Course: Intelligent Systems

Unit 4: Language Technologies

Language technologies
Part 1

Mariano Rico
2022
Technical University of Madrid

NLP at a glance

- Session 1 (**Today**)
  - Encodings
  - Corpus
  - Normalization
  - Hands-on 1
- Session 2 (in 2 weeks, **Tue 13 Dec**)
  - Part of Speech
  - Sparse Vector models
  - TF-IDF
  - Sentiment analysis
  - Hands-on 2
- Session 3 (in 3 weeks, **Tue 20 Dec**)
  - Document classification
  - Information extraction
  - Hands-on 3
- Session 4 (after Xmas, **Tue 10 Jan**)
  - The neural revolution
  - Language Models 4 NLP tasks
  - Hands-on 4

Table of Contents

1. **Intro**
2. **Encodings**
3. **Corpus**
4. **Normalization**
5. **Hands-on 1**

Who am I?

Mariano Rico

- Creator and responsible for the Spanish DBpedia
  - DBpedia mentor in GSoC

Google Summer of Code

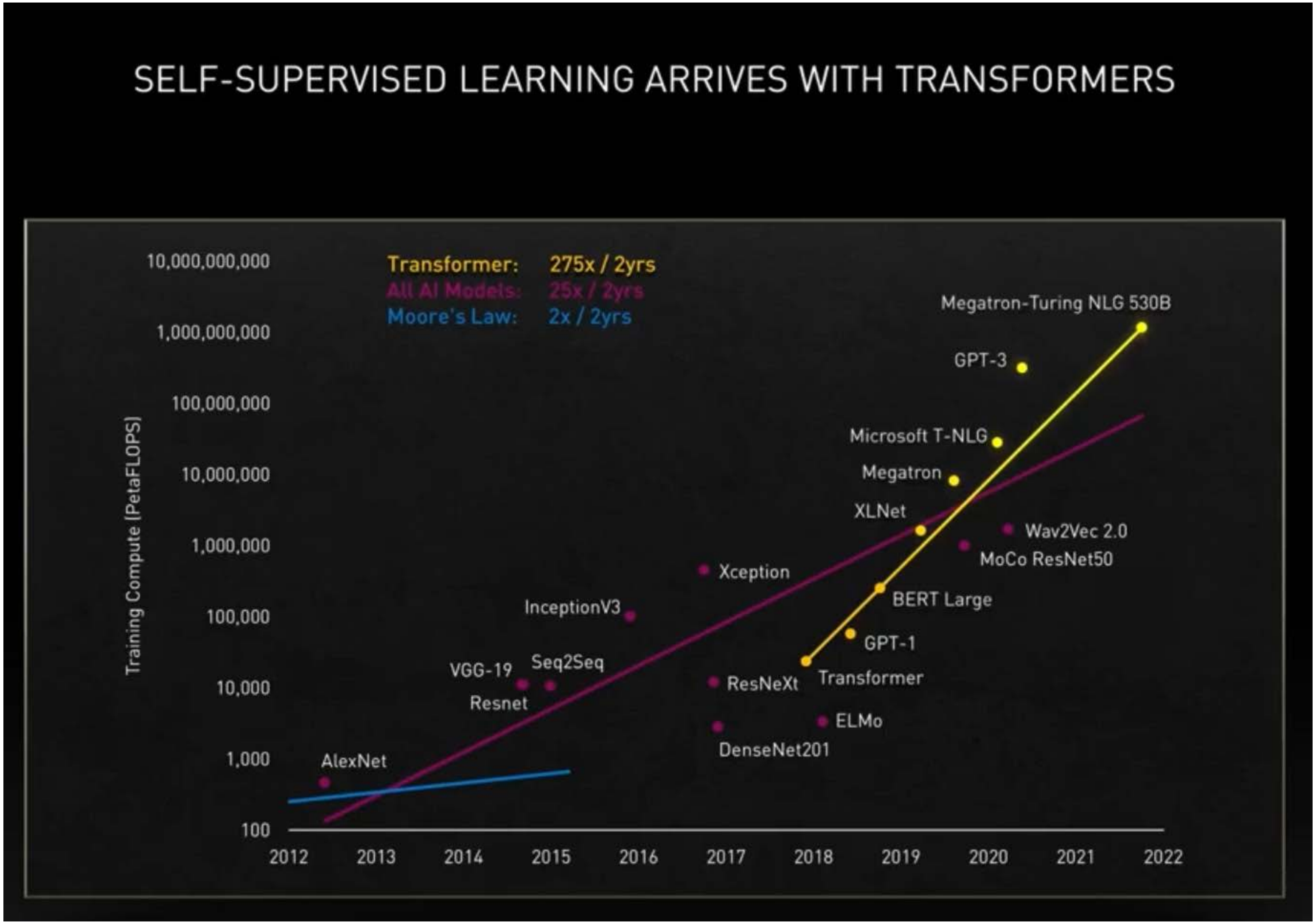
- *Office*: 2110
- *Email*: mariano.rico@upm.es

How are we going to work?

- Sessions with theory and practice
- Evaluation
  - No deliverable
  - Just attendance list and Moodle test
    - Attendance required: 75% (You can miss 1 in 4)
    - Moodle test (last 15 min of the January session)

