAR YOU ARE DEPRESSED

Discussion

Outline

1. Regular Expressions
2. ***Unix tools***
3. *Tokenization*
4. *Text processing with* Scikit-Learn

Unix Tools

- **grep**: search for a pattern (regex)
- **sort**
- **uniq –c** (count duplicates)
- **tr** (translate characters)
- **wc** (word – or line – count)
- **sed** (edit string -- replacement)
- **cat** (send file(s) in stream)
- **man** (manual)
- **cut** (columns in tab-separated files)
- **paste** (paste columns)
- **head** (top of the file)
- **tail** (bottom of the file)
- **rev** (reverse lines)
- **comm** (compare)
- **shuf** (shuffle lines of text)
- Input/output redirection:
  - **>**, **<**, **|**

- *Piping commands together can be simple yet powerful in Unix*
- *It gives flexibility.*
- *Traditional Unix philosophy: small tools that can be composed*

Simple Tokenization in UNIX

- (Inspired by Ken Church's UNIX for Poets.)
- Given a text file, output the word tokens and their frequencies

```
tr -sc 'A-Za-z' '\n' < shakes.txt
      | sort
      | uniq –c
```

Change all non-alpha to newlines

Sort in alphabetical order

Merge and count each type

```
1945 A                25 Aaron
  72 AARON             6 Abate
  19 ABBESS            1 Abates
   5 ABBOT             5 Abbess
 ... ...               6 Abbey
                       3 Abbot
                      ... …
```

Simple Tokenization in UNIX

- (Inspired by Ken Church's UNIX for Poets.)
- Given a text file, output the word tokens and their frequencies

```
cat shakes.txt
      | tr -sc 'A-Za-z' '\n'
      | sort
      | uniq –c
```

Change all non-alpha to newlines

Sort in alphabetical order

Merge and count each type

```
1945 A                25 Aaron
  72 AARON             6 Abate
  19 ABBESS            1 Abates
   5 ABBOT             5 Abbess
 ... ...               6 Abbey
                       3 Abbot
                      .... …
```

Counting

- Merging upper and lower case

```
tr 'A-Z' 'a-z' < shakes.txt | tr –sc 'A-Za-z' '\n' | sort | uniq –c
```

- Sorting the counts

```
tr 'A-Z' 'a-z' < shakes.txt | tr –sc 'A-Za-z' '\n' | sort | uniq –c | sort –n –r
```

```
23243 the
22225 i
18618 and
16339 to
15687 of
12780 a
12163 you
10839 my
10005 in
8954  d
```

What happened here?

Shakespeare - Sonnet 50

How heavy do I journey on the way,
When what I seek, my weary travel's end,
Doth teach that ease and that repose to say,
'Thus far the miles are measured from thy friend!'
The beast that bears me, tired with my woe,
Plods dully on, to bear that weight in me,
As if by some instinct the wretch did know
His rider **lov'd** not speed being made from thee.

....

Not to mention contractions such as had/would: he'd said, he'd say...

Some of the output

- `tr -sc 'A-Za-z' '\n' < nyt_200811.txt | sort | uniq -c | head –n 5`

```
  25476 a
   1271 A
      3 AA
      3 AAA
      1 Aalborg
```

- `tr -sc 'A-Za-z' '\n' < nyt_200811.txt | sort | uniq -c | head`
- Gives you the first 10 lines
- tail does the same with the end of the input
- (You can omit the “-n” but it’s discouraged.)

Sorting and reversing lines of text

- `sort`
- `sort –f` Ignore case
- `sort –n` Numeric order
- `sort –r` Reverse sort
- `sort –nr` Reverse numeric sort

- `echo “Hello” | rev`

grep

- Grep finds patterns specified as regular expressions
  - **g**lobally search for **r**egular **e**xpression and **p**rint

- Finding words ending in –ing:
- `grep 'ing$' nyt.words |sort | uniq –c`

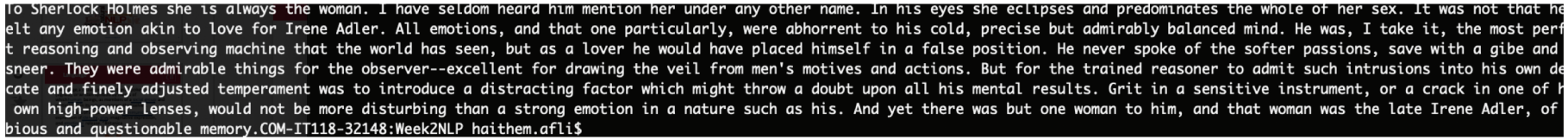
grep

- grep is a filter – you keep only some lines of the input
- `grep gh` keep lines containing “gh”
- `grep '^con'` keep lines beginning with “con”
- `grep 'ing$'` keep lines ending with “ing”
- `grep –v gh` keep lines NOT containing “gh”

