Course: Intelligent Systems

Unit 4: Language Technologies

Language technologies
Part 2

Mariano Rico
2022
Technical University of Madrid

NLP at a glance

- Session 1 (**29th Nov**)
  - Encodings
  - Corpus
  - Normalization
  - Hands-on 1
- Session 2 (in 2 weeks, **Today**)
  - 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

First of all

- Take the satisfaction survey (30 min) http://servicios.upm.es/encuestas
  - Enter your email (without @alumnos.upm.es) and passwd
  - Evaluate anonymously your teachers
    - Mari Carmen Suárez~~/Asunción Gómez~~
    - Daniel Manrique
    - Martín Molina
    - Mariano Rico

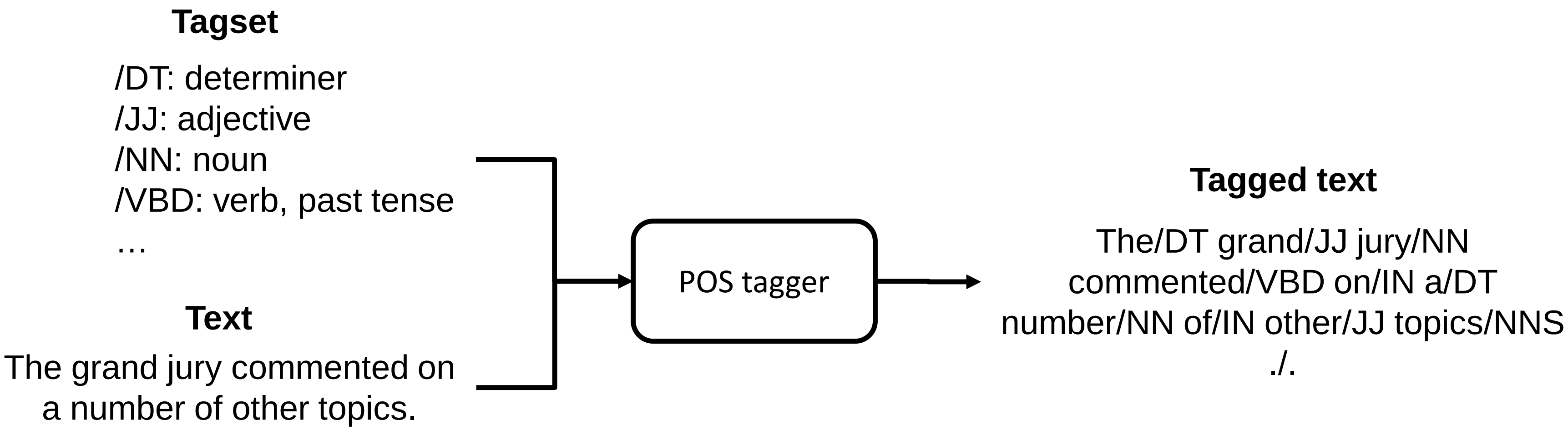
Table of Contents

1. **Part of Speech**
2. **Sparse Vector models**
3. **TF-IDF**
4. **Sentiment Analysis**
5. **Hands-on 2**

PART OF SPEECH

Part-of-speech tagging

- **Part of speech (POS)**:
  - – Noun, verb, pronoun, preposition, adverb, conjunction, participle, article, etc.
- **POS Tagging**
  - – Automatic assignment of part-of-speech descriptors (tags) to input tokens



Tagset

/DT: determiner
/JJ: adjective
/NN: noun
/VBD: verb, past tense
…

Text

The grand jury commented on a number of other topics.

POS tagger

Tagged text

The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

Lexical classes of English words

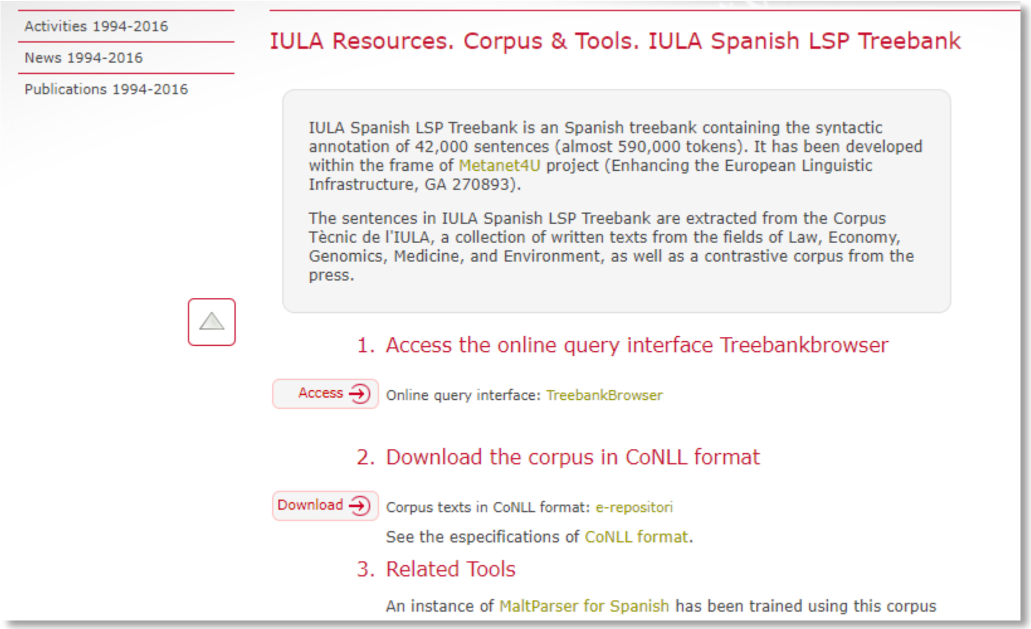
- Two broad categories
  - **Open class types**. Commonly accept the addition of new words
    - Nouns, verbs, adjectives, adverbs
  - **Closed class types**. New words are rarely added
    - Prepositions, determiners, pronouns, conjunctions, etc.
- Others
  - *Interjections* (oh, ah, hey, man, alas, uh, um)
  - *Negatives* (no, not)
  - *Politeness markers* (please, thank you)
  - *Greetings* (hello, goodbye)
  - *The existential there* (there are two on the table)
  - …

Universal Dependencies (UD)

- Framework for a coherent annotation of
  - POS
  - grammar trees
  - Syntactic dependencies
- Created by an open community
  - More than 300 collaborators
  - 200 *treebanks*
  - More than 100 languages
- UD annotations are the evolution of
  - Stanford Universal Dependencies. More info.
  - Google Universal POS. More info.
  - Interlingua from Interset for morph syntactic tagsets. More info.
- Even more info

Treebanks for Spanish

- IULA Spanish LSP Treebank
  - Syntactic annotation of 42.000 phrases (590.000 tokens, 631.642 lines)
    - 41MB (uncompressed) in CONLL format (CONLL tagset)
    - Warning, it is NOT CONLL-U.
  - The corpus contains text from newspapers, and texts from areas like law, economy, medicine, genomics, etc.



IULA Resources. Corpus & Tools. IULA Spanish LSP Treebank

IULA Spanish LSP Treebank is an Spanish treebank containing the syntactic annotation of 42,000 sentences (almost 590,000 tokens). It has been developed within the frame of Metanet4U project (Enhancing the European Linguistic Infrastructure, GA 270893).

The sentences in IULA Spanish LSP Treebank are extracted from the Corpus Tècnic de l'IULA, a collection of written texts from the fields of Law, Economy, Genomics, Medicine, and Environment, as well as a contrastive corpus from the press.

1. Access the online query interface Treebankbrowser

Online query interface: TreebankBrowser

2. Download the corpus in CoNLL format

Corpus texts in CoNLL format: e-repositori

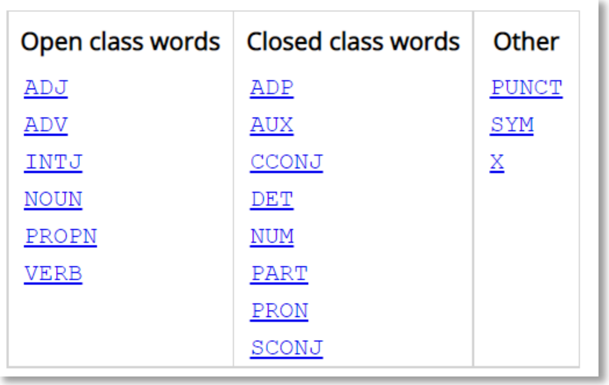
See the especifications of CoNLL format.

3. Related Tools

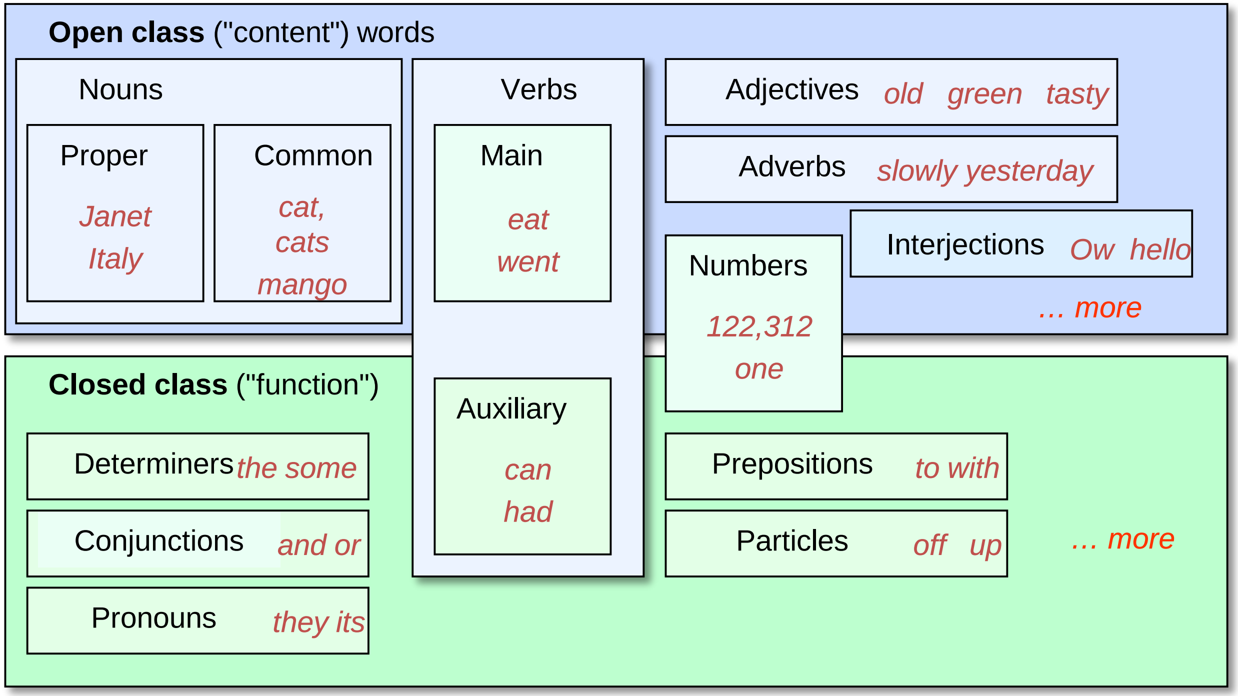
An instance of MaltParser for Spanish has been trained using this corpus

