

Capítulo 2: Machine Learning en Python

Capítulo 3: NLP en Python

Capítulo 4: Django Web Framework

NLP en Python

Data Science for Developer



Objetivos

- Explicar el concepto de Natural Language Processing (NLP).
- Identificar las técnicas de NLP.
- Aplicar una metodología de Data Science.
- Aplicar NLP usando NLTK y Scikit-learn.

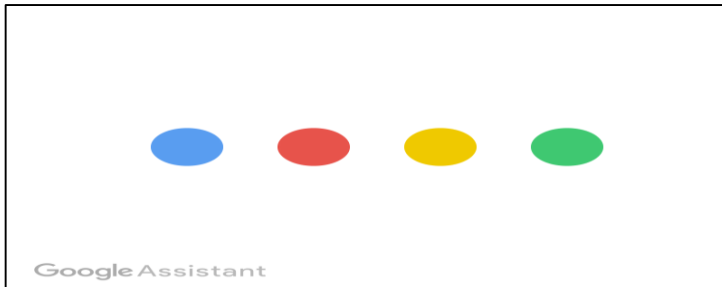


Agenda

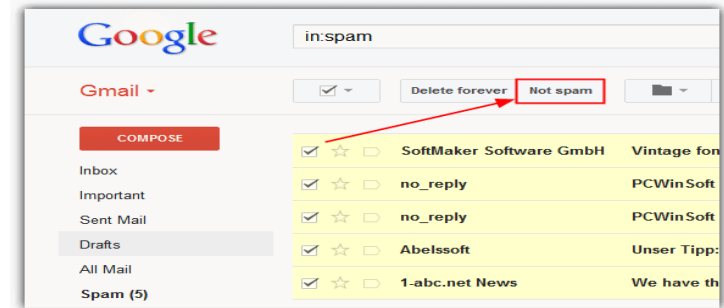
- Algunos usos de NLP
- ¿Qué es NLP?
- Técnicas de NLP
- Frameworks NLP
- Natural Language Toolkit (NLTK)
- Expresiones Regulares en Python
- Scikit-learn
- TF-IDF en Scikit-learn
- Similitud de documentos usando Distancia Coseno



Algunos usos de NLP



Asistente Digital



Filtro Spam



Traductor

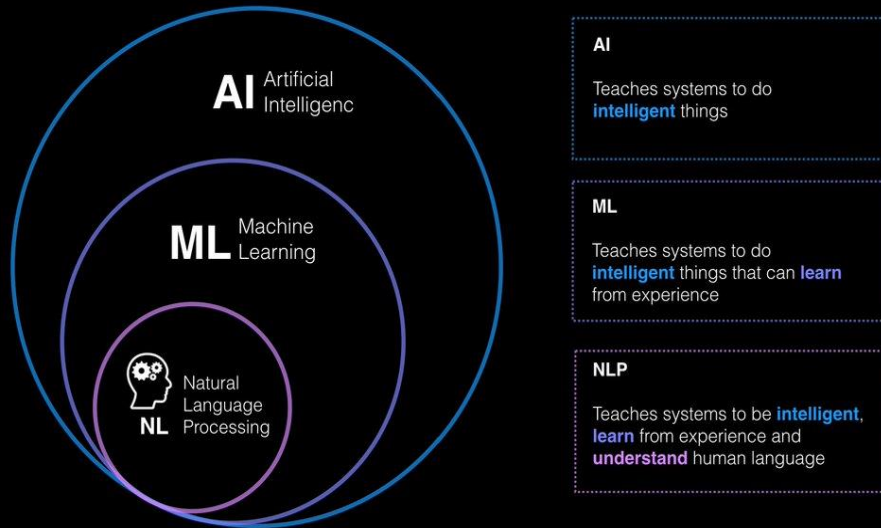


Similitud entre Documentos



¿Qué es NLP?