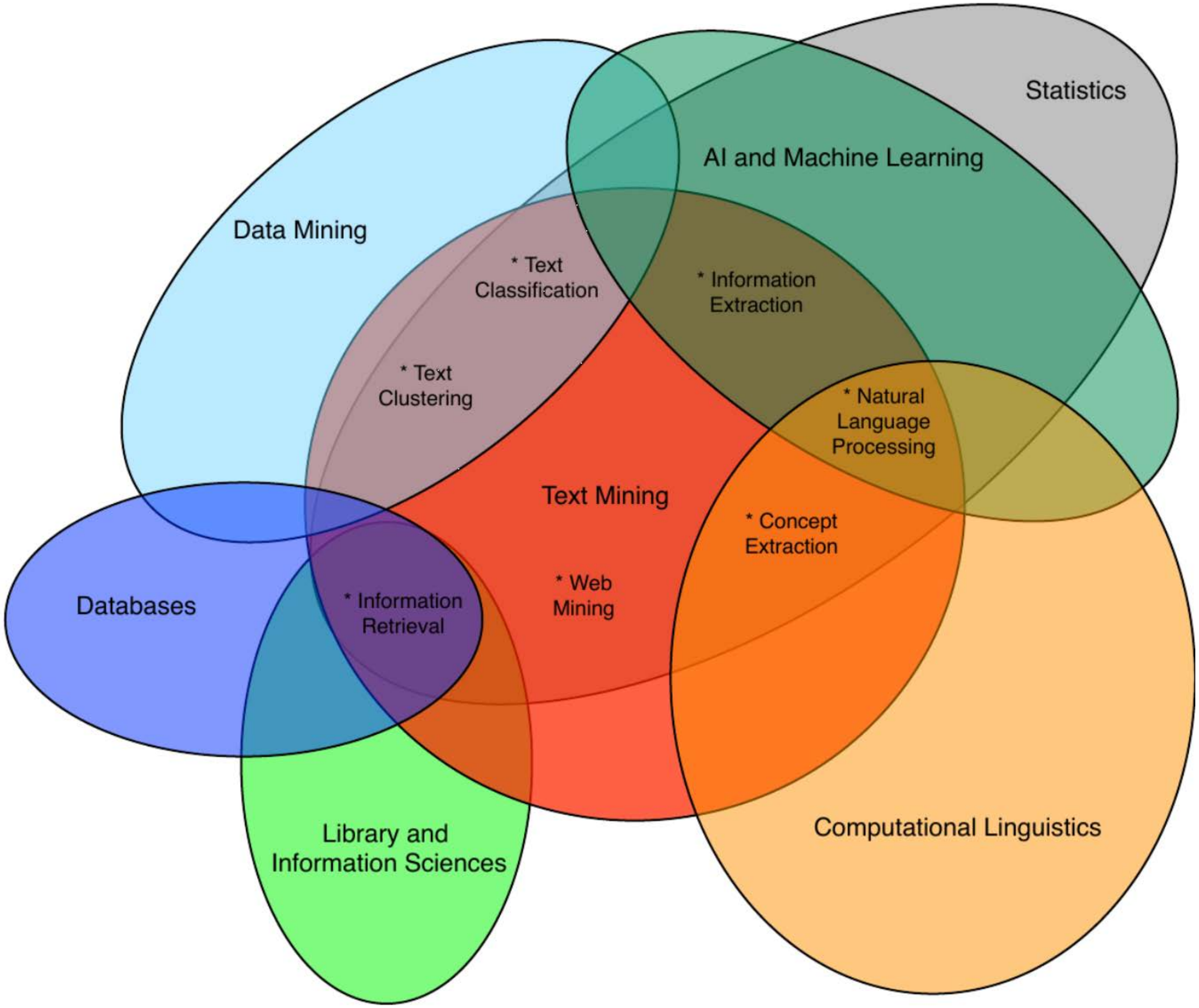
INTRO

The NLP hype



NVIDIA Nov. 2021 GTC Keynote by Jensen Huang

Where is NLP?

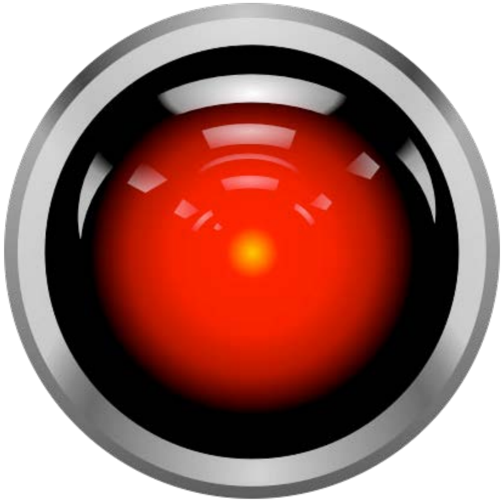


Miner, G., Elder, J., Hill. T, Nisbet, R., Delen, D. and Fast, A. (2012). Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications. Elsevier Academic Press. ISBN 978-0-12-386979-1

Motivating example

Open the pod bay doors, HAL

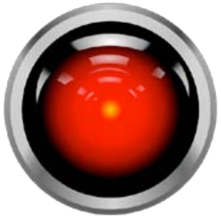
I’m sorry Dave, I’m afraid I can’t do that



Jurafsky, D. and Martin, J.H. (2000). Speech and Language Processing – An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall. ISBN 0-13-122798-X

Stanley Kubrick and Arthur C. Clarke, screenplay of 2001: A Space Odyssey

What does HAL know about?



I’m sorry Dave, I’m afraid I can’t do that

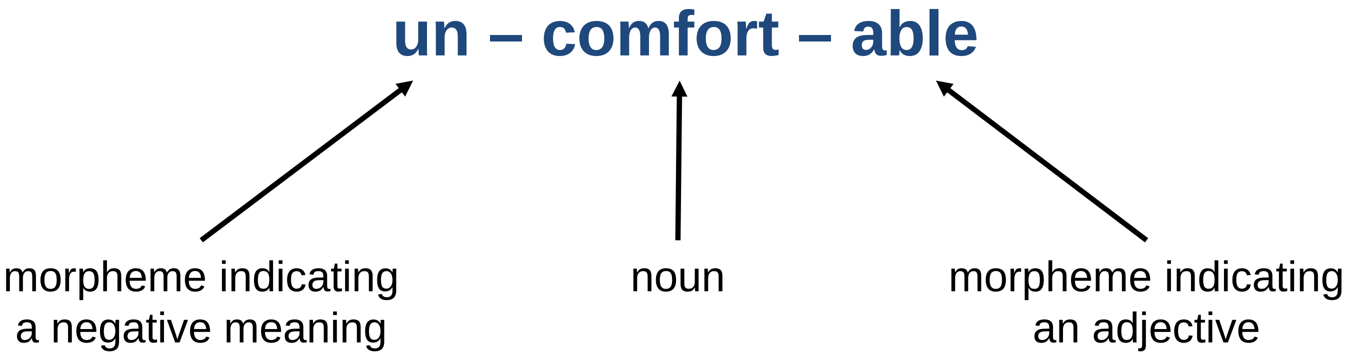
- **Phonology**
  - How words are pronounced in colloquial speech
- **Morphology**
  - Capture information about the shape and behaviour of words in context
- **Syntax**
  - Order and group words together
- **Semantics**
  - Meaning of words and how they combine to form larger meanings
- **Discourse**
  - Correctly structuring conversations
- **Pragmatics**
  - Appropriate use of polite and indirect language

Text characteristics

- **Ambiguity**
  - At all linguistic levels
  - Context is required for disambiguation
- **Dependency**
  - Words and phrases create context for each other
- **High dimensionality**
  - Tens of thousands of words (with abbreviations, spelling variants, etc.)
  - Only a very small percentage is used
- **Several input modes**
  - Different languages (human consumers)
  - Different formats (automated consumers)
- **Unstructured text**
  - Normal speech, chat room, tweet, etc.
- **Noisy data**
  - Erroneous data (e.g., misspellings)
  - Misleading (intentionally) data

Morphology

- *The study of the meaningful components of words*
- Processing of words in both their graphemic (i.e., written) and their phonemic (i.e., spoken) form
- A word grammar determines the way words must be constructed
- How much and what sort of information is expressed by morphology differs widely between languages
- *Scope*: human languages and not formal languages (constants + variables)