Universal Dependencies (UD)

- T*ags* (labels) for *POS*
  - The most important (*core*)
    - More info
  - Additional properties



Open class words	Closed class words	Other
ADJ	ADP	PUNCT
ADV	AUX	SYM
INTJ	CCONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	



Open class ("content") words

Nouns

Proper *Janet Italy*

Common *cat, cats mango*

Verbs

Main *eat went*

Auxiliary *can had*

Adjectives *old green tasty*

Adverbs *slowly yesterday*

Interjections *Ow hello*

Numbers *122,312 one*

… more

Closed class ("function")

Determiners *the some*

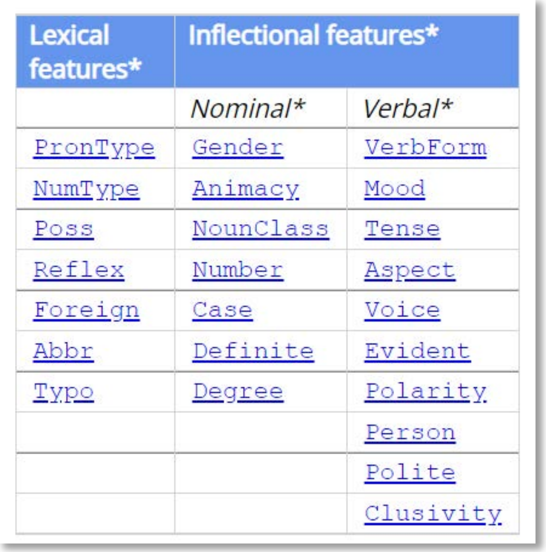
Conjunctions *and or*

Pronouns *they its*

Prepositions *to with*

Particles *off up*

… more

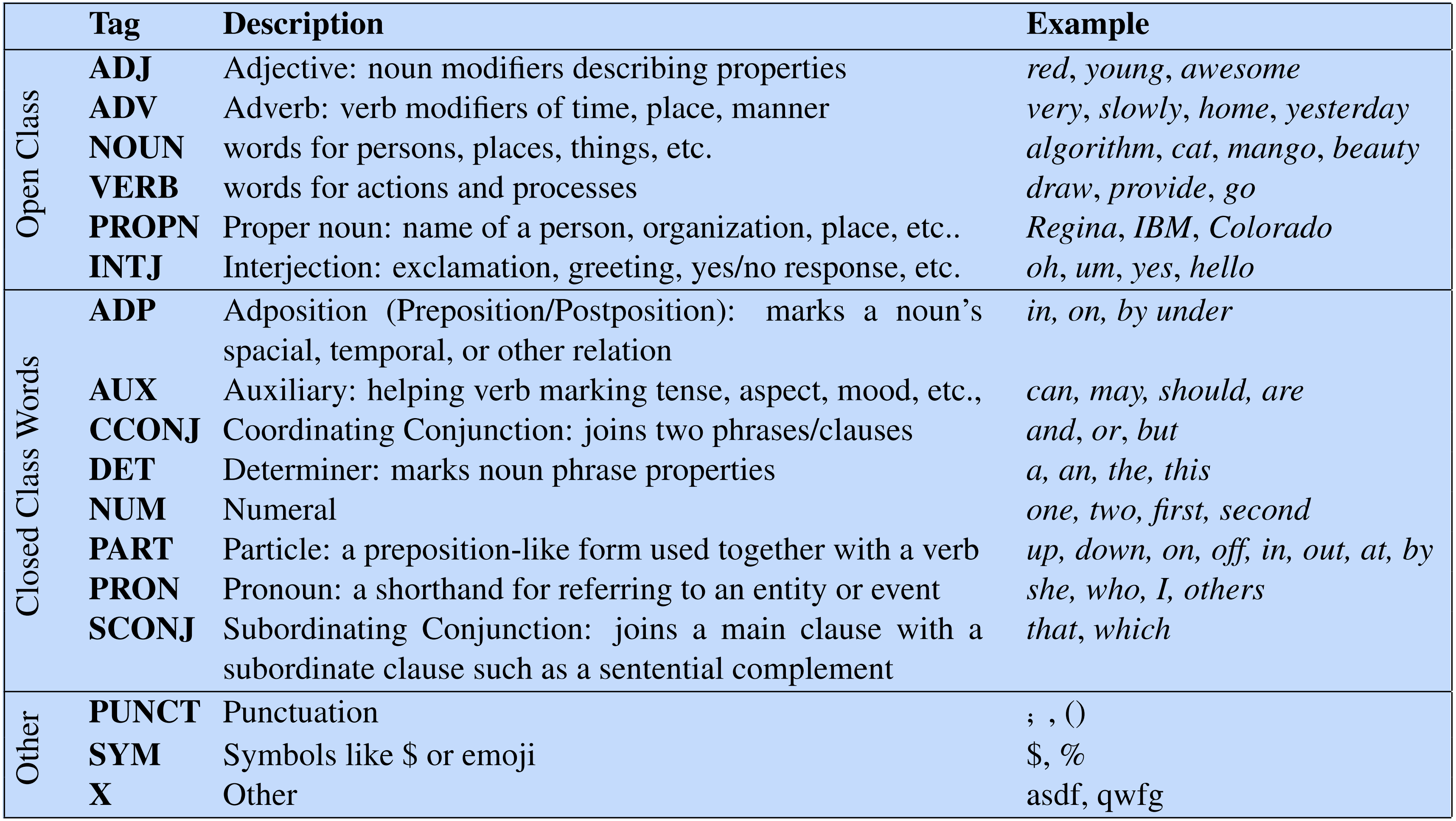


Lexical features*	Inflectional features*	
	*Nominal**	*Verbal**
PronType	Gender	VerbForm
NumType	Animacy	Mood
Poss	NounClass	Tense
Reflex	Number	Aspect
Foreign	Case	Voice
Abbr	Definite	Evident
Typo	Degree	Polarity
		Person
		Polite
		Clusivity

Source: Jurafsky 3rd ed.

Universal Dependencies (UD)

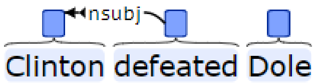
- POS tags in detail (Nivre et al. 2016)



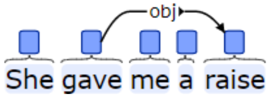
	Tag	Description	Example
Open Class	**ADJ**	Adjective: noun modifiers describing properties	*red*, *young*, *awesome*
	ADV	Adverb: verb modifiers of time, place, manner	*very*, *slowly*, *home*, *yesterday*
	NOUN	words for persons, places, things, etc.	*algorithm*, *cat*, *mango*, *beauty*
	VERB	words for actions and processes	*draw*, *provide*, *go*
	PROPN	Proper noun: name of a person, organization, place, etc..	*Regina*, *IBM*, *Colorado*
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	*oh*, *um*, *yes*, *hello*
Closed Class Words	**ADP**	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	*in, on, by under*
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	*can, may, should, are*
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	*and*, *or*, *but*
	DET	Determiner: marks noun phrase properties	*a, an, the, this*
	NUM	Numeral	*one, two, first, second*
	PART	Particle: a preposition-like form used together with a verb	*up, down, on, off, in, out, at, by*
	PRON	Pronoun: a shorthand for referring to an entity or event	*she, who, I, others*
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	*that*, *which*
Other	**PUNCT**	Punctuation	; , ()
	SYM	Symbols like $ or emoji	$, %
	X	Other	asdf, qwfg

Universal Dependencies (UD)

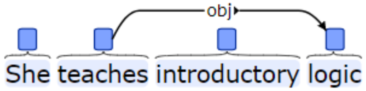
- T*ags* for **relations**
  - The most relevant:
    - nsubj: the subject



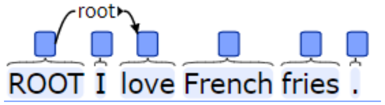
    - obj: the direct object

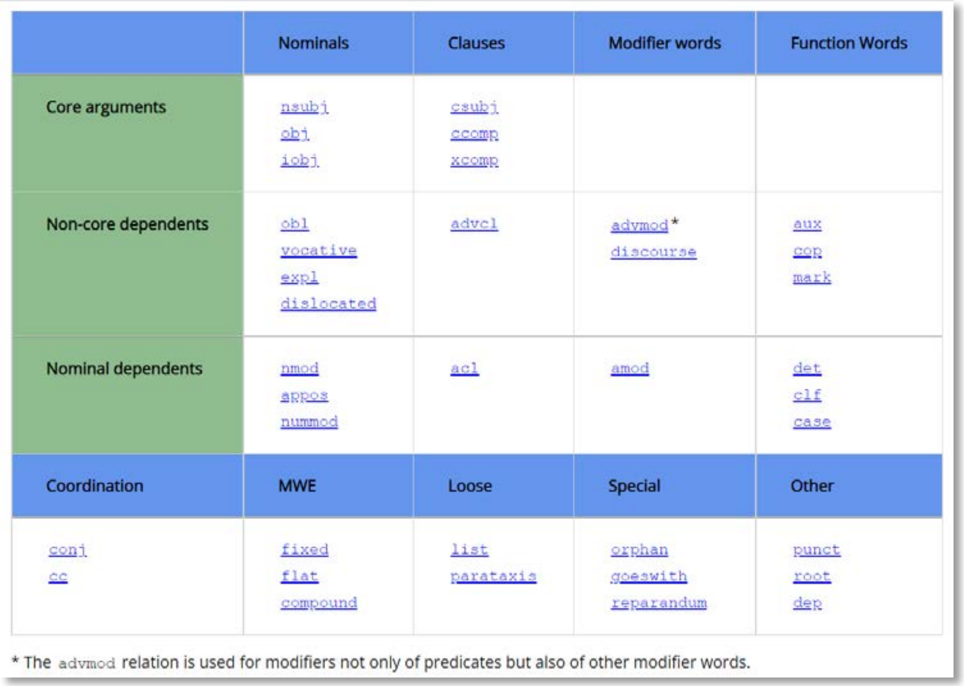


    - iobj: the indirect objet



    - root: the verb
      - Not represented explicitly in CoNLL-U





	Nominals	Clauses	Modifier words	Function Words
Core arguments	nsubj obj iobj	csubj ccomp xcomp		
Non-core dependents	obl vocative expl dislocated	advcl	advmod* discourse	aux cop mark
Nominal dependents	nmod appos nummod	acl	amod	det clf case
Coordination	**MWE**	**Loose**	**Special**	**Other**
conj cc	fixed flat compound	list parataxis	orphan goeswith reparandum	punct root dep

* The advmod relation is used for modifiers not only of predicates but also of other modifier words.