How AI, ML and NLP work together



NLP = Natural Language Processing

VS

NLP = Neuro linguistic programming



Técnicas de NLP

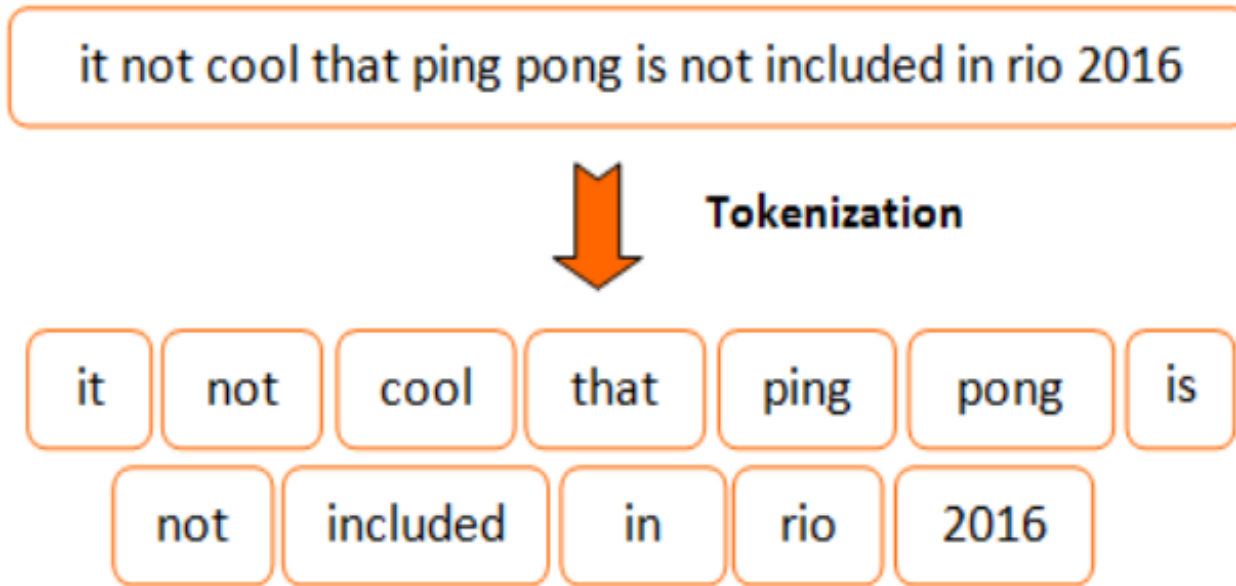
Tokenization	Stop words	Stemming
POS Tagging	Chunking	Named Entity Recognition
Lemmatization	Corpus	Word Net

Fuente:

<https://towardsdatascience.com/introducing-natural-language-processing-nlp-series-2-covering-essentials-of-nlp-phase-1-3a8d2efd8699>



Tokenization



<https://hackernoon.com/nlp-101-topic-modeling-for-humans-part-1-a030e8155584>



Stopwords



<https://towardsdatascience.com/introducing-natural-language-processing-nlp-series-2-covering-essentials-of-nlp-phase-1-3a8d2efd8699>



Stemming

Sample text: Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Lovins stemmer: such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Porter stemmer: such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Paice stemmer: such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

**Se identifican
elementos
esenciales de las
palabras (raíces).**

<https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html>



Existen muchos Frameworks de NLP en Python



Pattern



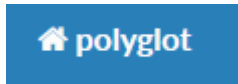
spaCy



<https://www.kdnuggets.com/2018/07/comparison-top-6-python-nlp-libraries.html>



Intro



- Soporte multi idioma.
- No tan popular como otros frameworks.
- Comunidad reducida.
- <https://pypi.org/project/polyglot/>



Funciones(1)



Detección de Idioma

```
text = Text("Bonjour, Mesdames.")
print("Language Detected: Code={}, Name={}\n".format(text.language.code, text.language.name))
```

```
Language Detected: Code=fr, Name=French
```

Tokenizar

```
zen = Text("Beautiful is better than ugly. "
           "Explicit is better than implicit. "
           "Simple is better than complex.")
print(zen.words)
```

```
[u'Beautiful', u'is', u'better', u'than', u'ugly', u'.', u'Explicit',
```



Funciones(2)



Detección de Idioma

```
text = Text(u"O primeiro uso de desobediência civil em massa ocorreu em setembro de 1906.")  
  
print("{:<16}{}".format("Word", "POS Tag")+"\n"+"-"*30)  
for word, tag in text.pos_tags:  
    print(u"{:<16}{:>2}".format(word, tag))
```

