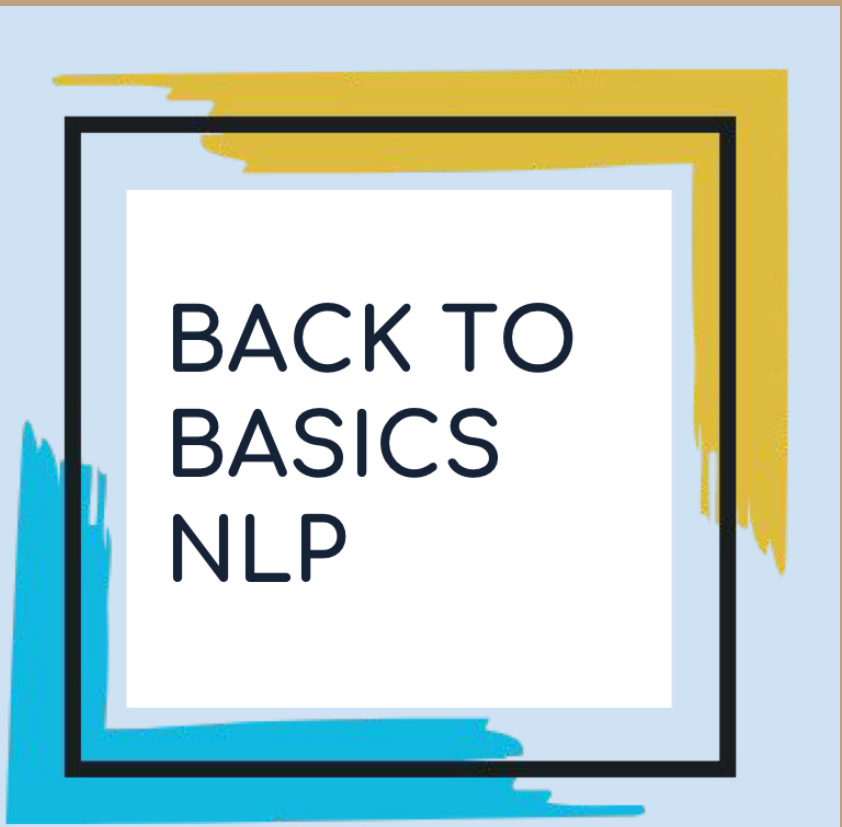


BACK TO BASICS NLP

Octubre 2018

PyConEs Málaga

Claudia Guirao

NATURAL LANGUAGE PROCESSING

notes for PyConEs 2018

@claudiaguirao

I TEXT CLEANING

- regular expressions (regexp): pattern, looking for regular text, useful for → identifying / removing / extracting → text entities
- text/string transformations with 
- what to do with punctuation?
  - conserve → emotions !!
  - remove → standarize
  - what about emojis? 😱
- dealing with the "encoding nightmare" UTF-8 ASCII

II TEXT STANDARIZATION

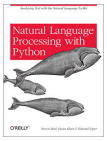
- tokenization: split our text into tokens a.k.a. words
- stemming: take the stems of our tokens
  some stemmers
  - PORTER
  - SNOWBALL
  - LOVINS
  - PAICE

  non of them works so well on Spanish

- lemmatization: grouping together the inflected forms so they can be analysed as a single item

How to perform this task?

custom functions



TextBlob

spaCy

FreeLing

CLiPS
COMPUTATIONAL LINGUISTICS & PSYCHOLINGUISTICS RESEARCH CENTER

- make decisions about other expressions

spetially relevant on informal contexts

LOL
WTF
IMHO

STOPWORDS: most common words in a language.