UD from R

```
library(udpipe)
model <- udpipe_download_model(language = "spanish-ancora") #Alternative: "spanish-gsd"
udmodel_es <- udpipe_load_model(file = model$file_model)

txt <- c("En un lugar de La Mancha, Don Quijote y Sancho esperaban a Cervantes.")
anno <- udpipe_annotate(udmodel_es, x = txt)
df <- as.data.frame(anno)
#Has 14 columns doc_id, paragraph_id, sentence_id, sentence, token_id, token,
#                 lemma,          upos,        xpos,    feats,   head_token_id,
#               dep_rel,          deps,        misc
df[,5:14]
```

```
   token_id     token     lemma  upos  xpos                                                 feats head_token_id dep_rel deps           misc
1         1        En        en   ADP   ADP                                          AdpType=Prep             3    case <NA>           <NA>
2         2        un       uno   DET   DET     Definite=Ind|Gender=Masc|Number=Sing|PronType=Art             3     det <NA>           <NA>
3         3     lugar     lugar  NOUN  NOUN                               Gender=Masc|Number=Sing            12     obl <NA>           <NA>
4         4        de        de   ADP   ADP                                          AdpType=Prep             6    case <NA>           <NA>
5         5        La        el   DET   DET      Definite=Def|Gender=Fem|Number=Sing|PronType=Art             6     det <NA>           <NA>
6         6    Mancha    Mancha PROPN PROPN                                                  <NA>             3    nmod <NA>  SpaceAfter=No
7         7         ,         , PUNCT PUNCT                                        PunctType=Comm             3   punct <NA>           <NA>
8         8       Don       Don PROPN PROPN                                                  <NA>            12   nsubj <NA>           <NA>
9         9   Quijote   Quijote PROPN PROPN                                                  <NA>             8    flat <NA>           <NA>
10       10         y         y CCONJ CCONJ                                                  <NA>            11      cc <NA>           <NA>
11       11    Sancho    Sancho PROPN PROPN                                                  <NA>             8    conj <NA>           <NA>
12       12 esperaban   esperar  VERB  VERB Mood=Ind|Number=Plur|Person=3|Tense=Imp|VerbForm=Fin             0    root <NA>           <NA>
13       13         a         a   ADP   ADP                                          AdpType=Prep            14    case <NA>           <NA>
14       14 Cervantes Cervantes PROPN PROPN                                                  <NA>            12     obj <NA>  SpaceAfter=No
15       15         .         . PUNCT PUNCT                                        PunctType=Peri            12   punct <NA> SpacesAfter=\\n
```

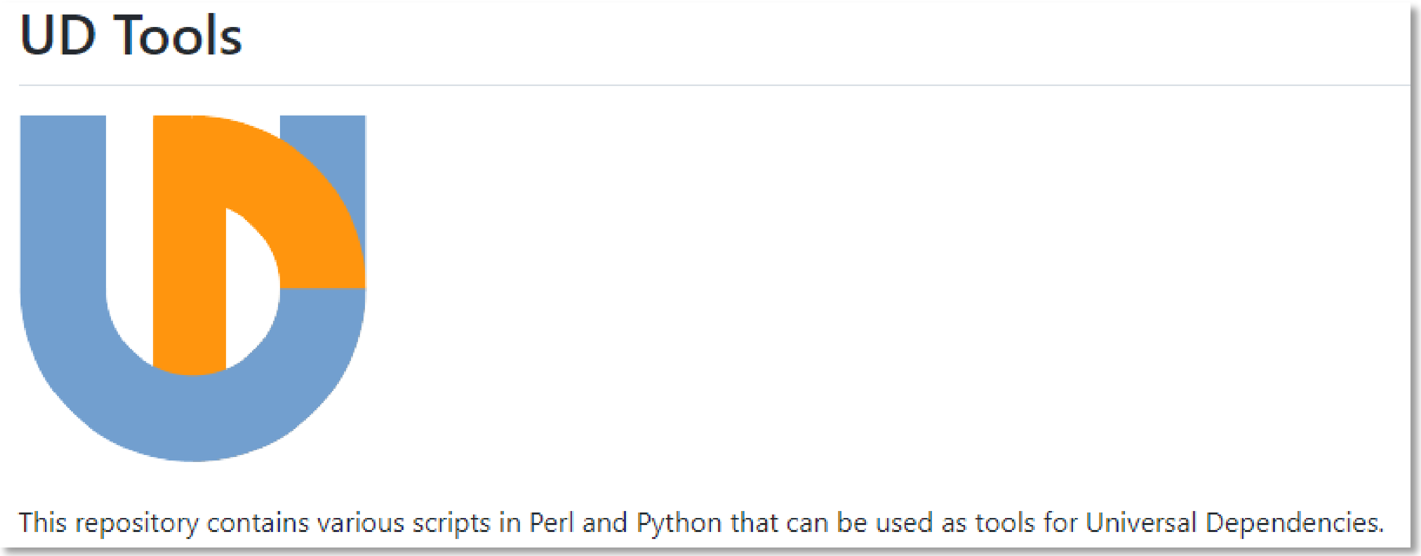
¡Warn!, anno is a list containing 3 things (the last two things were lost when converted to dataframe):

1) x: The x character vector with text.
2) conllu: annotation in CONLL-U format
3) error: A vector with the same length of x containing possible errors when annotating x

```
cat(anno$conllu, file = "my_annotacion.conllu") #You can load it with udpipe_read_conllu()
```

CoNLL-U tools

- UniversalDependencies/Tools
  - Relevant command line tools
    - `validate.py` Verifies that a file is CoNLL-U
    - `normalize_Unicode.pl` Convierta UTF-8 to NFC format
    - `conllu_to_conllx.pl` Convierts from CoNLL-U to the previous format (CoNLL-X) that some tools still use
    - `restore_conlu_lines.pl` Joins a CoNLL-U file with a CoNLL-X, returning a CoNLL-U file

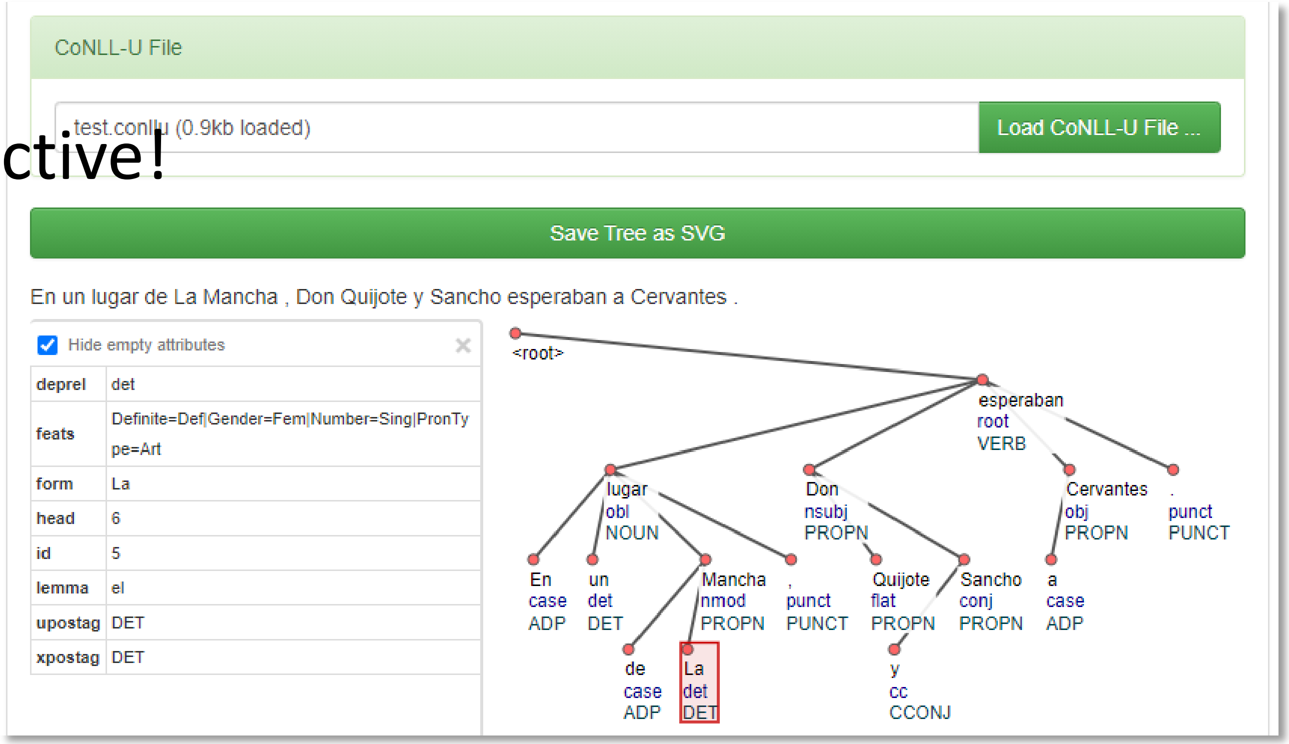


UD Tools

This repository contains various scripts in Perl and Python that can be used as tools for Universal Dependencies.