Tokenizar

Word	POS Tag

O	DET
primeiro	ADJ
uso	NOUN
de	ADP
desobediência	NOUN
civil	ADJ
em	ADP
massa	NOUN
ocorreu	ADJ
em	ADP
setembro	NOUN
de	ADP
1906	NUM
.	PUNCT



Funciones(3)



Detección de Idioma

```
word = Text("Preprocessing is an essential step.").words[0]  
print(word.morphemes)
```

```
[u'Pre', u'process', u'ing']
```

Tokenizar

```
from polyglot.transliteration import Transliterator  
transliterator = Transliterator(source_lang="en", target_lang="ru")  
print(transliterator.transliterate(u"preprocessing"))
```

```
препрокеcсинг
```



Intro

Pattern

- Web miner.
- Existe soporte NLP aunque es un extra.
- No hay soporte para Python 3.
- Soporte limitado de idiomas.
- <https://www.clips.uantwerpen.be/pattern>



Funciones(1)

Pattern

POS tagger

```
>>> from pattern.en import parse
>>>
>>> s = 'The mobile web is more important than mobile apps.'
>>> s = parse(s, relations=True, lemmata=True)
>>> print s

'The/DT/B-NP/O/NP-SBJ-1/the mobile/JJ/I-NP/O/NP-SBJ-1/mobile' ...
```

Búsqueda de patrones

```
>>> from pattern.en import parsetree
>>> from pattern.search import search
>>>
>>> s = 'The mobile web is more important than mobile apps.'
>>> s = parsetree(s, relations=True, lemmata=True)
>>>
>>> for match in search('NP be RB?+ important than NP', s):
>>>     print match.constituents()[-1], '=>', \
>>>         match.constituents()[0]

Chunk('mobile apps/NP') => Chunk('The mobile web/NP-SBJ-1')
```



Funciones(2)

Pattern

POS tagger

Búsqueda de patrones

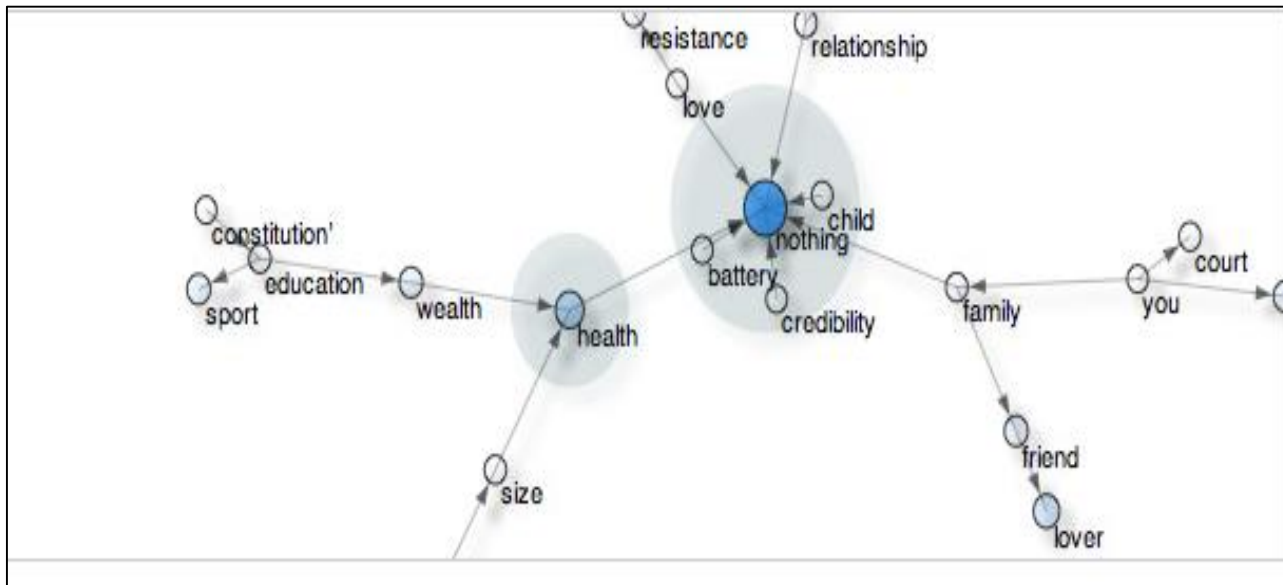
```
>>> from pattern.web import Twitter
>>> from pattern.en import tag
>>> from pattern.vector import KNN, count
>>>
>>> twitter, knn = Twitter(), KNN()
>>>
>>> for i in range(1, 10):
>>>     for tweet in twitter.search('#win OR #fail', start=i, count=100):
>>>         s = tweet.text.lower()
>>>         p = '#win' in s and 'WIN' or 'FAIL'
>>>         v = tag(s)
>>>         v = [word for word, pos in v if pos == 'JJ'] # JJ = adjective
>>>         v = count(v)
>>>         if v:
>>>             knn.train(v, type=p)
>>>
>>> print knn.classify('sweet potato burger')
>>> print knn.classify('stupid autocorrect')

'WIN'
'FAIL'
```