un – comfort – able

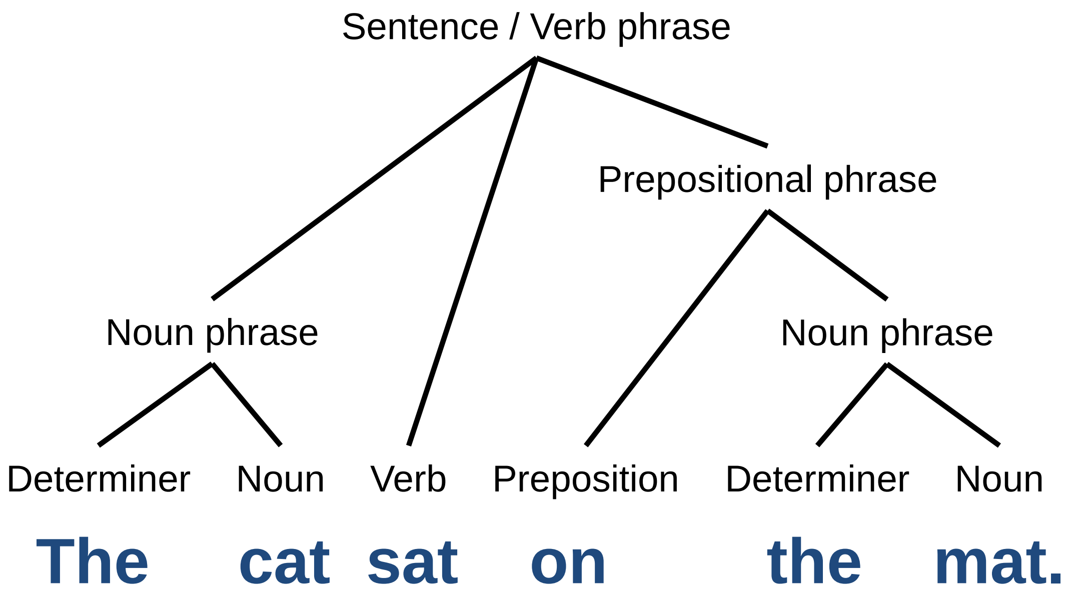
morpheme indicating
a negative meaning

noun

morpheme indicating
an adjective

Syntax

- *The study of the structural relationships between words:*
  - Word order
  - Sentence organization
  - Relationships between word types
- Using a grammar to assign a (more or less detailed) syntactic analysis to a string of words



Sentence / Verb phrase

Prepositional phrase

Noun phrase

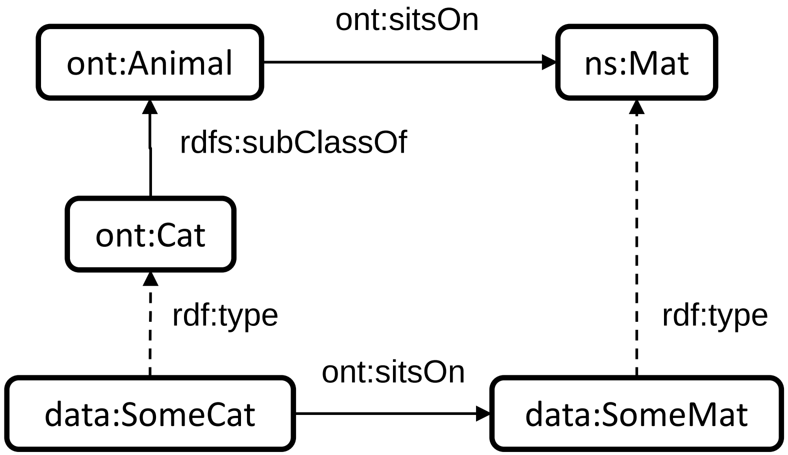
Noun phrase

Determiner Noun Verb Preposition Determiner Noun

The cat sat on the mat.

Semantics

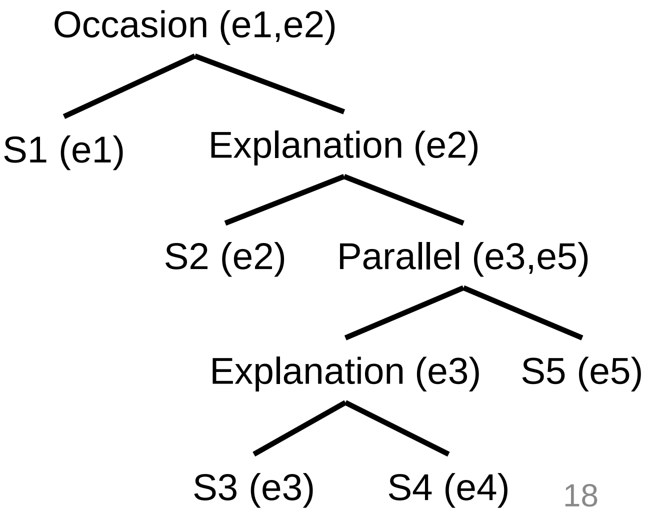
- *The study of meaning*
  - Bridging the gap from linguistic information (in text) to non-linguistic knowledge (of the world)
- Two areas:
  - How meaning is represented
  - How the meaning of sentences is computed systematically from the meaning of its syntactic constituents



Discourse

- *The study of linguistic units larger than a single sentence*
- Discourse: sequence of sentences with the aim of conveying or exchanging information
  - Hard to follow; need links with previous sentences
  - Each participant constructs a discourse model
- Main topics
  - Referring expressions
  - Coherence of discourse
  - Structure of discourse

John went to the bank to deposit his paycheck. (*S1*)
He then took a train to Bill’s car dealership. (*S2*)
He needed to buy a car. (*S3*)
The company he works for now isn’t near any public transportation. (*S4*)
He also wanted to talk to Bill about their softball league. (*S5*)



Occasion (e1,e2)
S1 (e1)
Explanation (e2)
S2 (e2)
Parallel (e3,e5)
Explanation (e3)
S5 (e5)
S3 (e3)
S4 (e4)

Pragmatics

- *The study of how language is used to accomplish goals*
- Explaining the meaning of linguistic messages in terms of their context of use
  - – I.e., disambiguating with contextual setting and intonation

It's cold in here

Semantics:

the temperature in the place is low

Pragmatics:

person A wants person B to close the door

ENCODINGS

Encodings

- Unicode
  - Industry standars to codify, visualize y transmit **text** for most
    - Languages (spoken, but also math or music notation)
    - Tech disciplines (e.g. game tiles, icons, arrows, emoji)
    - Dead languages (e.g. Ancient Greek, Ancient Persian, Ancient Turkish)
  - The last version is [Unicode 14.0.0]() (Sep. 2021)
    - 144.697 characters
    - 159 languages (scripts)

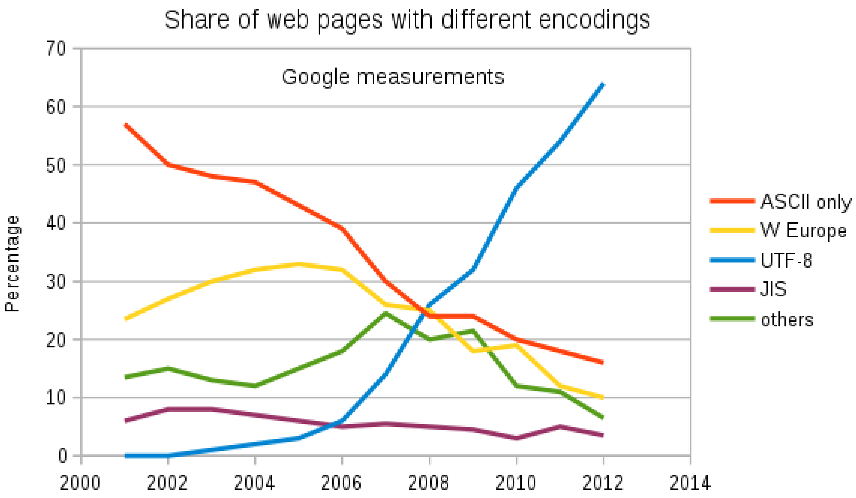
Carácter alfabético latino "A" (U+0041).