Playing with CoNLL-U files (1/2)

- CoNLL-U Viewer
  - One of the tools in UD
  - URL: https://universaldependencies.org/conllu_viewer.html
  - It is interactive!



test.conllu (0.9kb loaded)

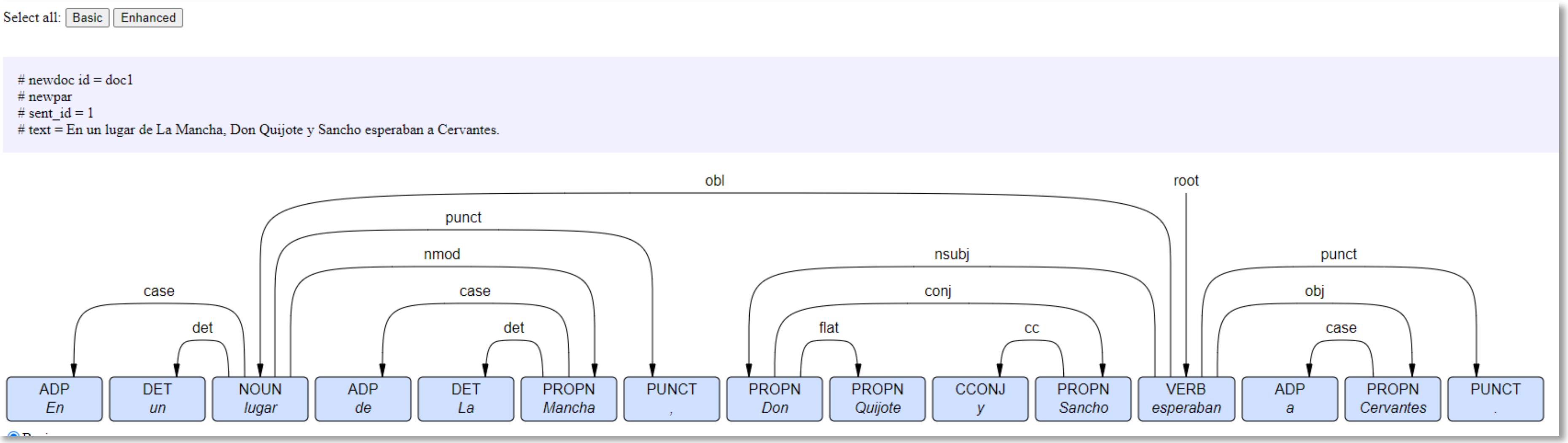
En un lugar de La Mancha , Don Quijote y Sancho esperaban a Cervantes .

☑ Hide empty attributes

deprel	det
feats	Definite=Def|Gender=Fem|Number=Sing|PronType=Art
form	La
head	6
id	5
lemma	el
upostag	DET
xpostag	DET

Playing with CoNLL-U files (2/2)

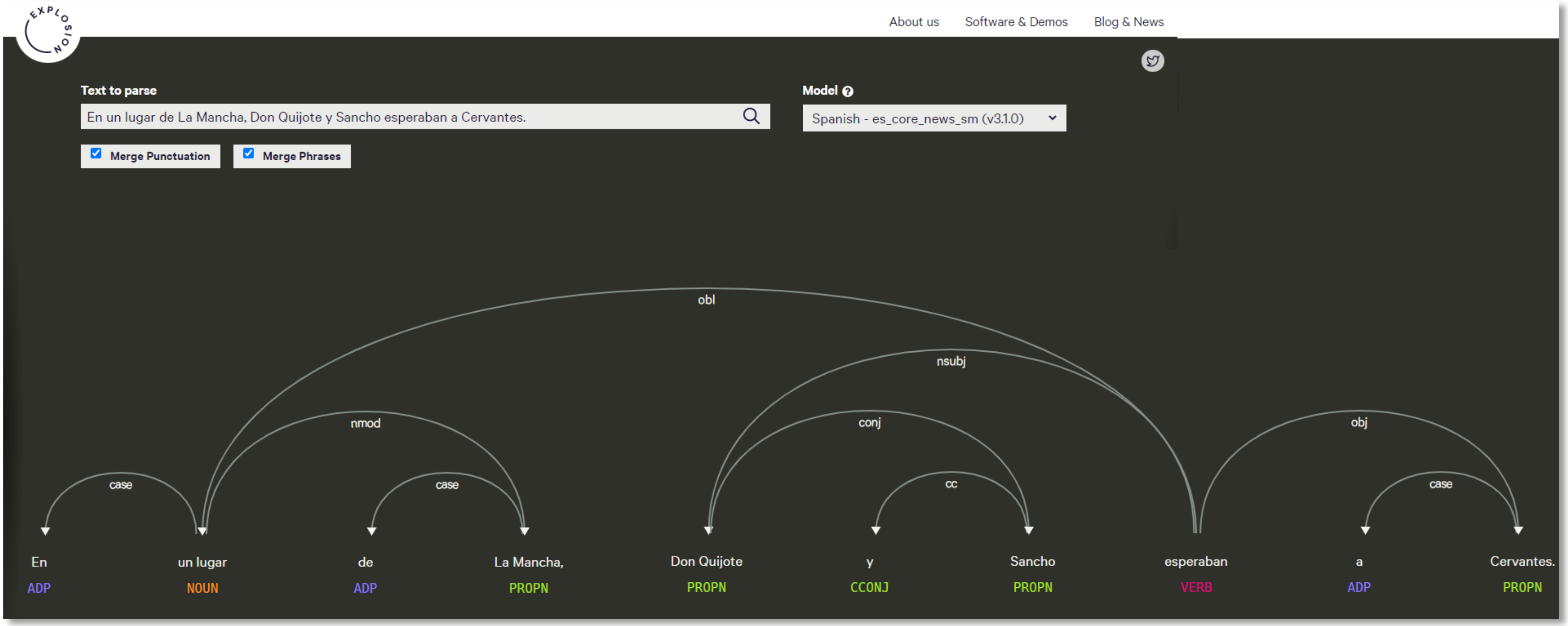
- The tool created by Kleiweg
  - Developed at Univ. Groningen
  - URL: https://urd2.let.rug.nl/~kleiweg/conllu
  - Load the file created previously



```
# newdoc id = doc1
# newpar
# sent_id = 1
# text = En un lugar de La Mancha, Don Quijote y Sancho esperaban a Cervantes.
```

Dependencies with SpaCy

- SpaCy (spacy.io) now it is explosion.ai
- Web app to test dependencies
  - – For Spanish only has the sm(all) model
- Spacy is faster than UD



Text to parse

En un lugar de La Mancha, Don Quijote y Sancho esperaban a Cervantes.

Model

Spanish - es_core_news_sm (v3.1.0)

☑ Merge Punctuation ☑ Merge Phrases

Processing the parse tree

- You can be interested in finding specific syntactical structures
  - E.g.: find all tokens where upos is “VERB”, and that have a child with the relation “nsubj” AND a child with the relation “obj”.
- In R you can use the package rsyntax

Evaluating POS taggers

- **Tagset metrics**
  - *Informativeness*. Not easy to measure; rough measures:
    - Size of the tagset
    - Amount of ambiguity present in the input
  - *Specificability*. Degree to which different linguists uniformly use the tagset when independently tagging the same texts
- **Tagger metrics** (using a benchmark corpus)
  - *Precision/accuracy*
  - *Recall*
  - *Error rate*
  - *Ambiguity*. Average number of analyses in the tagger's output

POS tagging applications

- **Syntax parsing**
  - Basic unit for parsing
- **Information extraction**
  - Indication of names, relations
- **Machine translation**
  - The meaning of a particular word depends on its POS tag
- **Sentiment analysis**
  - Adjectives are the major opinion holders
    - Good vs Bad, Excellent vs Terrible
- **Linguistic studies**
  - Thanks to large tagged text corpora
- …

Table of Contents

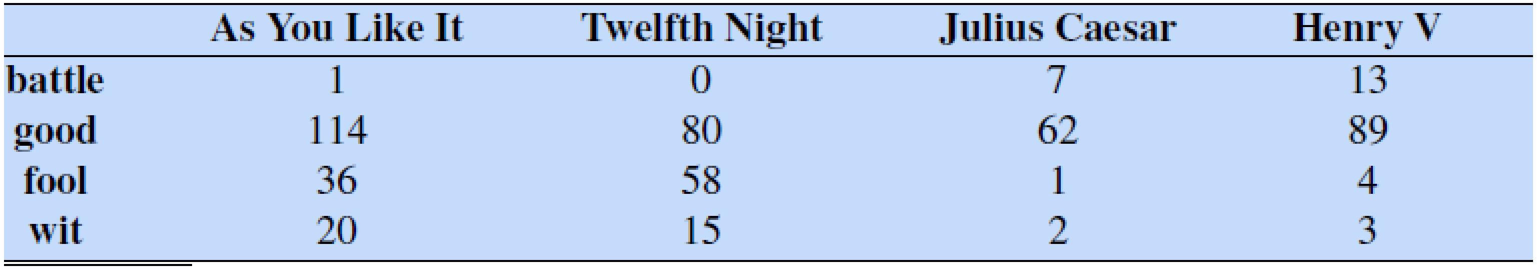
1. Part of Speech
2. **Sparse Vector models**
3. TF-IDF
4. Document classification
5. Hands-on 2

SPARSE VECTOR MODELS

The term-document matrix

- Each row is a word (token) in the vocabulary
- Each columns is a document in the corpus
- The cell value is the number of occurrences of the word in the document
  - Example: 4 plays by Shakespeare

Occurrence table

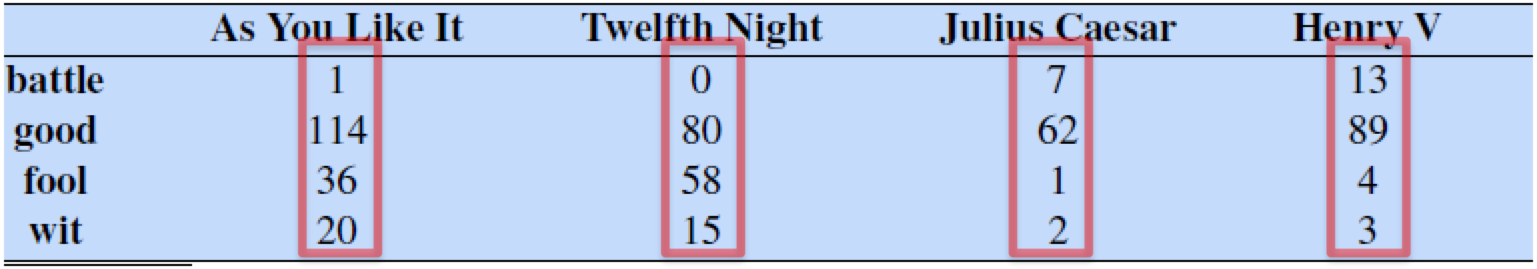


	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

The term-document matrix

- Each row is a word (token) in the vocabulary
- Each columns is a document in the corpus
- The cell value is the number of occurrences of the word in the document
  - – Example: 4 plays by Shakespeare