Funciones(3)

Pattern



Intro



- Soporta grandes cantidades de datos.
- Soporta Deep learning.
- No soporta el stack completo de NLP.
- <https://radimrehurek.com/gensim/>



Tokenización

```
>>> from pprint import pprint # pretty-printer
>>> from collections import defaultdict
>>>
>>> # remove common words and tokenize
>>> stoplist = set('for a of the and to in'.split())
>>> texts = [[word for word in document.lower().split() if word not in stoplist]
>>>            for document in documents]
>>>
>>> # remove words that appear only once
>>> frequency = defaultdict(int)
>>> for text in texts:
>>>     for token in text:
>>>         frequency[token] += 1
>>>
>>> texts = [[token for token in text if frequency[token] > 1]
>>>            for text in texts]
>>>
>>> pprint(texts)
[['human', 'interface', 'computer'],
 ['survey', 'user', 'computer', 'system', 'response', 'time'],
 ['eps', 'user', 'interface', 'system'],
 ['system', 'human', 'system', 'eps'],
 ['user', 'response', 'time'],
 ['trees'],
 ['graph', 'trees'],
 ['graph', 'minors', 'trees'],
 ['graph', 'minors', 'survey']]
```



Funciones(2)



Tokenización

```
>>> sims = sorted(enumerate(sims), key=lambda item: -item[1])
>>> print(sims) # print sorted (document number, similarity score) 2-tuples
[(2, 0.99844527), # The EPS user interface management system
(0, 0.99809301), # Human machine interface for lab abc computer applications
(3, 0.9865886), # System and human system engineering testing of EPS
(1, 0.93748635), # A survey of user opinion of computer system response time
(4, 0.90755945), # Relation of user perceived response time to error measurement
(8, 0.050041795), # Graph minors A survey
(7, -0.098794639), # Graph minors IV Widths of trees and well quasi ordering
(6, -0.1063926), # The intersection graph of paths in trees
(5, -0.12416792)] # The generation of random binary unordered trees
```



Funciones(3)



Tokenización

```
>>> # extract 400 LSI topics; use the default one-pass algorithm
>>> lsi = gensim.models.lsimodel.LsiModel(corpus=mm, id2word=id2word, num_topics=400)
>>>
>>> # print the most contributing words (both positively and negatively) for each of the first ten topics
>>> lsi.print_topics(10)
topic #0(332.762): 0.425*"utc" + 0.299*"talk" + 0.293*"page" + 0.226*"article" + 0.224*"delete" + 0.216*"dis
topic #1(201.852): 0.282*"link" + 0.209*"he" + 0.145*"com" + 0.139*"his" + -0.137*"page" + -0.118*"delete" +
topic #2(191.991): -0.565*"link" + -0.241*"com" + -0.238*"blacklist" + -0.202*"diff" + -0.193*"additions" +
topic #3(141.284): -0.476*"image" + -0.255*"copyright" + -0.245*"fair" + -0.225*"use" + -0.173*"album" + -0.
topic #4(130.909): 0.264*"population" + 0.246*"age" + 0.243*"median" + 0.213*"income" + 0.195*"census" + -0.
topic #5(120.397): 0.304*"diff" + 0.278*"utc" + 0.213*"you" + -0.171*"additions" + 0.165*"talk" + -0.159*"in
topic #6(115.414): -0.362*"diff" + -0.203*"www" + 0.197*"you" + -0.180*"undo" + -0.180*"kategorij" + 0.164*"u
topic #7(111.440): 0.429*"kategorij" + 0.276*"categorija" + 0.251*"category" + 0.207*"kategorija" + 0.198*"kat
topic #8(109.907): 0.385*"album" + 0.224*"song" + 0.209*"chart" + 0.204*"band" + 0.169*"released" + 0.151*"r
topic #9(102.599): -0.237*"league" + -0.214*"he" + -0.180*"season" + -0.174*"football" + -0.166*"team" + 0.1
```



