



BACK TO BASICS NLP

Octubre 2018
PyConEs Málaga
Claudia Guirao

Indice



	1	Motivación
	2	First things first
	3	Algoritmos NLP
	4	Casos de uso
	5	Bonus track



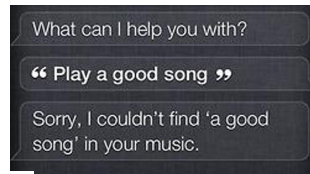
1 | Motivación

Motivación

“El lenguaje es una de las creaciones más grandes y complejas del ser humano”



¿qué podría salir mal si tratamos de simplificarlo a una matriz?



- ↪ Los proyectos de NLP requieren de un proceso iterativo más intenso si cabe que los modelos tradicionales de datos.
- ↪ Existen muchas más oportunidades de que se nos pase algo desapercibido
- ↪ En caso de duda recurran a su lingüista de cabecera



Ciclo de análisis NLP

Tratamiento

*Limpiar y estandarizar el texto
para su posterior análisis*



Word Embeddings

*Representación de términos basada en el
contexto que rodea a cada término,
resulta en la representación vectorial de
los términos pudiendo adicionarse o
substraerse para alcanzar nuevos
términos*

Inputs

*Recolección de fuentes y tratamiento.
En el caso del audio precisa ser
transcrito y eventualmente traducido*



Bag of words

*Representación vectorial de
los documentos, basada en
frecuencia de términos*

Modelado de temas

*Extracción de temas basada en la
relación implícita de las palabras y sus
relaciones (LDA y LDA2VEC)*

2 | First things first

¿De dónde ha venido todo este texto?

Speech Recognition

Viene dado por el proyecto en el que trabajo

Quiero conocer qué dicen las redes sociales

No tengo un corpus, pero voy a conseguirlo

- *Me gustaría trabajar con un corpus en castellano y etiquetado con temática y sentimiento*
- ¡A mi también!



Corpus = colección amplia de documentos monolingües o multilingües

Documento = cada una de las unidades que componen el corpus (frases, diálogos, libros enteros, etc.)

Trabajar con texto puede ser complicado

Limpiar el texto

- Transformación del texto de entrada
- ¿qué hacemos con la puntuación?
 - ✓
 - ✗
 - Emojis 🤖
- Cómo resolvemos complicaciones con el encoding -> [UNICODE](#)



PRO TIP: Invertir tiempo en mejorar nuestras habilidades con expresiones regulares y bash nos hará ganar velocidad y precisión



Librerías en Python

NLTK



TextBlob

spaCy

gensim

Trabajar con texto puede ser complicado

Estandarizar el texto

- Tokenización
- Estemización
- Lematización
- *LOL, TQM, WTF*
- Eliminar STOPWORDS

```
print(info_nominadas.iloc[1].descripcion)
print(tokenize_only_spacy(info_nominadas.iloc[1].descripcion))
print(tokenize_and_lemm_spacy(info_nominadas.iloc[1].descripcion))
```

Frida, una niña de seis años, afronta el primer verano de su vida con su nueva familia adoptiva, tras la muerte de su madre.

['niña', 'años', 'afronta', 'verano', 'vida', 'familia', 'adoptiva', 'muerte', 'madre']

['niño', 'año', 'afrontar', 'verano', 'vida', 'familia', 'adoptivo', 'muerte', 'madre']

- En inglés todo va bien,
pero en castellano los
estemizadores /
lematizadores no me
funcionan
- ¡A mi tampoco!

Aquí te dejo algunos recursos que quizá
te ayuden:

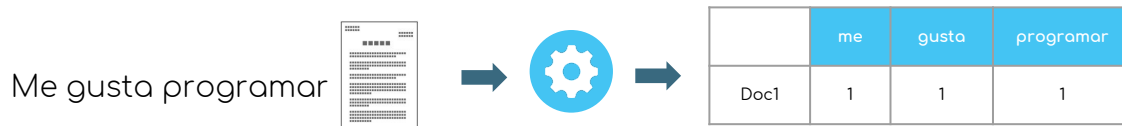
CLiPS
Computational Linguistics & Psycholinguistics
University of Antwerp