Occurrence table



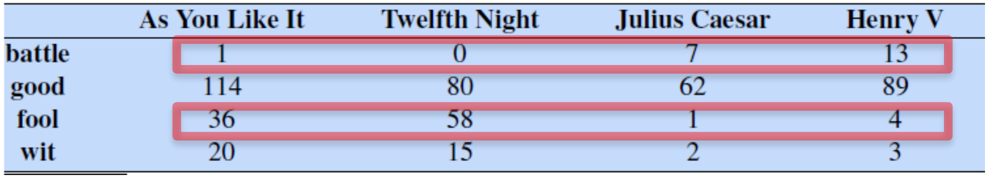
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

A document is characterized by a vector (4 dimensions in this case)

The term-document matrix

- Let us make a projection to 2 dimensions
  - Over any 2 axis in the space
  - Example: over axis fool and battle

Occurrence table

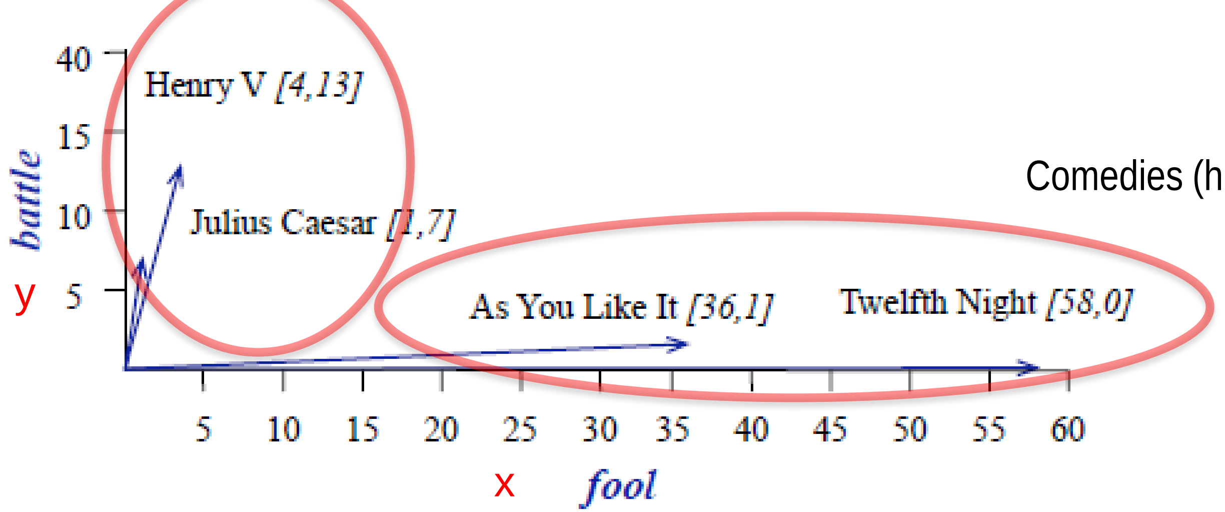


	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

y

x

Epic plays (high values of *battle*)



Comedies (high values of *fool*)

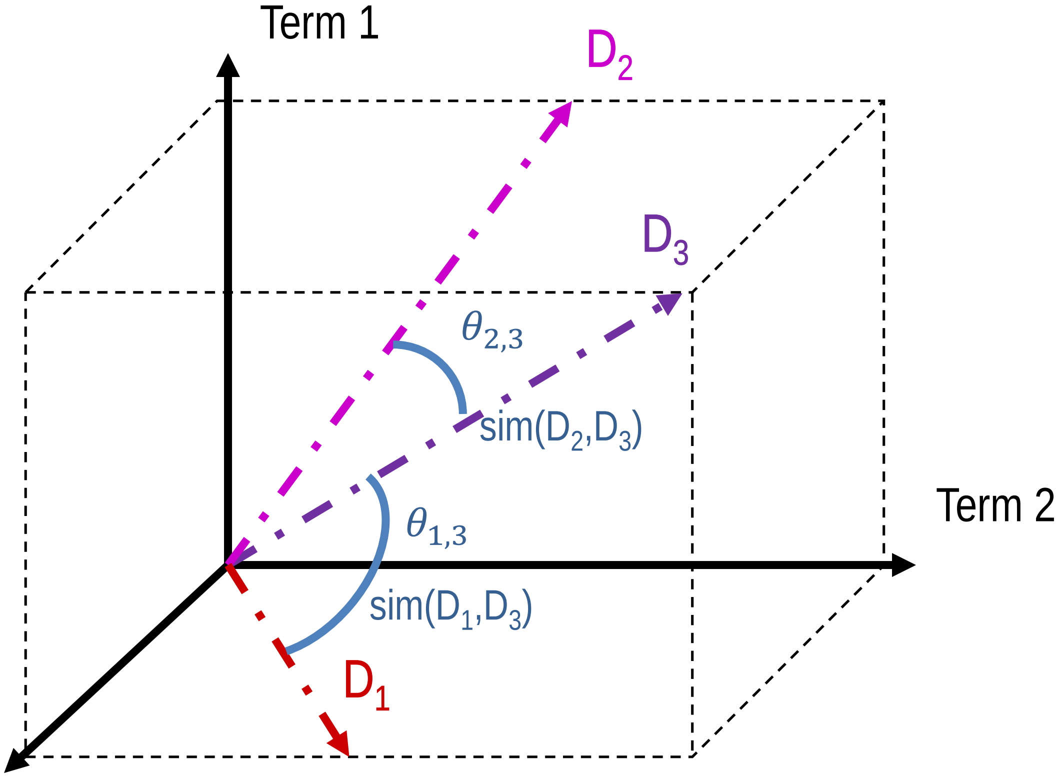
Semantic similarity

Similarity between words and vectors

- Operation with two vectors: dot product $a \cdot b$
  - We have to normalize the vectors (more words frequency do not implies more similarity)

$$\frac{a \cdot b}{|a|\ \ |b|}$$

  - It is the $\cos\theta$
  - As occurrences are always positive, the value of $\cos\theta$ is always between 0 and 1.



Semantic similarity

Similarity between words and vectors

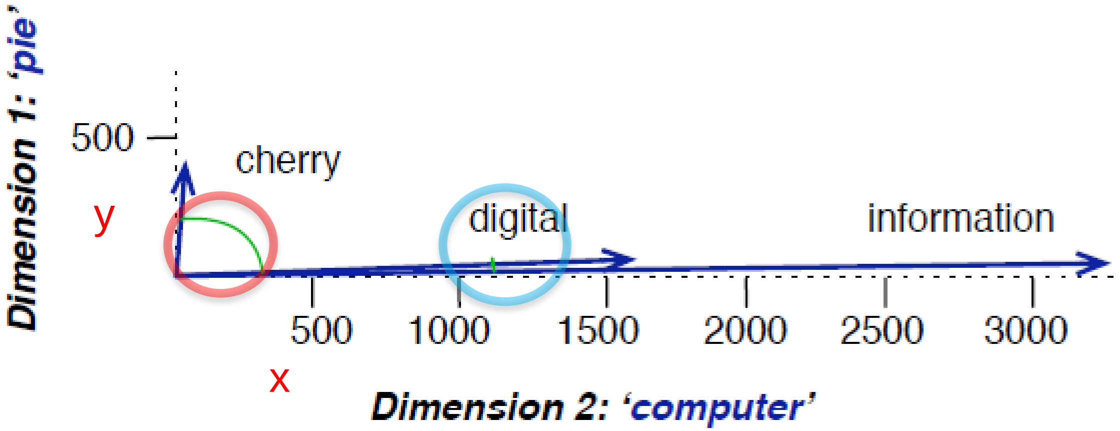
- An example (Jurafsky 2021)

Occurrence table over the dimensions of the columns

	y		x
	pie	**data**	**computer**
cherry	442	8	2
digital	5	1683	1670
information	5	3982	3325

$$\cos(\text{cherry}, \text{information}) = \frac{442 * 5 + 8 * 3982 + 2 * 3325}{\sqrt{442^2 + 8^2 + 2^2}\sqrt{5^2 + 3982^2 + 3325^2}} = .017$$

$$\cos(\text{digital}, \text{information}) = \frac{5 * 5 + 1683 * 3982 + 1670 * 3325}{\sqrt{5^2 + 1683^2 + 1670^2}\sqrt{5^2 + 3982^2 + 3325^2}} = .996$$



Semantic similarity
TF-IDF matrix

- TF from *term-frequency*
  - A Word occurring 100 times in a document is not 100 times more important than a word occurring only once
  - Calculate the **matrix** tf so:

    $$tf_{t,d} = \log_{10}(1 + occurrences(t, d))$$

    If $occurrences(t, d) = 0$ then $tf_{t,d} = 0$
- - it’s a hyphen, not a minus
- IDF from *inverse document frequency*
  - Gives a higher weight to words occurring only in some documents (valuable words for charactering)
  - Calculate the **vector** idf_t (it is not a matrix) so:

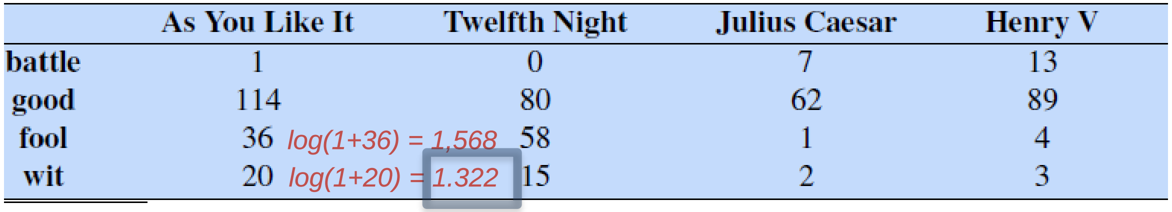
$$idf_t = \log_{10}\left(\frac{N}{df_t}\right), \text{ where } \begin{cases} N & \text{number of documents in the corpus} \\ df_t & \text{number of documents containing } t \end{cases}$$