spaCy Intro

- El framework más rápido.
- Muy fácil de usar.
- Soporte reducido para idiomas como castellano.
- Buena documentación.
- <https://spacy.io/>



Funciones (1)



POS Tagging

```
import spacy

nlp = spacy.load('en_core_web_sm')
doc = nlp(u'Apple is looking at buying U.K. startup for $1 billion')

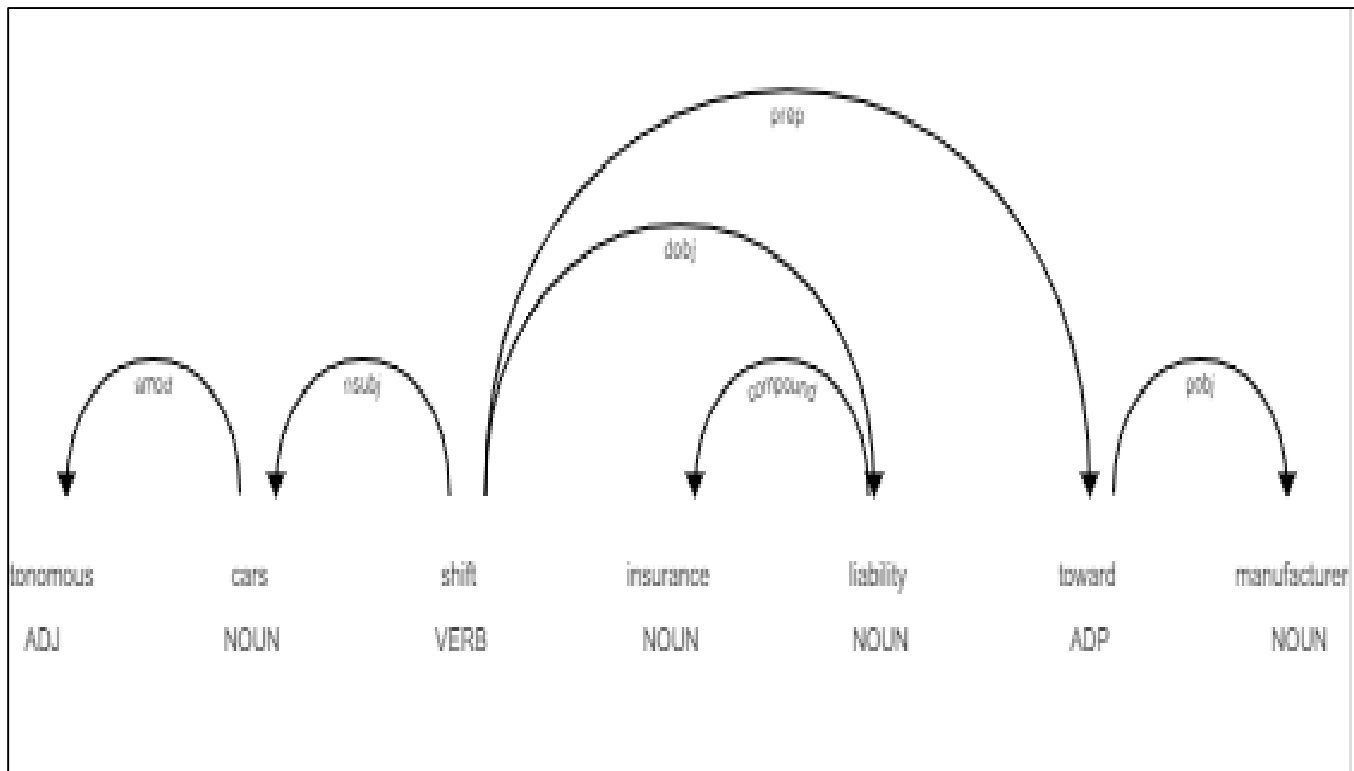
for token in doc:
    print(token.text, token.lemma_, token.pos_, token.tag_, token.dep_,
          token.shape_, token.is_alpha, token.is_stop)
```