Sílaba devanagari "Aum" (ॐ) (U+0950).

Ideograma chino "yue" (月) (U+6708).

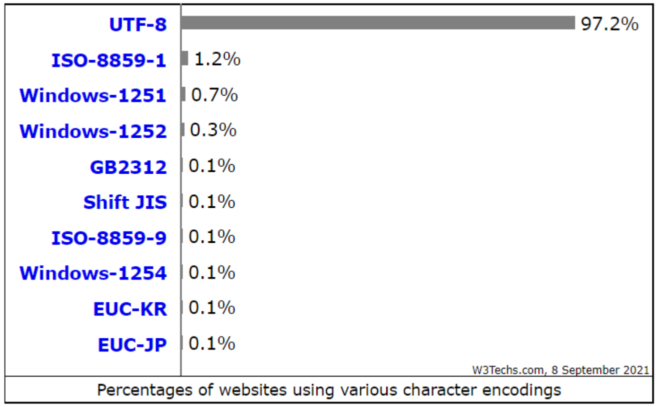
Encodings

- UTF-8
  - Recommended for web and mail
    - In Sep. 2016, 88% web pages used UTF-8
    - In Sep. 2022, it is 97.8% (last data here)
  - Encodes all Unicode characters
  - The encoding (codification)
    - is variable length
      - From 1 to 4 bytes
    - The first 128 characters Unicode are the ASCII characters
      - Letters, numbers
      - And control characters
        - » Carriage return (\r)
        - » End of string (\0)



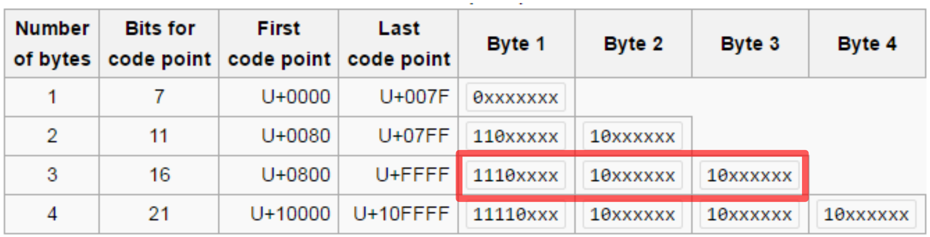
By Chris55 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=51421096

How to read the diagram:
UTF-8 is used by 97.2% of all the websites whose character encoding we know.



Encodings

- Playing with the euro symbol in UTF-8
  - The euro symbol (€) was added to Unicode in version 2.1 (May 1998)
  - Its identifier (code point) is **U+20AC** (“U+” + 4 hexadecimal numbers)
    - DO NO use 4 bytes
    - How many bytes uses?: Follow the table
      - 0800< **20AC** < FFFF ➔ 3 bytes
      - **2** hex → 0010 bin
      - **0** hex → 0000 bin
      - **A** hex → 1010 bin
      - **C** hex → 1100 bin
      - Result: 11100010 10000010 10101100 (Hex: E2 82 AC; Dec: 226, 130, 172)

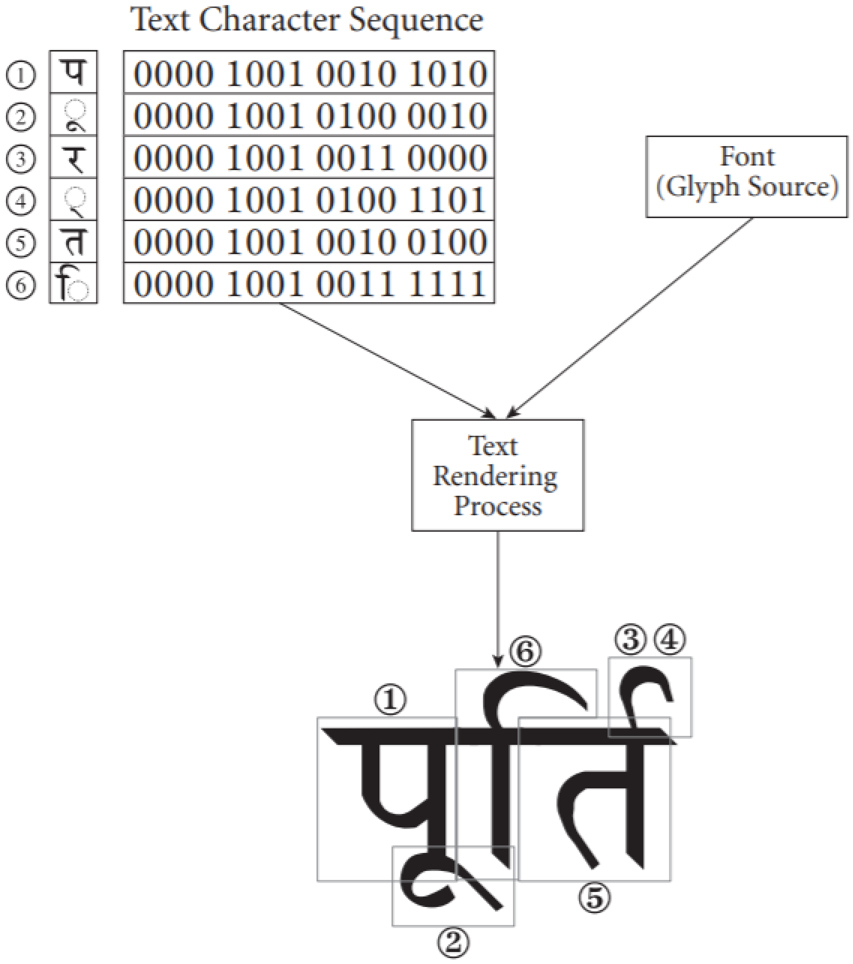


Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Encodings

- A whole world
  - Specification here

Figure 2-3. Unicode Character Code to Rendered Glyphs



Text Character Sequence

①	प	0000 1001 0010 1010
②	◌ू	0000 1001 0100 0010
③	र	0000 1001 0011 0000
④	◌्	0000 1001 0100 1101
⑤	त	0000 1001 0010 0100
⑥	ि◌	0000 1001 0011 1111

Applying encodings

CoNLL-U files

- It is one of the standards to store linguistic information
  - Basic Format
  - Extended Format (CoNLL-U Plus)
- It is a file with
  - UTF-8 encoding
  - A unique line separator: **\n** (LF, *Line Feed*, ASCII char 10)
    - It is not \r (CR, *Carriage Return*, ASCII char 13)
  - Unicode characters with **NFC normalization** . More info.
    - An example for Spanish

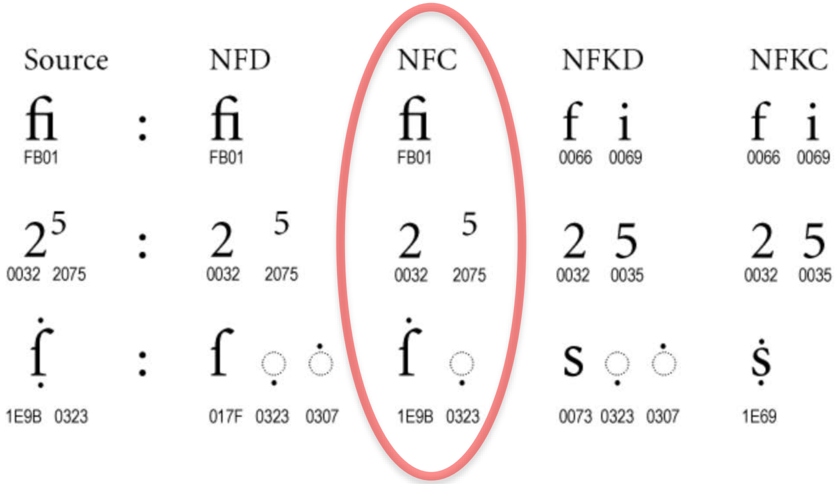
```
 e    +   ´   =  é
U+0065 + U+0301 = U+00E9

"\U0065\U0301" == "\U00e9"
```

¡¡FALSE!!