Semantic similarity

TF-IDF matrix

Example with plays by Shakespeare

Occurrence table



	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

log(1+36) = 1,568

log(1+20) = 1.322

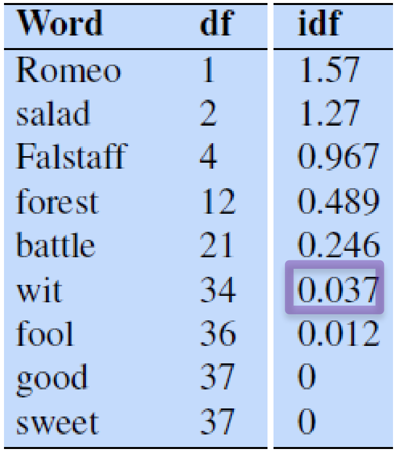
Matrix *term-frequency*

(the cell’s value is the number of occurrences of the term (word in the row) in the document of the column). Let’s compute:

$$tf_{t,d} = \log_{10}(1 + occurrencies(t, d))$$

Vector *df*

(number of documents containing the word)



Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

log(N/df)

= log(37/1) = 1.57

= log(37/2) = 1.27

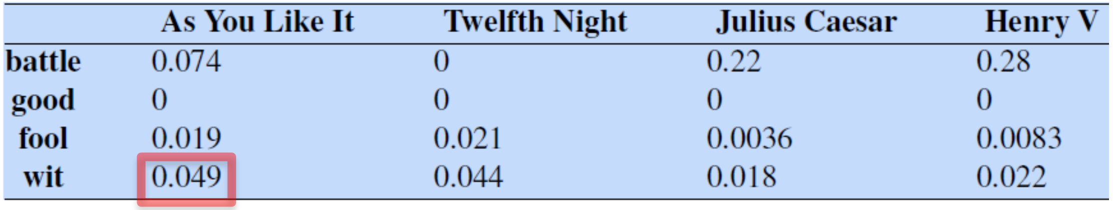
= log(37/4) = 0.967

etc…

$$idf_t = \log_{10}\left(\frac{N}{df_t}\right)$$

We know that N (number of plays) is 37

TF-IDF matrix: $w_{t,d} = tf_{t,d} * \ idf_t$



	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

$$w_{wit,As\ You\ Like\ It} = tf_{wit,As\ You\ Like\ It} * \ idf_{wit}$$

$$= \boxed{1.322} * \boxed{0.037} = \boxed{0.049}$$

Semantic similarity
TF-IDF matrix

- From R
  - The `quanteda` package computes the tf-idf matrix from a given corpus
    - Function `textstat_simil()` returns a matrix of similarities
    - Function `textstat_dist()` returns a matrix of distances
      - With distances you can do dendrograms

Table of Contents

1. Part of Speech
2. Sparse Vector models
3. TF-IDF
4. **Sentiment analysis**
5. Hands-on 2

SENTIMENT ANALYSIS

Sentiment analysis

- It is a case of text classification
  - Sentiment **Polarity**
    - Each text has a **label**: A or B (binary)
      - In favor of A, against A (in favor of B)
      - I like A, I do not like A (I like B)
      - Republican, monarchist
      - Spam, not spam
  - Sentiment **Valence**
    - Each text has a **number**
      - Examples:
        - » Evaluation “starts”: from 1 (I like not much) to 5 (I like very much)
        - » Continuous variables: between 0.0 and 1.0

Polar sentiments

- Given a dictionary of polar words
  - We compute the **polarity** of any text
    - Counting the occurrences of words classified as positive (npos) and negative words (nneg)
    - using an evaluation function
      - Typical functions: `log(npos/nneg)` (so-called "logit" scale)
        - » With quanteda, use the function textstat_polarity()

```
library("quanteda") #Contains the corpus data_corpus_inaugural
library("quanteda.sentiment") #Has several sentiment dictionaries
#One of these is data_dictionary_geninqposneg (General Inquirer dictionary positive-negative)
print(data_dictionary_geninqposneg, max_nval = 5)
Dictionary object with 2 key entries.
Polarities: pos = "positive"; neg = "negative"
- [positive]:
  - abide, ability, able, abound, absolve [ ... and 1,648 more ]
- [negative]:
  - abandon, abandonment, abate, abdicate, abhor [ ... and 2,005 more ]
#We calculate sentiments for corpus texts
tail(data_corpus_inaugural) %>%
  textstat_polarity(dictionary = data_dictionary_geninqposneg)
##           doc_id sentiment
## 1      2001-Bush 0.9233579
## 2      2005-Bush 0.9829457
## 3     2009-Obama 0.5666378
## 4     2013-Obama 0.7597420
```

Sentiments with valence

- Given a dictionary of words with valence
  - We compute the **valence** of any text
    - Calculating the average valence of its words (for a given **sentiment**)

```
library("quanteda.sentiment") #Has several sentiment dictionaries
#One of them is data_dictionary_ANEW (Affective Norms for English Words)
print(data_dictionary_ANEW, max_nval = 5)
Dictionary object with 3 key entries.
Valences set for keys: pleasure, arousal, dominance
- [pleasure]:
  - abduction, able, abortion, absent, absurd [ ... and 2,466 more ]
- [arousal]:
  - abduction, able, abortion, absent, absurd [ ... and 2,466 more ]
- [dominance]:
  - abduction, able, abortion, absent, absurd [ ... and 2,466 more ]
lapply(valence(data_dictionary_ANEW), head, 8) #Print the valence of dictionary words
$pleasure
abduction      able  abortion    absent    absurd abundance     abuse    accept 
     2.76      6.74      3.50      3.69      4.26      6.59      1.80      6.80 

$arousal
abduction      able  abortion    absent    absurd abundance     abuse    accept 
     5.53      4.30      5.39      4.73      4.36      5.51      6.83      5.53 

$dominance
abduction      able  abortion    absent    absurd abundance     abuse    accept 
     3.49      6.83      4.59      4.35      4.73      5.80      3.69      5.41 
tail(data_corpus_inaugural) %>%  #Computes the valence of corpus texts
  textstat_valence(dictionary = data_dictionary_ANEW["pleasure"])
       doc_id sentiment
1      2001-Bush  6.091330
2      2005-Bush  6.308839
```

On sentiments

- The 8 Plutchik's emotions (1980)
  - Anger (*ira*), fear (*miedo*), Sadness (*tristeza*), disgust (*aversión*), surprise (*sorpresa*), anticipation (*anticipación*), trust (*confianza*) and joy (*alegría*).
  - 3 intensity levels for each emotion
- Like colors, emotions can be combined
  - E.g.: joy+trust = love
- Advanced systems can measure these emotions and their intensity

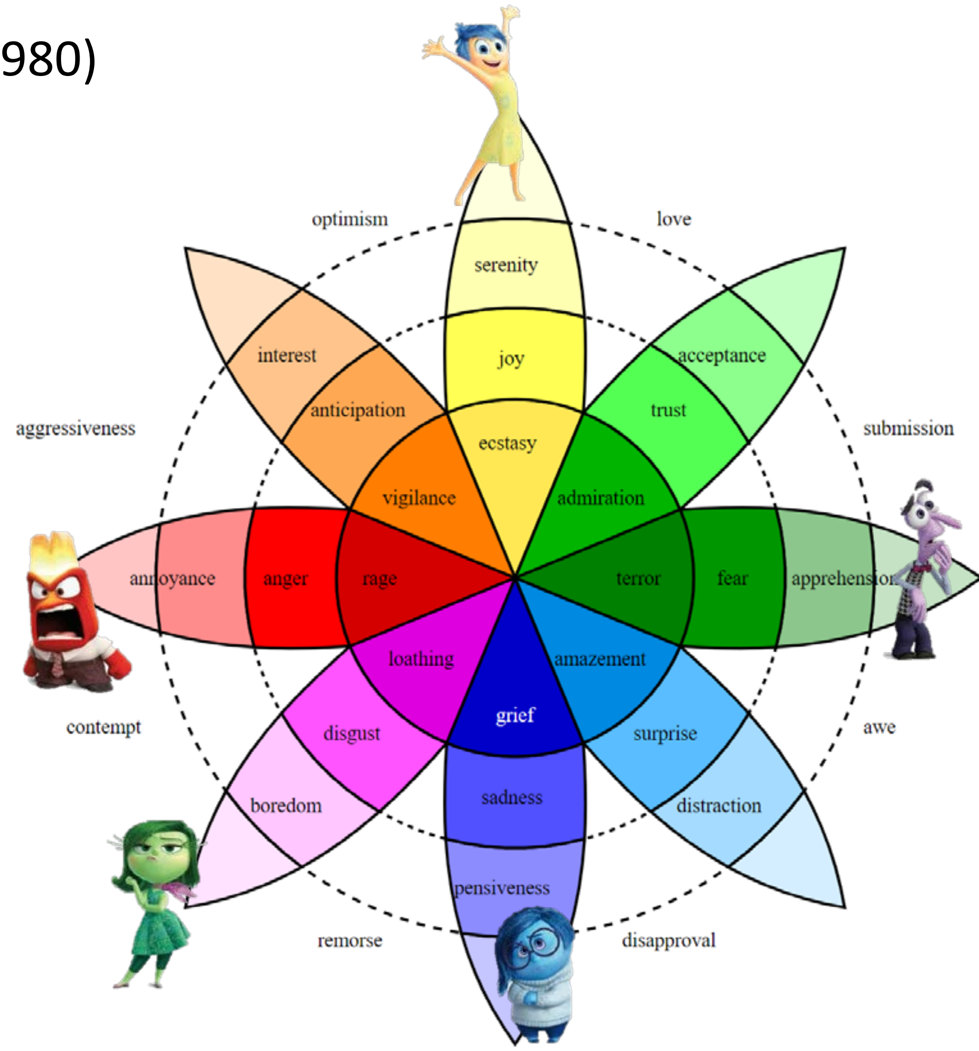


Image source: Wikipedia

Creating dictionaries with quanteda (1/3)

- Quanteda has several functions to
  - **creating** dictionaries (here a tutorial)
    - Dictionaries can be
      - Created from lists of characters by using `dictionary()` or `as.dictionary()`
      - Converted to named lists of characters by using `as.list()`
      - Checked by using `is.dictionary()`
  - **reading** dictionaries
    - Manage several dictionary formats:
      - "wordstat“. Used by the software WordStat (Provalis Research)
      - "LIWC“. Used by Linguistic Inquiry and by software Word Count
      - "yoshikoder“. Used by software Yoshikoder
      - "lexicoder“ v2 and v3. Used by Lexicoder
      - "YAML“. The standard YAML format