Funciones (2)

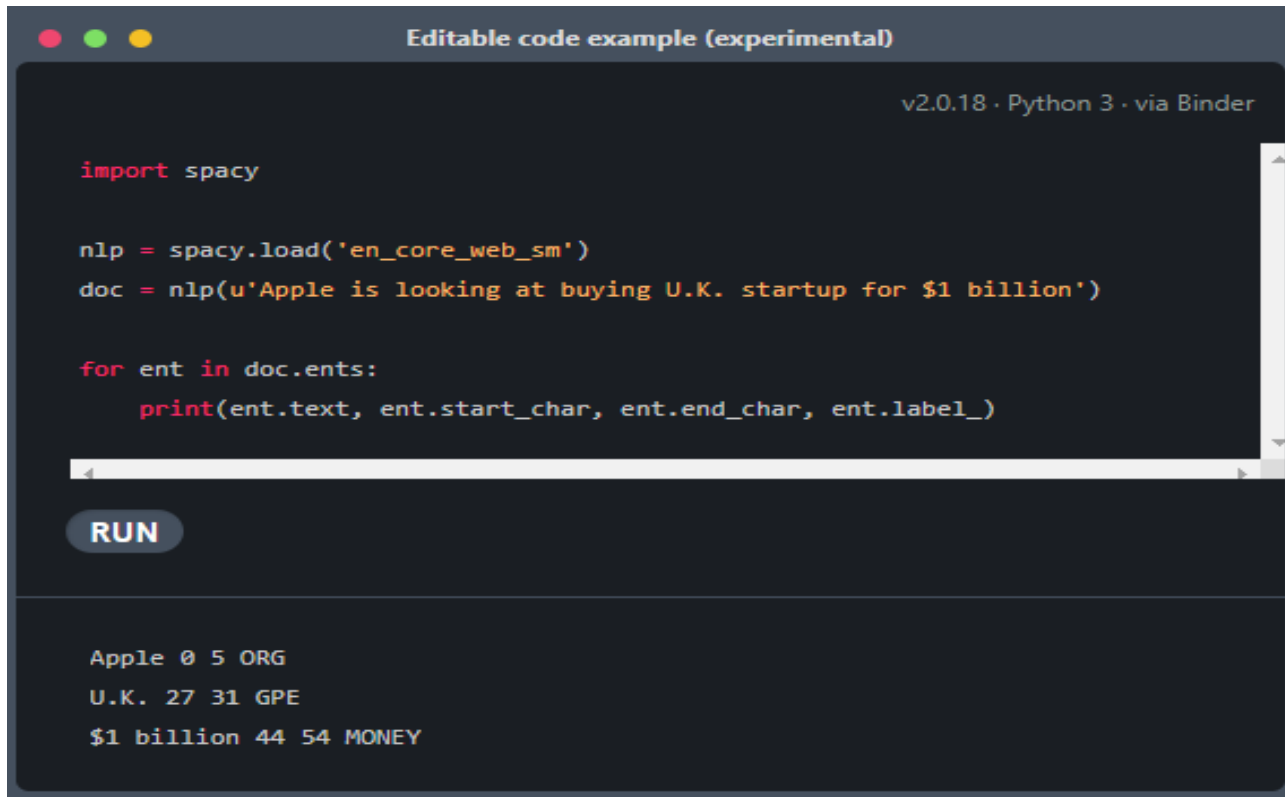
spaCy

POS Tagging



spaCy Funciones (3)