- Use `utf8::utf8_normalize()`

```
utf8_normalize("\U0065\U0301") == "\U00e9"
```

TRUE



Plural CORPORA

CORPUS

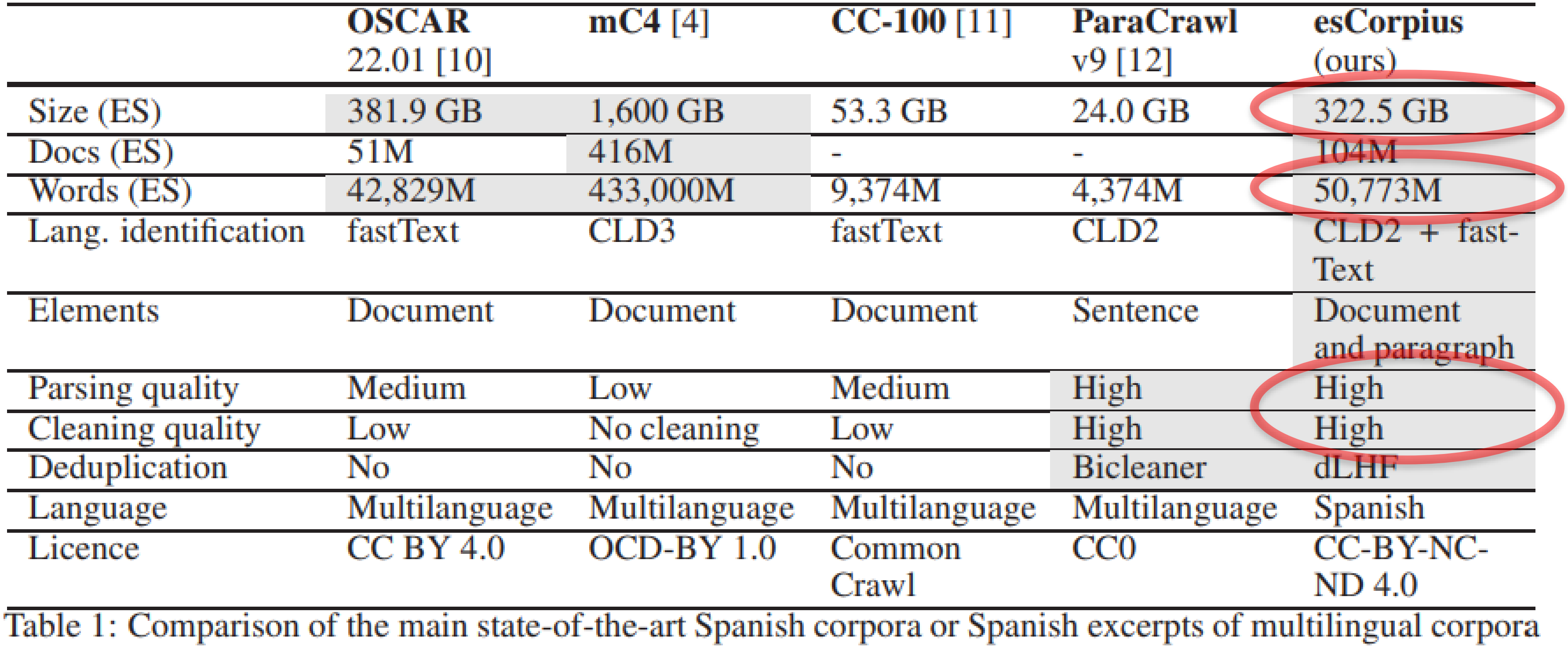
Where can I find texts?

- Project Gutenberg
  - https://www.gutenberg.org
- Project Internet Archive
  - https://archive.org/

  Also has the WayBack Machine (a log of Internet)
- The corpus del español
  - Web site in Spanish and English
  - Its creator is Mark Davies (Univ. Illinois, EE.UU.)
    - Also has corpora for English and Portuguese.
- Leipzig Corpora Collection
  - For Spanish (from Spain and pan-hispanic).
    - Download page (different corpus sizes)
    - Test page (good visualization)

Corpora for Spanish

- From EsCorpius (paper, data)



	OSCAR 22.01 [10]	**mC4** [4]	**CC-100** [11]	**ParaCrawl** v9 [12]	**esCorpius** (ours)
Size (ES)	381.9 GB	1,600 GB	53.3 GB	24.0 GB	322.5 GB
Docs (ES)	51M	416M	-	-	104M
Words (ES)	42,829M	433,000M	9,374M	4,374M	50,773M
Lang. identification	fastText	CLD3	fastText	CLD2	CLD2 + fastText
Elements	Document	Document	Document	Sentence	Document and paragraph
Parsing quality	Medium	Low	Medium	High	High
Cleaning quality	Low	No cleaning	Low	High	High
Deduplication	No	No	No	Bicleaner	dLHF
Language	Multilanguage	Multilanguage	Multilanguage	Multilanguage	Spanish
Licence	CC BY 4.0	OCD-BY 1.0	Common Crawl	CC0	CC-BY-NC-ND 4.0

Table 1: Comparison of the main state-of-the-art Spanish corpora or Spanish excerpts of multilingual corpora

How to manage a corpus

- I will use R
  - Package quanteda
    - Other classical packages: tm
      - Link to hands-on using tm (by Raúl García)
      - Do not support Spanish
  - Package udpipes
  - Package spacyr

Some order in the corpus

NORMALIZATION

What is text normalization?