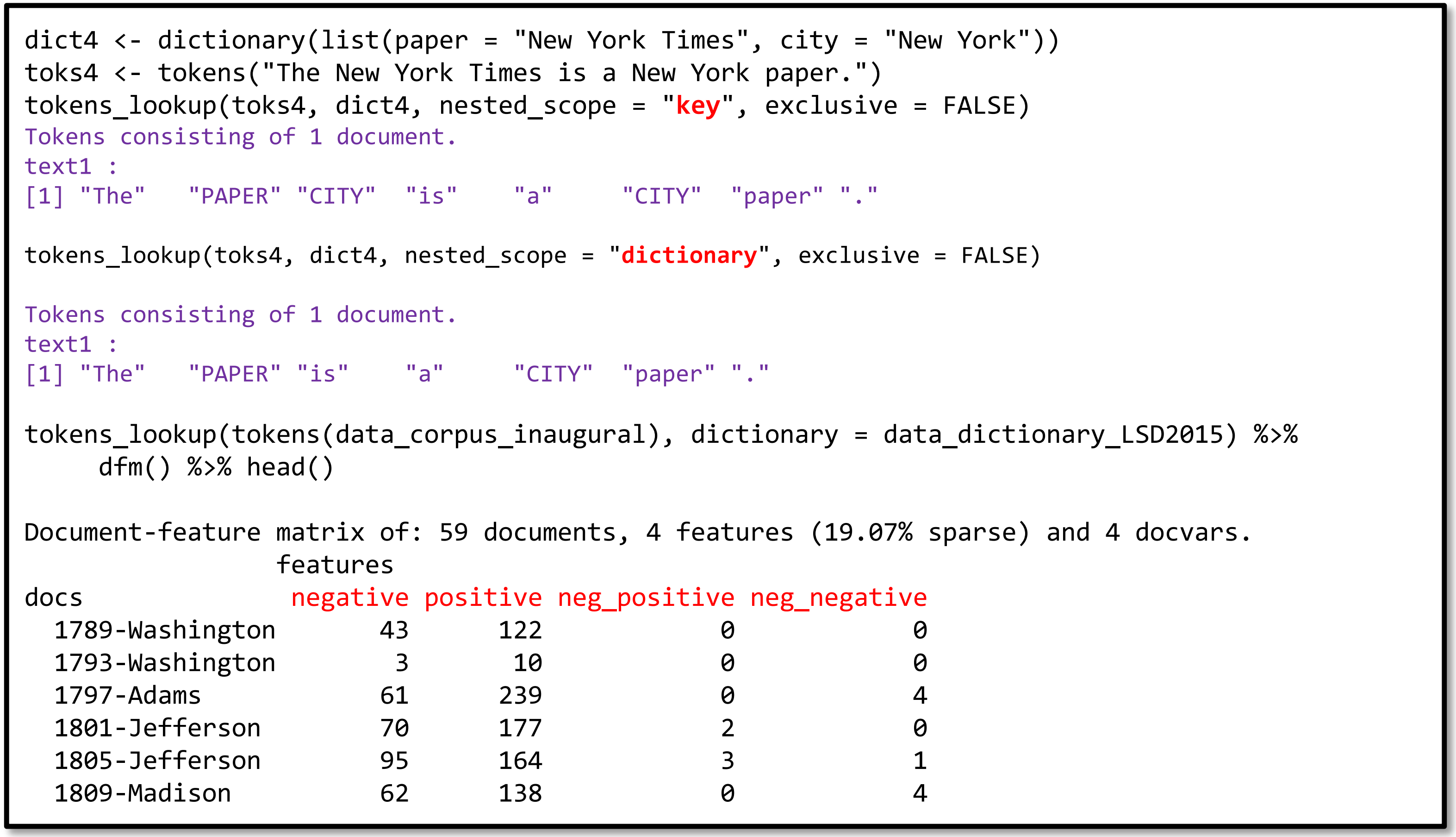
Creating dictionaries with quanteda (2/3)

- In `kwic()`
  - We can show text *windows* centered on words from any category of the dictionary

```
head(kwic(tokens(data_corpus_inaugural),
          pattern=data_dictionary_LSD2015["neg_positive"]
         )
    )

   [1801-Jefferson, 561:562]      long-lost liberty, it was | not wonderful   | that the agitation of the
   [1801-Jefferson, 706:707]      , that this Government is |   not strong    | enough; but would the
   [1805-Jefferson, 772:773]              in any view is it |   not better    | that the opposite bank of
 [1805-Jefferson, 1591:1592]           unaided by power, is | not sufficient  | for the propagation and protection
 [1805-Jefferson, 2055:2056]  human nature that they should |  not approve    | and support them. In
     [1813-Madison, 176:177] successful termination. May we |  not cherish    | this sentiment without presumption

head(kwic(tokens(data_corpus_inaugural),
         pattern=data_dictionary_LSD2015["neg_positive"]
        )
    )
 [1797-Adams, 329:330]  to its recommendations, if | not disobedience | to its authority, not
       [1797-Adams, 428:429]  the people of America were |  not abandoned   | by their usual good sense
       [1797-Adams, 675:676]            and theirs, I did |   not hesitate   | to express my approbation of
     [1797-Adams, 2352:2353]           in early life, and |   not obscured   | but exalted by experience and
 [1805-Jefferson, 2092:2093]        interest; and we need |    not doubt     | that truth, reason,
     [1809-Madison, 336:337] time been distressing us is |  not chargeable  | on any unwarrantable views,
```

Creating dictionaries with quanteda (3/3)

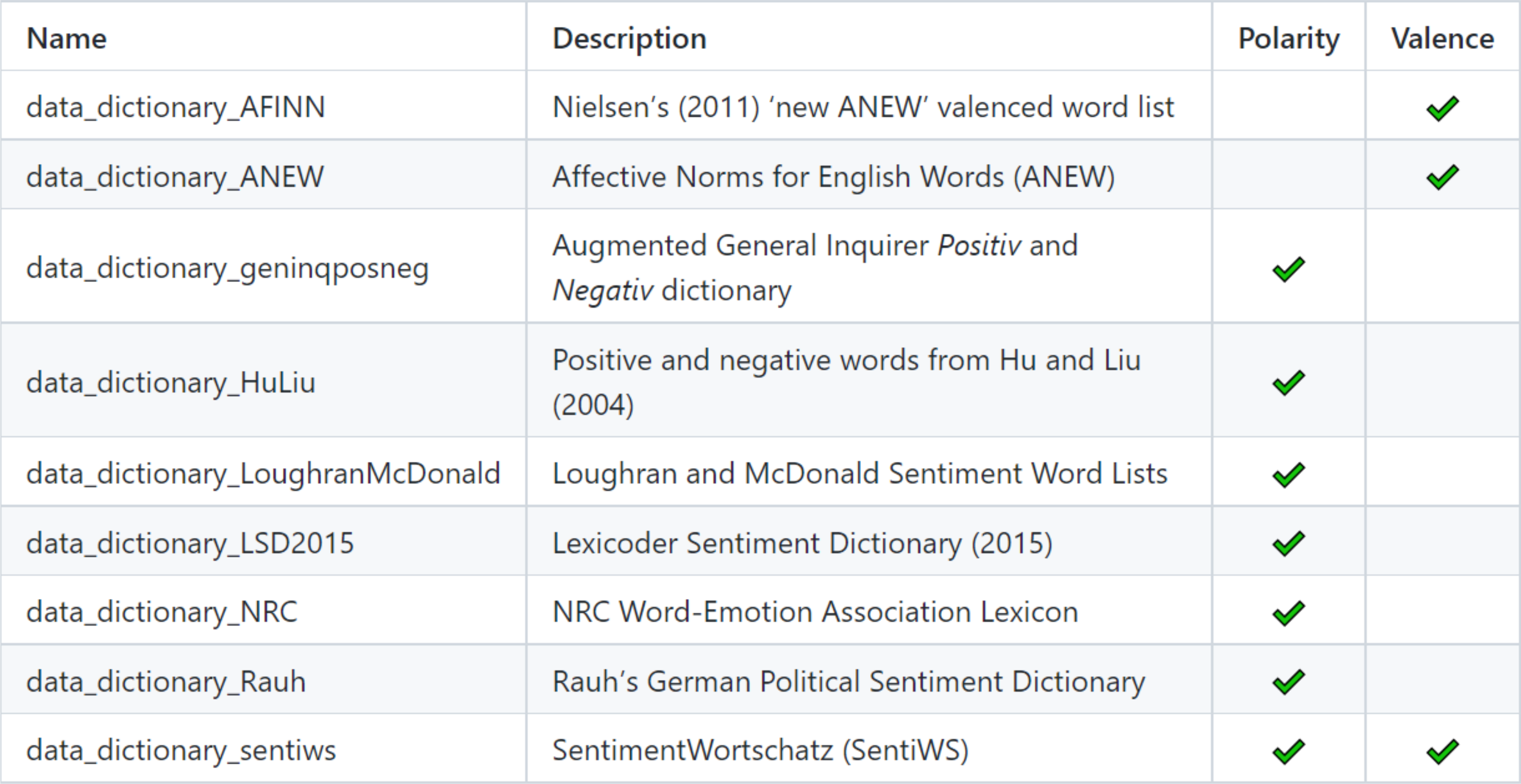
- In `tokens_lookup()`
  - We can replace *tokens* by their category in the dictionary



```
dict4 <- dictionary(list(paper = "New York Times", city = "New York"))
toks4 <- tokens("The New York Times is a New York paper.")
tokens_lookup(toks4, dict4, nested_scope = "key", exclusive = FALSE)
Tokens consisting of 1 document.
text1 :
[1] "The"   "PAPER" "CITY"  "is"    "a"     "CITY"  "paper" "."

tokens_lookup(toks4, dict4, nested_scope = "dictionary", exclusive = FALSE)

Tokens consisting of 1 document.
text1 :
[1] "The"   "PAPER" "is"    "a"     "CITY"  "paper" "."

tokens_lookup(tokens(data_corpus_inaugural), dictionary = data_dictionary_LSD2015) %>%
     dfm() %>% head()

Document-feature matrix of: 59 documents, 4 features (19.07% sparse) and 4 docvars.
                 features
docs              negative positive neg_positive neg_negative
  1789-Washington       43      122            0            0
  1793-Washington        3       10            0            0
  1797-Adams            61      239            0            4
  1801-Jefferson        70      177            2            0
  1805-Jefferson        95      164            3            1
  1809-Madison          62      138            0            4
```