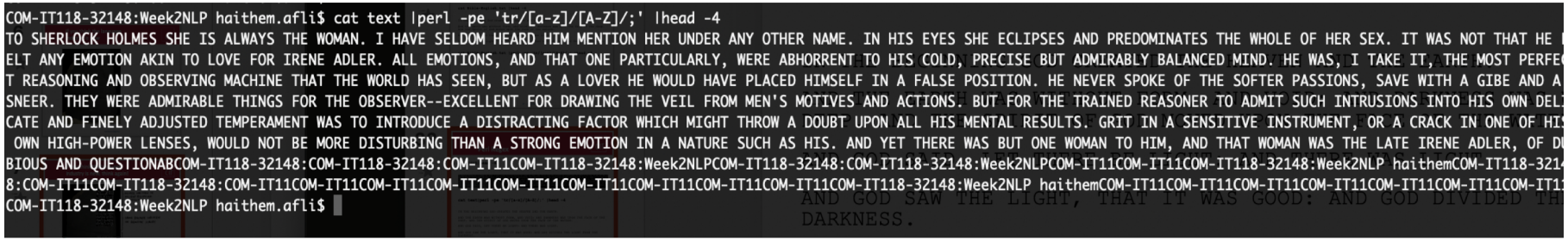
- `grep -P` Perl regular expressions (extended syntax)
- `grep -P '^[A-Z]+$' nyt.words |sort|uniq -c` ALL UPPERCASE

Perl Regex

```
cat text|head -4
```




```
cat text|perl -pe 'tr/[a-z]/[A-Z]/;' |head -4
```



Extended Counting Exercises

How common are different sequences of vowels (e.g., ieu)?

Extended Counting Exercises

MTU
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

How common are different sequences of vowels (e.g., ieu)?

```
cat text|perl -pe 'tr/[A-Z]/[a-z]/; s/[^aieou]/\n/g;'|sort |uniq -c |sort –nr
```

MTU
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

Discussion

Recap

Go to **www.menti.com** and use the code **9816 0270**

Or use the link/QR Code below

https://www.menti.com/wxhysyji96

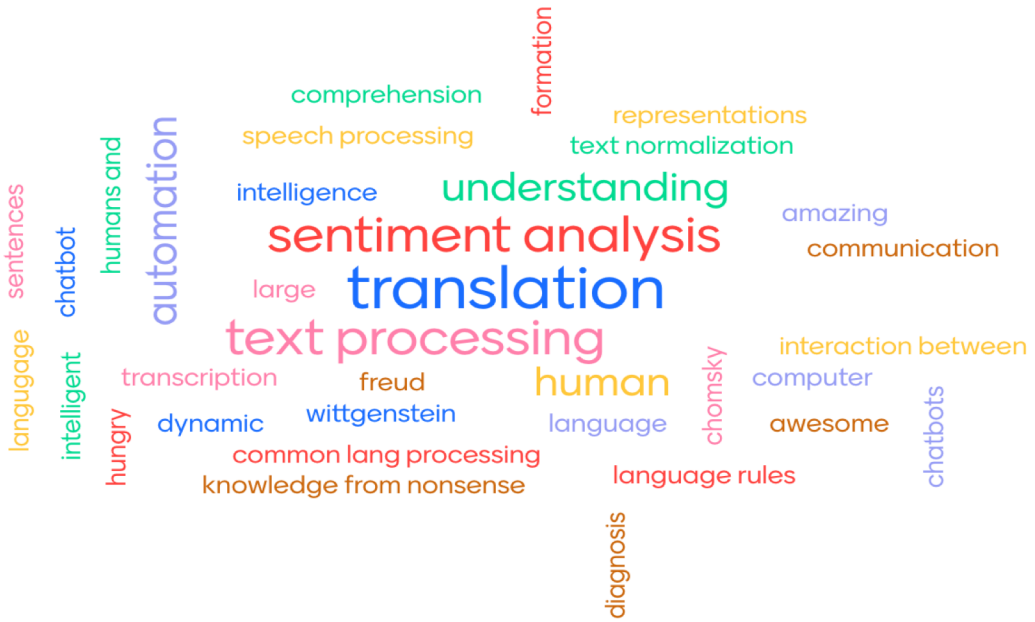
Recap

1) What are the main Open-Problems in natural language processing?