POS Tagging



The screenshot shows a Jupyter Notebook window titled "Editable code example (experimental)". The code in the notebook is as follows:

```
import spacy

nlp = spacy.load('en_core_web_sm')
doc = nlp(u'Apple is looking at buying U.K. startup for $1 billion')

for ent in doc.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
```

Below the code editor is a "RUN" button. The output of the code is displayed in the cell below the button:

```
Apple 0 5 ORG
U.K. 27 31 GPE
$1 billion 44 54 MONEY
```



Funciones (4)



POS Tagging

```
import spacy
from spacy.matcher import Matcher

nlp = spacy.load('en_core_web_sm')
matcher = Matcher(nlp.vocab)
# add match ID "HelloWorld" with no callback and one pattern
pattern = [{ 'LOWER': 'hello' }, { 'IS_PUNCT': True }, { 'LOWER': 'world' }]
matcher.add('HelloWorld', None, pattern)

doc = nlp(u'Hello, world! Hello world!')
matches = matcher(doc)
for match_id, start, end in matches:
    string_id = nlp.vocab.strings[match_id] # get string representation
    span = doc[start:end] # the matched span
    print(match_id, string_id, start, end, span.text)
```

RUN

```
15578876784678163569 HelloWorld 0 3 Hello, world
```



Ejercicio N° 3.1: spaCy

Al finalizar el laboratorio, el alumno logrará:

- Demostrar competencias básicas en uso de un framework NLP.



Natural Language Analysis Toolkit (NLTK)

- Desarrollado el año 2001 por la University of Pennsylvania.
- Académico, bueno para aprender NLP.
- Diseñado para Python con los siguientes objetivos:
 - Simple
 - Consistente
 - Extendible
 - Modular
- <https://www.nltk.org/>



Natural Language Analysis
with Python NLTK



NLTK - Algunos Usos

Tokenizer

```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens

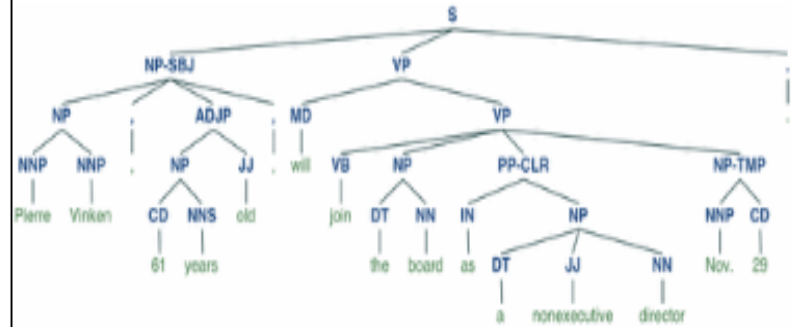
['At', 'eight', 'o'clock', 'on', 'Thursday', 'morning',
'Arthur', 'did', 'n't', 'feel', 'very', 'good', '.']
```

Entidades

```
>>> entities = nltk.chunk.ne_chunk(tagged)
>>> entities
Tree('S', [(('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'),
              ('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN')),
            Tree('PERSON', [(('Arthur', 'NNP'))],
                  (('did', 'VBD'), ("n't", 'RB'), ('feel', 'VB'),
                   ('very', 'RB'), ('good', 'JJ'), ('.', '.'))])
```

Arbol de estructuras

```
>>> from nltk.corpus import treebank
>>> t = treebank.parsed_sents('wsj_0001.mrg')[0]
>>> t.draw()
```