Dictionaries available in quanteda

- Dictionaries included in package `quanteda.sentiment`
  - None in Spanish



Name	Description	Polarity	Valence
data_dictionary_AFINN	Nielsen's (2011) 'new ANEW' valenced word list		✓
data_dictionary_ANEW	Affective Norms for English Words (ANEW)		✓
data_dictionary_geninqposneg	Augmented General Inquirer *Positiv* and *Negativ* dictionary	✓	
data_dictionary_HuLiu	Positive and negative words from Hu and Liu (2004)	✓	
data_dictionary_LoughranMcDonald	Loughran and McDonald Sentiment Word Lists	✓	
data_dictionary_LSD2015	Lexicoder Sentiment Dictionary (2015)	✓	
data_dictionary_NRC	NRC Word-Emotion Association Lexicon	✓	
data_dictionary_Rauh	Rauh's German Political Sentiment Dictionary	✓	
data_dictionary_sentiws	SentimentWortschatz (SentiWS)	✓	✓

Usage examples

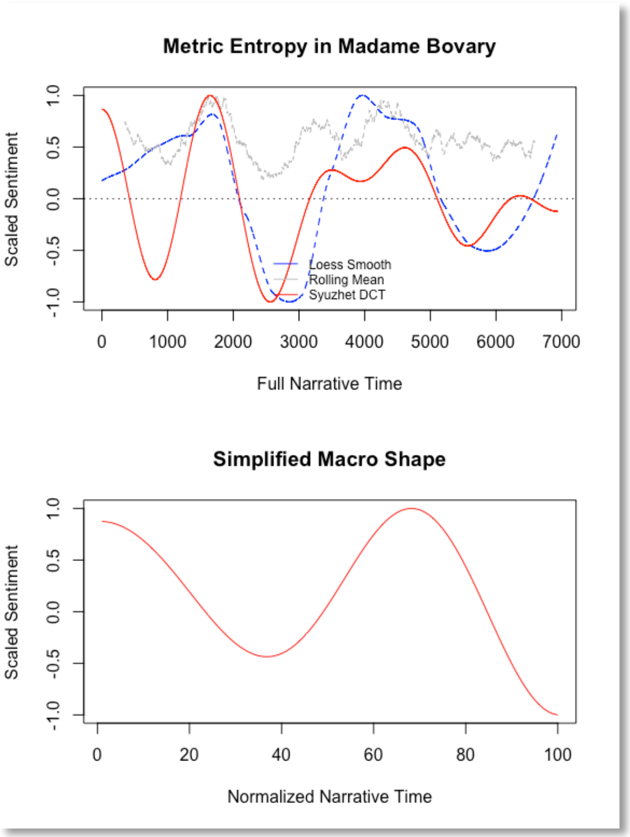
Creating dictionaries

- For **Spanish**:
  - Polarity
    - Cruz, F. L. *et al.* (2014). Building layered, multilingual sentiment lexicons at synset and lemma levels. Expert Systems with Applications, 41(13), 5984-5994.
      - Dataset ML-SentiCon (en XML)
  - Valence
    - Hinojosa, J. A. *et al.* (2016). Affective norms of 875 Spanish words for five discrete emotional categories and two emotional dimensions. Behavior research methods, 48(1), 272-284.
      - Dataset (Excel spreadsheet)
        - » Descriptive statistics for valence, arousal (and concreteness), as well as for each of five discrete emotions (happiness, anger, sadness, fear, disgust).

Download files `sentim_es.rds` and `polar_es.rds`
from `https://tinyurl.com/MRADSNLP`

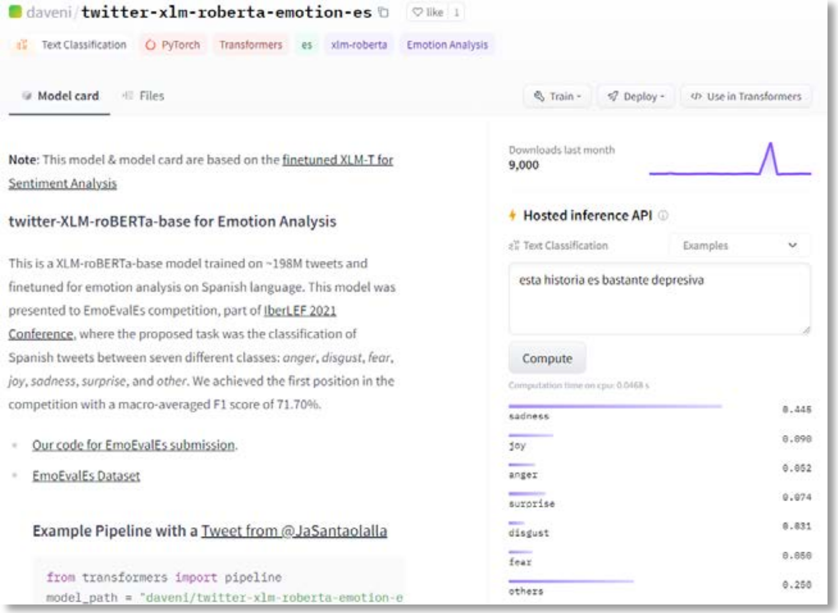
Other dictionaries

- Dictionaries included in the syuzhet package
  - Has 4 sentiment lexicons
    - AFINN (by Nielsen F. A. in the AFINN WORD DATABASE)
    - BING (Minqing H. and Bing L. in the OPINION LEXICON)
    - NRC (Saif M. and Turney P. D. in the NRC EMOTION LEXICON)
      - 8 emotions and 2 sentiments
      - Support for several languages (**Spanish** among them)
  - Computes the "*emotional entropy*" to detect contradictory text sections (that can produce surprise).
  - Dividing of texts and analysis of each piece.



Advanced systems

- They use *machine learning* (and, probably, *deep learning*)
  - The order of words is very important (*bag of words*)
    - “The hotel was very good and not expensive”
    - “The hotel was very expensive and not good”
- Can detect several sentiments (not only binaries)
  - Example: detection of sadness, joy, anger, surprise, disgust, fear and “others”.



Note: This model & model card are based on the finetuned XLM-T for Sentiment Analysis

twitter-XLM-roBERTa-base for Emotion Analysis

This is a XLM-roBERTa-base model trained on ~198M tweets and finetuned for emotion analysis on Spanish language. This model was presented to EmoEvalEs competition, part of IberLEF 2021 Conference, where the proposed task was the classification of Spanish tweets between seven different classes: *anger*, *disgust*, *fear*, *joy*, *sadness*, *surprise*, and *other*. We achieved the first position in the competition with a macro-averaged F1 score of 71.70%.

- Our code for EmoEvalEs submission.
- EmoEvalEs Dataset

Example Pipeline with a Tweet from @JaSantaolalla

```
from transformers import pipeline
model_path = "daveni/twitter-xlm-roberta-emotion-e
```

Downloads last month
9,000

Hosted inference API

Text Classification

Examples

esta historia es bastante depresiva

Compute

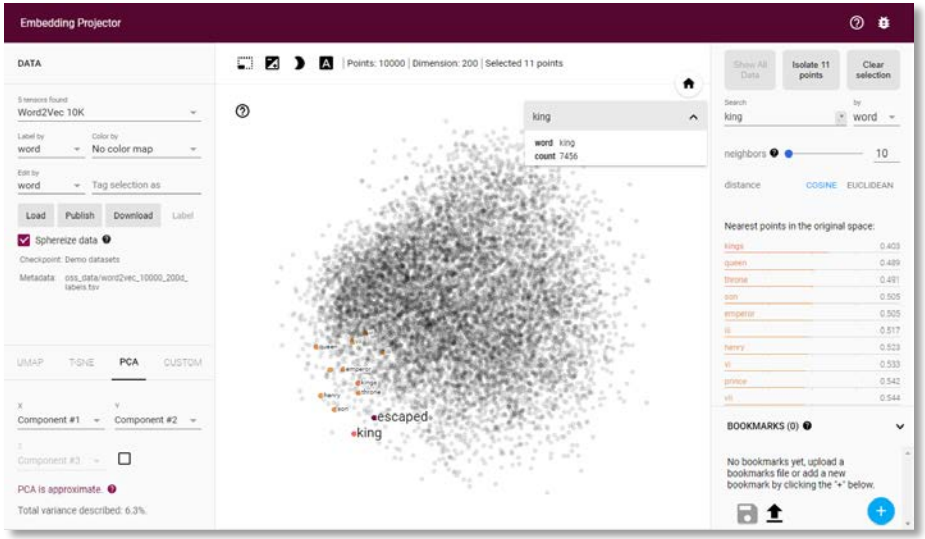
Computation time on cpu: 0.0468 s

```
sadness    0.445
joy        0.090
anger      0.052
surprise   0.074
disgust    0.031
fear       0.050
others     0.250
```

Test here

Embeddings

- We show that converting words into vectors we can get *clusters*
- The TF-IDF matrices also allow you filtering
  - Example of filtering TF-IDF matrices (with `quanteda`)
- Projections from nD to 2D or 3D
  - Tensoflow embedding projector
    - Projections to 2D or 3D
    - Neighbors of a word
      - In the original space
      - In the projected space



Points: 10000 | Dimension: 200 | Selected 11 points

5 tensors found: Word2Vec 10K

Label by: word

Color by: No color map

Edit by: word

Tag selection as

☑ Sphereize data

Checkpoint: Demo datasets

Metadata: oss_data/word2vec_10000_200d_labels.tsv

X: Component #1

Y: Component #2

Z: Component #3 □

PCA is approximate.

Total variance described: 6.3%.

king

word: king

count: 7456

Search: king

by: word

neighbors: 10

distance: COSINE EUCLIDEAN

Nearest points in the original space:

kings	0.403
queen	0.489
throne	0.491
son	0.505
emperor	0.505
iii	0.517
henry	0.523
vi	0.533
prince	0.542
viii	0.544

No bookmarks yet, upload a bookmarks file or add a new bookmark by clicking the "+" below.

Questions?

Course: Intelligent Systems

Unit 4: Language Technologies

Language technologies Part 2

Mariano Rico
2022
Technical University of Madrid