2) Describe NLP technologies with three words!

Go to **www.menti.com** and use the code **9816 0270**

Describe NLP technologies with three words!



Press **S** to show image

MTU
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

Discussion

Outline

1. Regular Expressions
2. *Unix tools*
3. ***Tokenization***
4. *Text processing with* Scikit-Learn

Issues in Tokenization

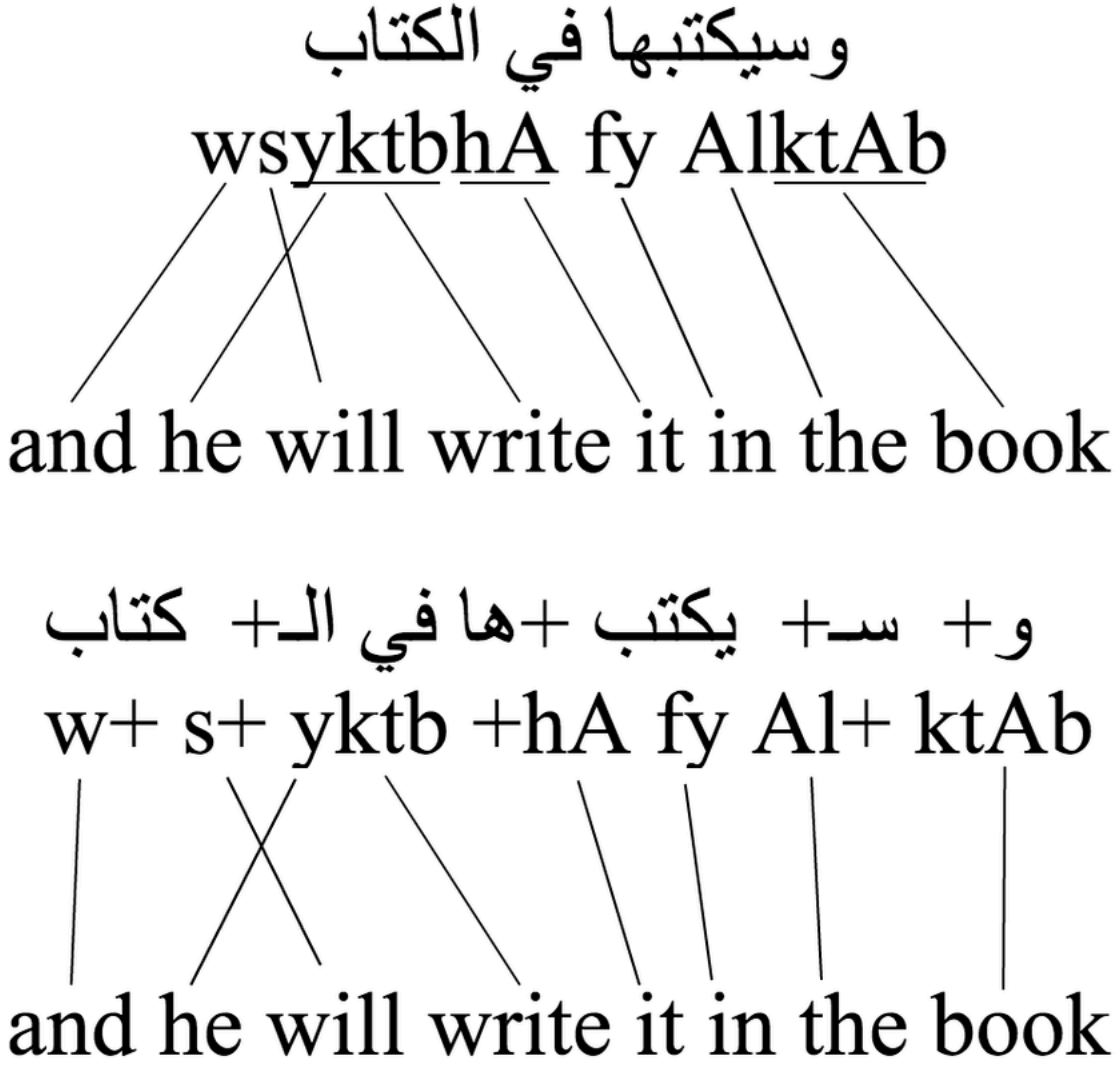
- `Finland's capital` → `Finland Finlands Finland's ?`
- `what're, I'm, isn't` → `What are, I am, is not`
- `Hewlett-Packard` → `Hewlett Packard ?`
- `state-of-the-art` → `state of the art ?`
- `Lowercase` → `lower-case lowercase lower case ?`
- `San Francisco` → one token or two?
- m.p.h., PhD. → ??