∄ a universal stop words list → it depends of the domain

TIP: build your own list and iterate

some libraries has its own list of stopwords
- NLTK
- SPACY
- Google STOPWORDS

III MAIN NLP ELEMENTS

notes for PyConES 2018
@claudiaguirao

Corpus: large collection of documents
- monolingual
- multilingual

- general domain (news, books, etc.)
- specific domain → search for corpus on your domain

· some libraries provide corpus on several fields → NLTK

I need a Spanish corpus on my analysis — general / specific

corpus = singular
corpora = plural

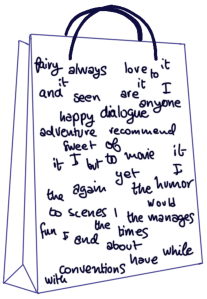
I need a Spanish tagged corpus on my analysis → good luck
↳ you will probably end tagging manually your documents

TASS
Taller de Análisis Semántico en la SEPLN

Documents → each unit in our corpus
- phrase?
- book?
- dialog?

each doc is made up of words = terms

IV BAG OF WORDS




documents

vector space representation

	d1	d2	d3	d4	d5
complex	2		3	2	3
algorithm	3			4	4
entropy	1			2	
traffic		2	3		
network		1	4		

term-document matrix

Variants of term frequency (tf) weight

weighting scheme	tf weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Term frequency: measures how frequently a term occurs in a document

$$TF(t) = \frac{\text{\# times term t appears in a doc}}{\text{\# of terms in the doc}}$$

Inverse document frequency: how important a term is.

TF considers all terms equally important

IDF weigh down the frequent terms while scale up the rare ones

$$IDF = \log\text{-}e\left(\frac{\text{total number of documents}}{\text{\# of documents with term t in it}}\right)$$

TF IDF = term "specificity"

combination of the freq in the document and its freq. on the corpus

$$TF\,IDF(t, d, D) = TF(t, d) \cdot IDF(t, D)$$

how to perform these tasks?
1) custom counter functions
2) sklearn vectorizer. scikit learn

documents → observations
sentiment / classification → target
terms → features

notes for PyConES 2018

@claudiaguirao

DISTANCES = useful for speech correction, search engines and analysis. We can calculate distances through 2 words by several distance measures

LEVENSHTEIN = distance between 2 words

WAGNER & FISHER MINIMUN EDIT DISTANCE = Levenshtein distance
- insertion 1
- deletion 1
- replacement 2

INTENTION
EXECUTION
1) delete i
2) replace n by e
3) substitute t by x
4) insert u
5) replace n by c

V DISTRIBUTIONAL REPRESENTATION

WORD2VEC

gensim
topic modelling for humans

unsupervised algorithm

word2vec

"PS! Thank you for such an awesome top"

→ the meaning of a word is given by its context

→ it is used to predict the nearby word

→ in this case the order is relevant → the context

→ based on NN

→ 2 outputs
- model for the next word
- vectors that can be extracted / added → represents the meaning of a word

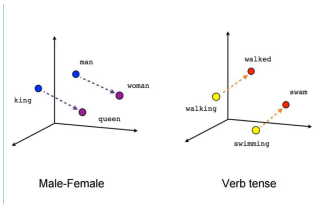
→ 2 ways to train this NN (CBOW // SKIPGRAM)

the cat climbed a tree

Given context:
a, cat, the, tree

Estimate prob. of:
climbed

Word2vec works locally, given a word guesses neighboring words. So it predicts the other words in a sentence given the central pivot (moving window) → WINDOW CONTEXT



VI TOPIC MODELLING

→ extract the main topics of a document by knowing the implicit words / terms relationships

LDA: LATENT DIRICHLET ALLOCATION

unsupervised algorithm

topic modelling algorithm, by Andrew NG & others

DOC_1732

LDA

"PS! Thank you for such an awesome top"

→ works globally, it is a probabilistic model, learns a document vector that predicts words inside that document

→ outputs a previously fixed number of topics and the words related with among their weights.

→

Document 0	10	0	0
Document 1	0	10	0
Document 2	0	0	10
Document 3	10	10	10

Document 0 — Remove Document
Document 1 — Remove Document
Document 2 — Remove Document
Document 3 — Remove Document

LDA2VEC

mix both approaches - local - global

unsupervised algorithm

from Chris Moody paper

VII USE CASES

notes for PyConEs 2018
@claudiaguirao

1) LABELLED DOCUMENTS
- sentiment analysis
- document classification

2) UNLABELLED DOCUMENTS
- segmentation / clustering
- topic modelling
- entities recognition

other analysis & information retrieval
- indexers & search engines
- speech correction
- comparing with general domain
- ontology and network analysis
- translation
- tagging
- intents identification and chatbots