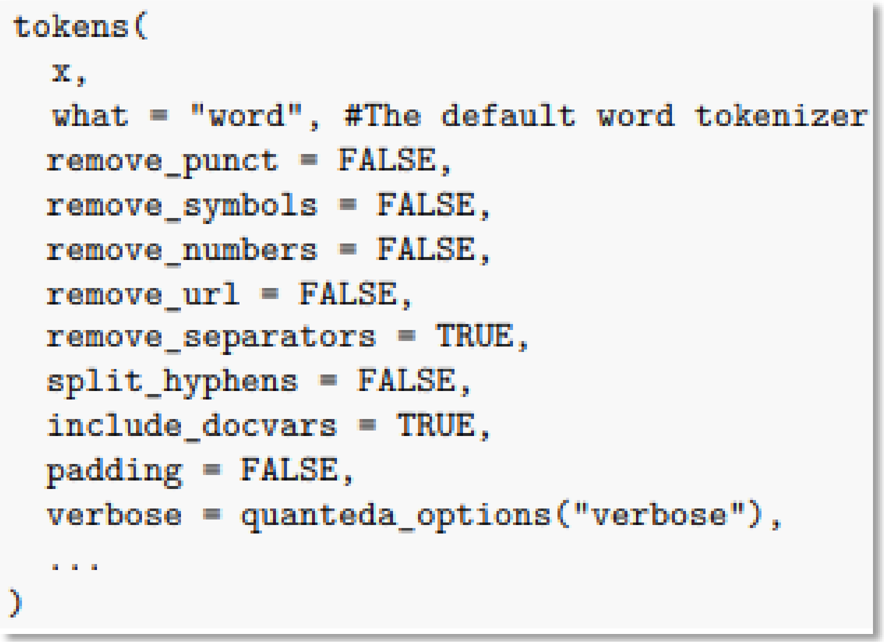
- Put the words in a uniform way to unify equivalent words
  - E.g. in Spanish: **EE.UU., EEUU → Estados Unidos**
  - E.g. in English: **USA**, **U.S.A, US, U.S. → United States of America**
- We only see word normalization (*tokenization*)
  - We will not consider *sentence normalization*
- There is a standard for *tokenization* (*Text Interchange Formats*, TIF) since 2017

Word normalization

- Normalization 1: convert to lowercase
  - Problem: semantics (e.g. in English: **US** and **us** are not the same)

```
tolower(c("US", "us"))
    [1] " us" "us"
```

  - Pay attention to the parameters in function `tokens()` in `quanteda` package



```
tokens(
  x,
  what = "word", #The default word tokenizer
  remove_punct = FALSE,
  remove_symbols = FALSE,
  remove_numbers = FALSE,
  remove_url = FALSE,
  remove_separators = TRUE,
  split_hyphens = FALSE,
  include_docvars = TRUE,
  padding = FALSE,
  verbose = quanteda_options("verbose"),
  ...
)
```

Word normalization

- Normalization 2: **stemming** (keep the root of words)
  - Useful to convert:
    - Plural forms into singular forms (e.g. mice->mouse, casas→casa)
    - Verbal forms into its infinitive form (e.g. went→go, fui→ir)
  - Problem: semantics (e.g. in Spanish: root of **como** ("como es así", relative adverb), **cómo** ("¿cómo estás?", interrogative adverb) and **comí** (from verb "comer") is the same: **com**

```
library(quanteda)
char_wordstem(c("como", "comí", "cómo"),  language = "spanish")
[1] "com" "com" "com"
```

  - To do ***stemming***, *the* Martin Porter algorithm is used
    - Since 2014 (retirment of M. Porter) it is maintained by the community through the Snowball project. For R you have the snowballC package

Word normalization

- Normalization 3: remove **stop words** (frequent words with low relevance)
  - Problem: always double check whether any of them are relevant to the domain in which it is applied
    - e.g.:
      `"to be or not to be, that is the question"`
      →`"question"`

```
library(quanteda)
head(stopwords("spanish") #There are 308
[1] "de"  "la"  "que" "el"  "en"  "y"
```

  - Also package stopwords (uses info from several sources)

```
library(stopwords)
head(stopwords(language = "es", source = "stopwords-iso"))#There are 732
[1] "0" "1" "2" "3" "4" "5"
```

Subword normalization

- Instead of defining tokens from words (white-space segmentation), or from characters in languages without word separators (single-character segmentation), we do the slicing using an algorithm.
- The tokens will be words or chunks of words (subwords, typically morphemes)
- A morpheme is the smallest meaning-bearing unit in a language. E.g .: *disliked* has 3 morphemes: *dis*, *lik*, *ed*
- Advantages:
  - We will know how to process **unknown words** (not in the dictionary)
  - We can have a fix-size vocabulary

Subword normalization Methods

- Three main methods
  - Byte-pair encoding (**BPE**) ← We'll see only this (easiest)
  - Unigram language model (Kudo, 2018)
  - Wordpiece (Schuster and Nakajima, 2012)
- In these three methods there are two elements
  - The **token learner**
    - From a training corpus it creates the vocabulary (the tokens)
    - In Machine Learning jargon this is the ***training phase***
  - The **token segmenter**
    - Takes a sentence as input and *tokenizes* the sentence with the tokens created by the token learner.
    - In *Machine Learning* jargon this is the ***test phase***

Subword normalization
BPE method

- The ***token learner*** does this

  The initial vocabulary V are the **letters** of the words in the training corpus

  $$V = \{A, B, C, \dots a, b, c, \dots\}$$

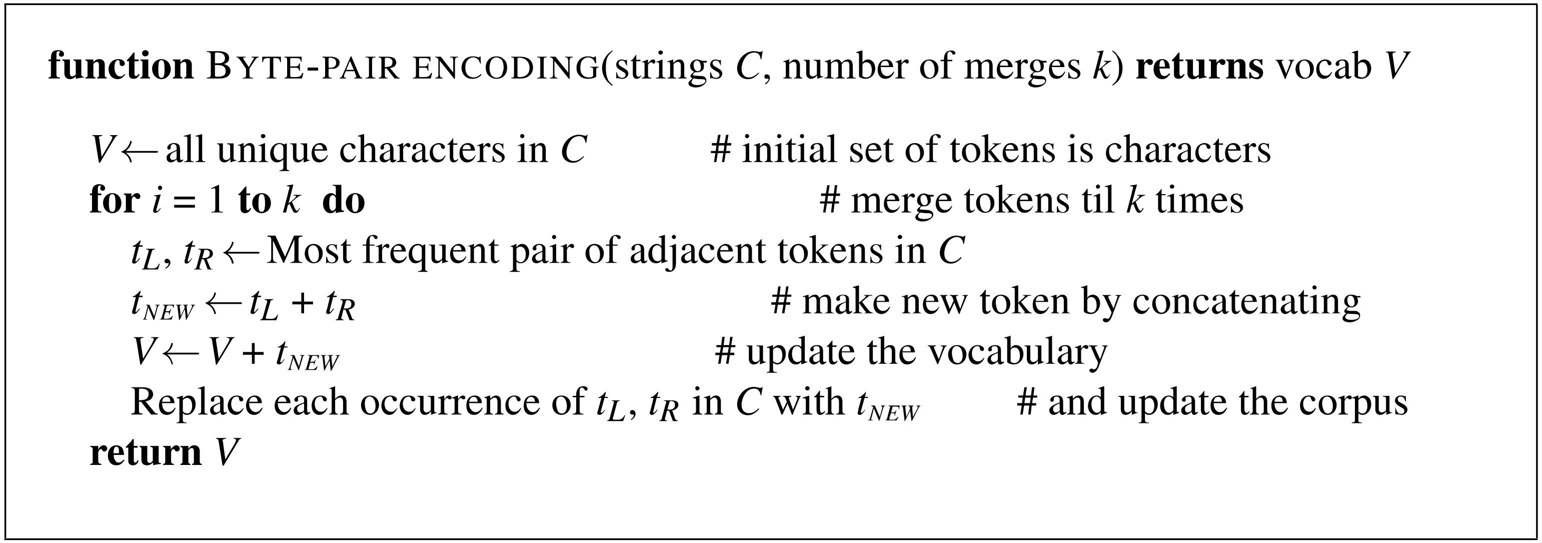
  Repeat:

  - Look for the two adjacent letters that occur most frequently in the training corpus. Let's say they are A and B
  - Add to the vocabulary the new token AB (union of A and B)
  - Replace the adjacency $A \;\; B$ by AB in the training corpus

  Until k joins have been made

Subword normalization
BPE method

- Byte-pair encoding (BPE), de Sennrich *et al.*, 2016
- The *Token learner* as an algorithm:



```
function BYTE-PAIR ENCODING(strings $C$, number of merges $k$) returns vocab $V$

  $V \leftarrow$ all unique characters in $C$            # initial set of tokens is characters
  for $i = 1$ to $k$ do                                   # merge tokens til $k$ times
    $t_L, t_R \leftarrow$ Most frequent pair of adjacent tokens in $C$
    $t_{NEW} \leftarrow t_L + t_R$                         # make new token by concatenating
    $V \leftarrow V + t_{NEW}$                             # update the vocabulary
    Replace each occurrence of $t_L, t_R$ in $C$ with $t_{NEW}$        # and update the corpus
  return $V$
```

Subword normalization
BPE method

- The ***Token learner*** for non-programmer humans:
  - Example of a training corpus (from Jurafsky 2020)

  low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

  $V = \{d,\ e,\ i,\ l,\ n,\ o,\ r,\ s,\ t,\ w\}$

Step 1: In the training corpus we add the _ before each space. Like this:

low_ low_ low_ low_ low_ lowest_ lowest_ newer_ newer_ newer_ newer_ newer_ newer_ wider_ wider_ wider_ new_ new_

$V = \{\_, d,\ e,\ i,\ l,\ n,\ o,\ r,\ s,\ t,\ w\}$

Subword normalization
BPE method

- The ***Token learner*** for non-programmer humans:
  - Example of a training corpus (from Jurafsky 2020)

low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

$$V = \{d,\ e,\ i,\ l,\ n,\ o,\ r,\ s,\ t,\ w\}$$

Step 2: We split each Word by its letters in the result of step 1. Like this:

```
5      l  o  w  _
2      l  o  w  e  s  t  _
6      n  e  w  e  r  _
3      w  i  d  e  r  _
2      n  e  w  _
```

$V = \{\_, d,\ e,\ i,\ l,\ n,\ o,\ r,\ s,\ t,\ w\}$

Subword normalization
BPE method

- The ***Token learner*** for non-programmer humans:
  - Example of a training corpus (from Jurafsky 2020)

low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

$V = \{d,\ e,\ i,\ l,\ n,\ o,\ r,\ s,\ t,\ w\}$

We make joins. Let us join e r to obtain **er** . $k = 1$. This is the result:

5	l o w _
2	l o w e s t _
6	n e w **er** _
3	w i d **er** _
2	n e w _

$V = \{\_, d,\ e,\ i,\ l,\ n,\ o,\ r,\ s,\ t,\ w, \mathbf{er}\}$

Subword normalization
BPE method

- The ***Token learner*** for non-programmer humans:
  - Example of a training corpus (from Jurafsky 2020)

low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

$V = \{d, e, i, l, n, o, r, s, t, w\}$

We make joins. Let us join er and _ to obtain **er_** . $k = 2$. This is the result:

5	l	o	w	_			
2	l	o	w	e	s	t	_
6	n	e	w	**er_**			
3	w	i	d	**er_**			
2	n	e	w	_			

$V = \{\_, d, e, i, l, n, o, r, s, t, w, \text{er}, \textbf{er\_}\}$

Subword normalization
BPE method

- The ***Token learner*** for non-programmer humans:
  - Example of a training corpus (from Jurafsky 2020)

low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

$V = \{d, \ e, \ i, \ l, \ n, \ o, \ r, \ s, \ t, \ w\}$

We make joins. Let us join n and e to obtain **ne** . $k = 3$. This is the result:

5	l o w _
2	l o w e s t _
6	**ne** w er_
3	w i d er_
2	**ne** w _

$V = \{\_, d, \ e, \ i, \ l, \ n, \ o, \ r, \ s, \ t, \ w, \ \text{er}, \text{er\_}, \mathbf{ne}\}$

Subword normalization
BPE method

- The ***Token learner*** for non-programmer humans:
  - Example of a training corpus (from Jurafsky 2020)

low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

$$V = \{d,\ e,\ i,\ l,\ n,\ o,\ r,\ s,\ t,\ w\}$$

We continue making joins:

Join	**Vocabulary**
ne and w	$V = \{\_, d, e, i, l, n, o, r, s, t, w, \mathrm{er}, \mathrm{er\_}\ ,\ \mathrm{new}\}$
l and o	$V = \{\_, d, e, i, l, n, o, r, s, t, w, \mathrm{er}, \mathrm{er\_}\ ,\ \mathrm{new}, \mathrm{lo}\}$
new and er_	$V = \{\_, d, e, i, l, n, o, r, s, t, w, \mathrm{er}, \mathrm{er\_}\ ,\ \mathrm{new}, \mathrm{lo}, \mathrm{low}, \mathrm{newer}\}$
low and _	$V = \{\_, d, e, i, l, n, o, r, s, t, w, \mathrm{er}, \mathrm{er\_}\ ,\ \mathrm{new}, \mathrm{lo}, \mathrm{low}, \mathrm{newer}, \mathrm{low\_}\}$

Subword normalization
BPE method

- The ***Token segmenter*** for non-programmer humans:
  - Applies to test data (not training data)
  - Apply each union (learned in training) to the test data
    - Greedily
    - In the order in which they were learned
  - Note that the frequencies from the test text are not used (they were used in the training phase)
  - Following the previous example:

    $k = 1$: convierts all the occurencies of e followed by r into er
    $k = 2$: convierts all the occurencies of er followed by _ into er_
    $k = 3$: convierts all the occurencies of n followed by e into ne
    …etc.