Tokenization: language issues

- French
  - ***L'ensemble*** → one token or two?
    - ***L*** ? ***L'*** ? ***Le*** ?
    - Want ***l'ensemble*** to match with ***un ensemble***

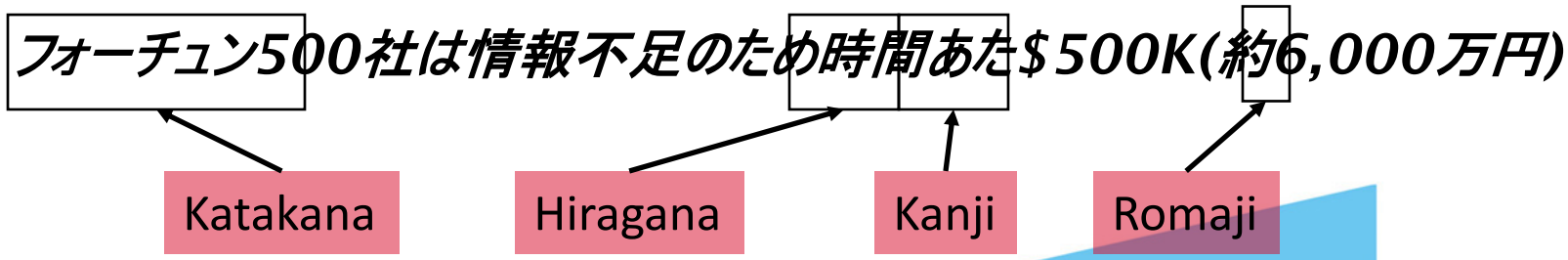
- German noun compounds are not segmented
  - ***Lebensversicherungsgesellschaftsangestellter***
  - 'life insurance company employee'
  - German information retrieval needs **compound splitter**

Tokenization: language issues



Tokenization: language issues

- Chinese and Japanese no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
  - Sharapova now lives in US southeastern Florida
- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats



End-user can express query entirely in hiragana!

Word Tokenization in Chinese

- Also called **Word Segmentation**
- Chinese words are composed of characters
- Characters are generally 1 syllable and 1 morpheme.
- Average word is 2.4 characters long.
- Standard baseline segmentation algorithm:

➢Maximum Matching (also called Greedy)

Maximum Matching Word Segmentation Algorithm

Given a wordlist of Chinese, and a string.

1. Start a pointer at the beginning of the string
2. Find the longest word in dictionary that matches the string starting at pointer
3. Move the pointer over the word in string
4. Go to 2

Max-match segmentation illustration

- Thecatinthehat
- Thetabledownthere

the cat in the hat

the table down there

theta bled own there

- Doesn’t generally work in English!