FreeLing

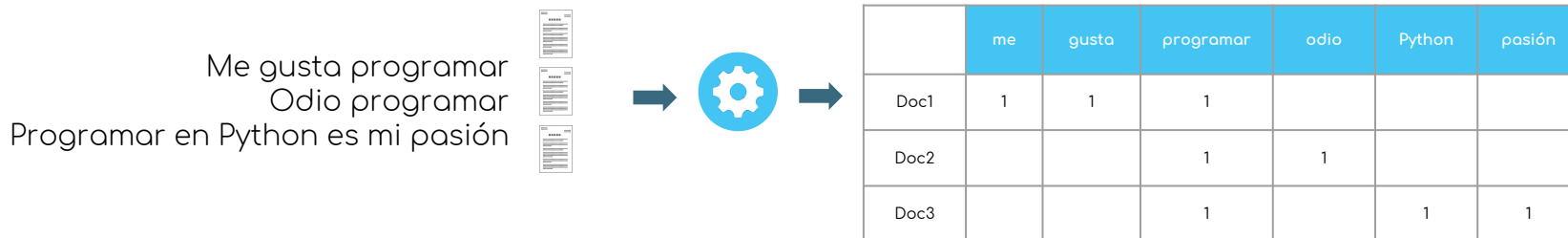
3 | Algoritmos NLP

Bag of words

- Es uno de los posibles enfoques cuando trabajamos con documentos. Mediante él podemos representar nuestros documentos ignorando el orden de las palabras
- Así tenemos tantas “bolsas” como documentos, que contienen un subconjunto de palabras de nuestro diccionario (todas las palabras del corpus)
- Es el enfoque más sencillo y computacionalmente menos costoso



Term Document Matrix



Bag of words

Term Frequency

Mide la frecuencia en la que los términos aparecen en un documento

Inverse document frequency

Mientras que TF considera todos los términos igual de importantes, IDF pesa la frecuencia de los términos potenciando los más extraños

TF-IDF

Es un indicador de la especificidad de un término, combina la frecuencia en el documento con la frecuencia en el corpus

Variants of term frequency (tf) weight

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

$$\text{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

Bag of words



¿Cómo realizar estas operaciones?

- 1) Desarrollando nuestras propias funciones
- 2) SkLearn vectorizer

```
from sklearn.feature_extraction.text import TfidfVectorizer

#define vectorizer parameters
tfidf_vectorizer = TfidfVectorizer(max_df=0.8, max_features=200000,
                                   min_df=3, stop_words=MY_STOP_WORDS,
                                   use_idf=True, tokenizer=tokenize_and_lemm_spacy, ngram_range=(1,3))

%time tfidf_matrix = tfidf_vectorizer.fit_transform(synopses) #fit the vectorizer to synopses

print(tfidf_matrix.shape)
```

```
CPU times: user 41 s, sys: 600 ms, total: 41.6 s
Wall time: 20.9 s
(921, 2432)
```

Es la máxima frecuencia dentro de los documentos que un feature dado puede ser utilizado en la matriz tf-idf. Si fijamos aquí un umbral mayor al 80% probablemente contenga poco significado, al menos en el contexto de este análisis.

Puede tomar un porcentaje o un entero. Por ejemplo 3, en ese caso debe aparecer al menos 3 documentos para ser considerado.

Función que se va a emplear para tokenizar, lematizar

Corpus = training set
Documento = observación
Terms = variables o features
Sentimiento/tag/tema = target

Word Embeddings

Word2vec

Permite mapear las palabras en vectores, así el significado de cada palabra se asume implícito en las palabras que le rodean.

Word2vec es uno de los algoritmos más conocidos para transformar las palabras en este tipo de representación

Basado en **redes neuronales**, existen 2 alternativas: Skip-Gram y CBOW

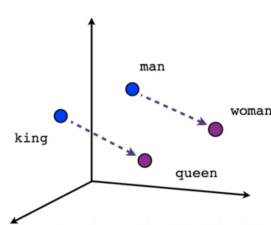
Produce 2 outputs:

- Modelo que permite predecir la próxima palabra
- Vectores representativos de los términos, creando un espacio vectorial semántico basado en la idea que el significado de una palabra puede ser aprendido de un entorno lingüístico

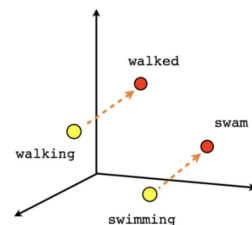
word2vec

“PS! Thank you for such an
awesome top”