Subword normalization
BPE method

- The ***Token*** ***segmenter*** for non-programmer humans:
  - Results of the example

    newer is *tokenized* as newer (only one token)

    lower is *tokenized* as low er (two tokens)

Subword normalization
BPE method in R

- For the *token segmenter*, the package sentecepiece implements the BPE method and the method *Unigram language model* (trained with Wikipedia texts)
- For the *token learner* (and also *segmenter*) use the package tokenizers.bpe

```
library("tokenizers.bpe")
modelobpe <- bpe(c("low", "low", "low", "low", "low", "lowest", "lowest",
                   "newer", "newer", "newer", "newer", "newer", "newer",
                   "wider", "wider", "wider", "new", "new"))
bpe_encode(modelobpe, x = "newer lower", type="subwords")
[[1]]
[1] "▁newer" "▁low"   "er"
```

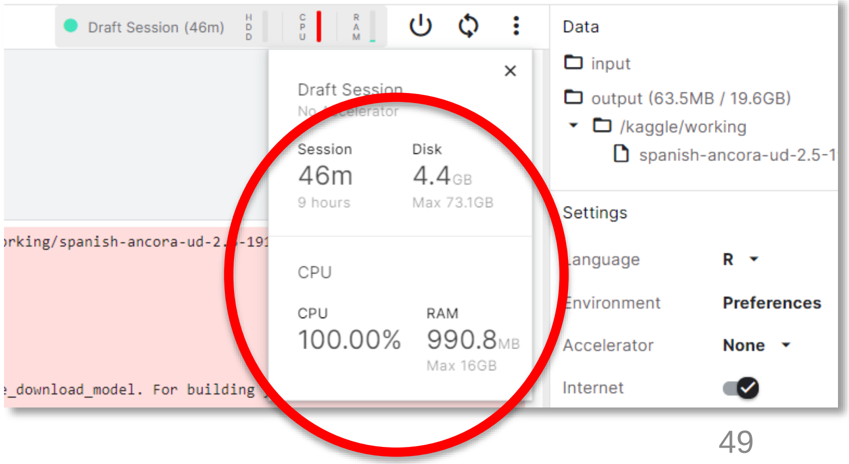
Warning! In this package the _ indicates start of word instead of end.

At last! ☺

HANDS-ON

Software to be used

- **R**
  - Language and environment for statistical computing
  - https://www.r-project.org/
  - GNU GPL
- **RStudio**
  - IDE for R
  - https://www.rstudio.com/
  - Open Source Edition (AGPL v3)
- UPM Remote desktops
  - Windows & Ubuntu
  - 6GB RAM
  - Access to MS OneDrive (1TB)
- Kaggle noteebooks (for R)
  - 16GB RAM
  - GPU (40h/week)



Draft Session (46m)

Draft Session
No Accelerator

Session 46m
9 hours

Disk 4.4GB
Max 73.1GB

CPU

CPU 100.00%

RAM 990.8MB
Max 16GB

```
rking/spanish-ancora-ud-2.-191
_download_model. For building
```

Data

input

output (63.5MB / 19.6GB)

/kaggle/working

spanish-ancora-ud-2.5-1

Settings

Language: R

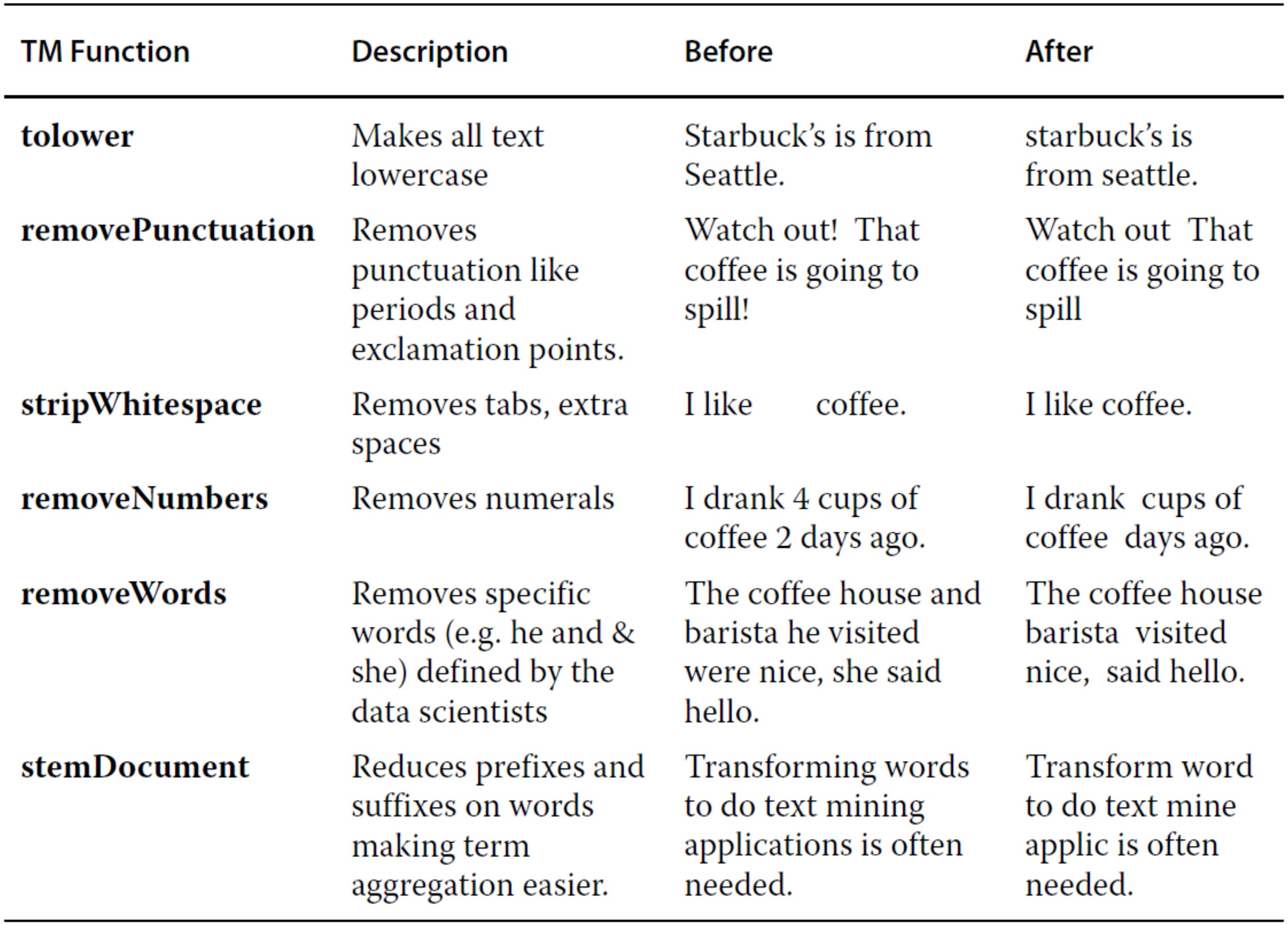
Environment: Preferences

Accelerator: None

Internet: ✓

Package tm

- Basic utilities for text mining



TM Function	Description	Before	After
tolower	Makes all text lowercase	Starbuck's is from Seattle.	starbuck's is from seattle.
removePunctuation	Removes punctuation like periods and exclamation points.	Watch out! That coffee is going to spill!	Watch out That coffee is going to spill
stripWhitespace	Removes tabs, extra spaces	I like coffee.	I like coffee.
removeNumbers	Removes numerals	I drank 4 cups of coffee 2 days ago.	I drank cups of coffee days ago.
removeWords	Removes specific words (e.g. he and & she) defined by the data scientists	The coffee house and barista he visited were nice, she said hello.	The coffee house barista visited nice, said hello.
stemDocument	Reduces prefixes and suffixes on words making term aggregation easier.	Transforming words to do text mining applications is often needed.	Transform word to do text mine applic is often needed.

Source: Text mining in Practice with R (Kwartler 2017)

Course: Intelligent Systems

Unit 4: Language Technologies

Language technologies
Part 1

Mariano Rico
2022
Technical University of Madrid