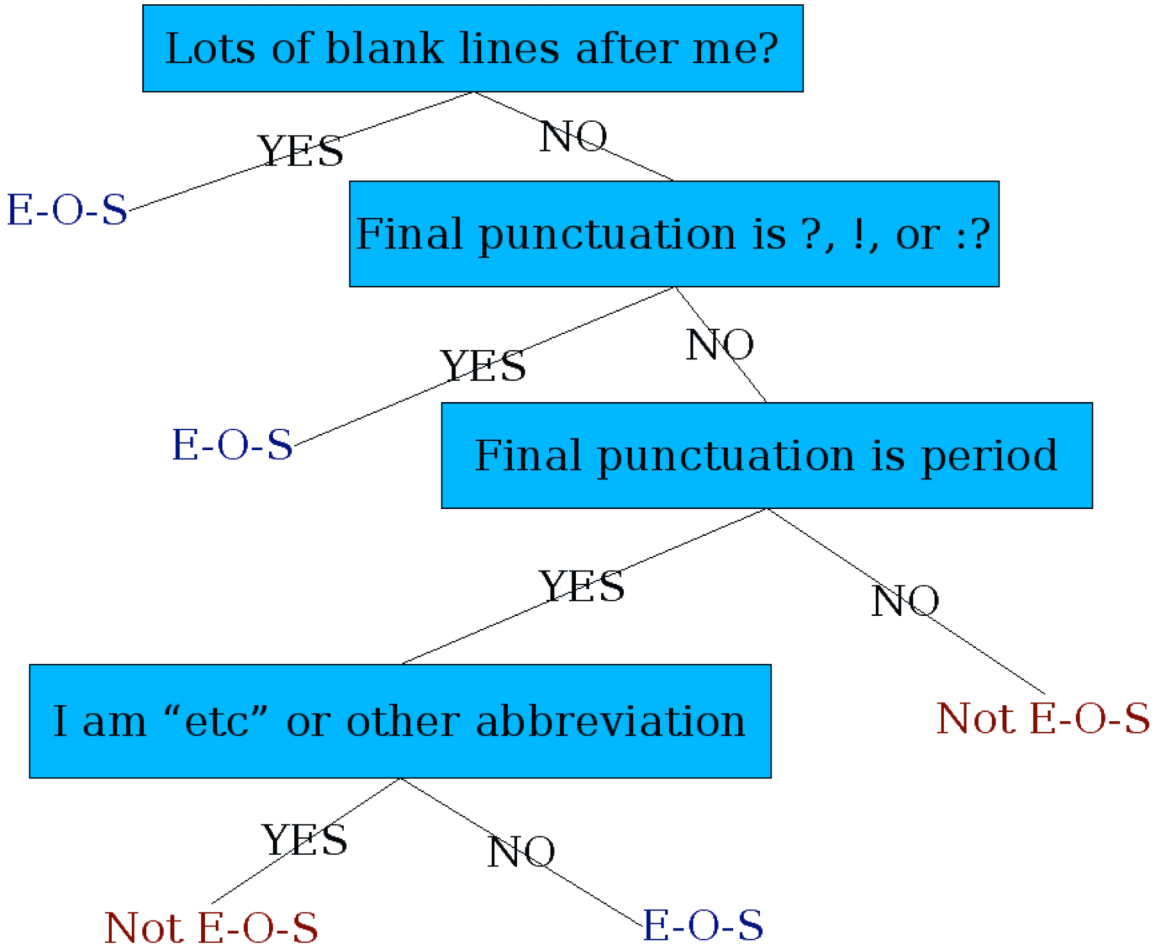
- But works astonishingly well in Chinese
- 莎拉波娃现在居住在美国东南部的佛罗里达。
- 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达

➢Modern probabilistic segmentation algorithms even better

Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous
- Sentence boundary
- Abbreviations like Inc. or Dr.
- Numbers like .02% or 4.3
- Build a binary classifier
- Looks at a "."
- Decides EndOfSentence/NotEndOfSentence
- Classifiers: hand-written rules, regular expressions, or machine-learning

Determining if a word is end-of-sentence: a Decision Tree



More sophisticated decision tree features

- Case of word with ".": Upper, Lower, Cap, Number
- Case of word after ".": Upper, Lower, Cap, Number

- Numeric features
- Length of word with "."
- Probability(word with "." occurs at end-of-s)
- Probability(word after "." occurs at beginning-of-s)

Implementing Decision Trees

- A decision tree is just an if-then-else statement
- The interesting research is choosing the features
- Setting up the structure is often too hard to do by hand
- Hand-building only possible for very simple features, domains
- For numeric features, it's too hard to pick each threshold
- Instead, structure usually learned by machine learning from a training corpus

Decision Trees and other classifiers

MTU
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

- We can think of the questions in a decision tree
- As features that could be exploited by any kind of classifier
- Logistic regression
- SVM
- Neural Nets
- etc.

MTU
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

Discussion

Next Week

1. *Text processing with* **Scikit-Learn**
2. **Language Modeling**

Thank You!

Haithem.afli@mtu.ie

Haithem Afli

@AfliHaithem