Fuente ["Introducing our Hybrid Ido2vec Algorithm"](#)



Male-Female

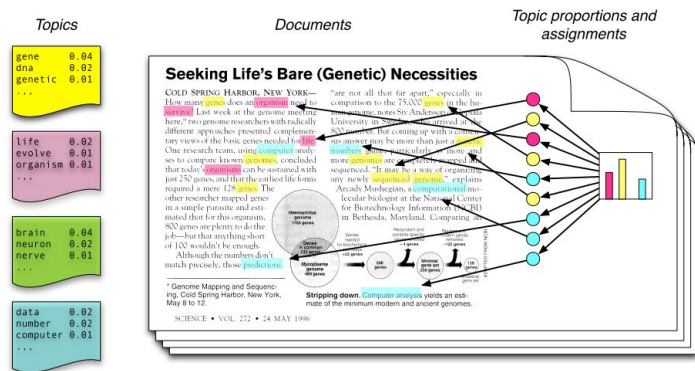


Verb tense

Topic Modelling

Latent Dirichlet Allocation (LDA)

Es un modelo generativo que permite que conjuntos de observaciones puedan ser explicados por grupos no observados que explican por qué algunas partes de los datos son similares. Por ejemplo, si las observaciones son palabras en documentos, presupone que cada documento es una mezcla de un pequeño número de categorías (también denominados como temas) y la aparición de cada palabra en un documento se debe a una de las categorías a las que el documento pertenece.



DOC_1732

LDA

“PS! Thank you for such an awesome top”

Fuente ["Introducing our Hybrid lda2vec Algorithm"](#)

Topic Modelling

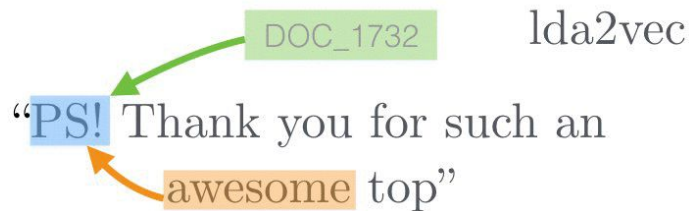
LDA2VEC

Se inspira en LDA, en él word2vec se expande para aprender simultáneamente de las palabras, los documentos y los vectores de términos.

Se obtiene modificando la estructura de skip-gram, en él se pivota sobre el vector de una palabra y el de un documento, añadiendo así un vector de contexto, usado para predecir las palabras que le rodean. Estos vectores se usan para detectar similitudes.

Estos vectores se entrenan en el contexto de las palabras, lo cual tiene 2 ventajas:

1. Permite comprender bien los términos que configuran un tema
2. Devuelve vectores de un contexto especializado



Fuente ["Introducing our Hybrid lda2vec Algorithm"](#)

Así, si el vector de un documento es una combinación de *comida y bebida*, los términos más frecuentes serán “pan”, “queso” y “vino”. Si el vector representa *geografía y lugares*, lo más probable es que aparezcan ciudades y países.

4 | Casos de uso

Casos de uso

Documentos etiquetados

- Análisis de sentimiento
- Clasificación documental

Documentos no etiquetados

- Segmentación y clustering
- Topic modelling
- Reconocimiento de entidades

Information retrieval

- Indexadores y motores de búsqueda
- Corrección ortográfica
- Comparativa con el dominio general
- Ontología y análisis de grafos
- Traducción
- Etiquetado
- Chatbots
- etc.



5

Bonus track

Bonus track

Apuntes sobre la presentación

- NATURAL LANGUAGE PROCESSING -

notes for PyConES 2018
@clavdia.guizao




①

TEXT CLEANING

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

②

TEXT STANDARDIZATION

- tokenization: split our text into tokens a.k.a words
- stemming: take the stems of our tokens
 - some stemmers
 - Porter
 - Snowball
 - Lownis
 - Porter
 - How to perform this task?
 - custom functions
 -   
 - spaCy
 - FreeLing
 - CLIPS
- lemmatization: grouping together the inflected forms so they can be analysed as a single item
- make decisions about other expressions
 - LOL
 - WTF
 - IMHO

Notebook NLP sobre las sinopsis de los premios Goya



<https://github.com/intiveda/back2basicsNLP>

Contacto y preguntas

Claudia Guirao Fernández
@claudiaguirao