NLTK - Stopwords

```
# Excluir los llamados stopwords
import nltk
from nltk.corpus import stopwords
print (stopwords.words("spanish"))
```

```
['de', 'la', 'que', 'el', 'en', 'y', 'a', 'los', 'del', 'se', 'las', 'por', 'un', 'para', 'con', 'no', 'una', 'su', 'al', 'lo',
'como', 'más', 'pero', 'sus', 'le', 'ya', 'o', 'este', 'sí', 'porque', 'esta', 'entre', 'cuando', 'muy', 'sin', 'sobre', 'tambi
én', 'me', 'hasta', 'hay', 'donde', 'quien', 'desde', 'todo', 'nos', 'durante', 'todos', 'uno', 'les', 'ni', 'contra', 'otros',
'ese', 'eso', 'ante', 'ellos', 'e', 'esto', 'mí', 'antes', 'algunos', 'qué', 'unos', 'yo', 'otro', 'otras', 'otra', 'él', 'tant
o', 'esa', 'estos', 'mucho', 'quienes', 'nada', 'muchos', 'cual', 'poco', 'ella', 'estar', 'estas', 'algunas', 'algo', 'nosotro
s', 'mi', 'mis', 'tú', 'te', 'ti', 'tu', 'tus', 'ellas', 'nosotras', 'vosotros', 'vosotras', 'os', 'mío', 'mía', 'míos', 'mía
s', 'tuyo', 'tuya', 'tuyos', 'tuyas', 'suyo', 'suya', 'suyos', 'suyas', 'nuestro', 'nuestra', 'nuestros', 'nuestras', 'vuestro
o', 'vuestra', 'vuestros', 'vuestras', 'esos', 'esas', 'estoy', 'estás', 'está', 'estamos', 'estáis', 'están', 'esté', 'estés',
'estemos', 'estéis', 'estén', 'estaré', 'estarás', 'estará', 'estaremos', 'estaréis', 'estarán', 'estaría', 'estarías', 'estaría
mos', 'estaríais', 'estarían', 'estaba', 'estabas', 'estábamos', 'estabais', 'estaban', 'estuve', 'estuviste', 'estuvo', 'estu
vimos', 'estuvisteis', 'estuvieron', 'estuviera', 'estuvieras', 'estuviéramos', 'estuvierais', 'estuvieran', 'estuviese', 'estu
vieses', 'estuviésemos', 'estuvieseis', 'estuviesen', 'estando', 'estado', 'estada', 'estados', 'estadas', 'estad', 'he', 'ha
s', 'ha', 'hemos', 'habéis', 'han', 'haya', 'hayas', 'hayamos', 'hayáis', 'hayan', 'habré', 'habrás', 'habrá', 'habremos', 'hab
réis', 'habrán', 'habría', 'habrías', 'habríamos', 'habríais', 'habrían', 'había', 'habías', 'habíamos', 'habíais', 'habían',
'hube', 'hubiste', 'hubo', 'hubimos', 'hubisteis', 'hubieron', 'hubiera', 'hubieras', 'hubiéramos', 'hubierais', 'hubieran', 'h
ubiese', 'hubieses', 'hubiésemos', 'hubieseis', 'hubiesen', 'habiendo', 'habido', 'habida', 'habidos', 'habidas', 'soy', 'ere
s', 'es', 'somos', 'sois', 'sea', 'seas', 'seamos', 'seáis', 'sean', 'seré', 'serás', 'será', 'seremos', 'seréis', 'será
n', 'sería', 'serías', 'seríamos', 'seríais', 'serían', 'era', 'eras', 'éramos', 'erais', 'eran', 'fui', 'fuiste', 'fue', 'fuim
os', 'fuisteis', 'fueron', 'fuera', 'fueras', 'fuéramos', 'fuerais', 'fueran', 'fuese', 'fueses', 'fuésemos', 'fueseis', 'fuese
n', 'sintiendo', 'sentido', 'sentida', 'sentidos', 'sentidas', 'siente', 'sentid', 'tengo', 'tienes', 'tiene', 'tenemos', 'tené
is', 'tienen', 'tenga', 'tengas', 'tengamos', 'tengáis', 'tengan', 'tendré', 'tendrás', 'tendrá', 'tendremos', 'tendréis', 'ten
drán', 'tendría', 'tendrías', 'tendríamos', 'tendríais', 'tendrían', 'tenía', 'tenías', 'teníamos', 'teníais', 'tenían', 'tuv
e', 'tuviste', 'tuvo', 'tuvimos', 'tuvisteis', 'tuvieron', 'tuviera', 'tuvieras', 'tuviéramos', 'tuvierais', 'tuvieran', 'tuvie
se', 'tuvieses', 'tuviésemos', 'tuvieseis', 'tuviesen', 'teniendo', 'tenido', 'tenida', 'tenidos', 'tenidas', 'tened']
```



Expresiones Regulares en Python

- Es una herramienta poderosa incorporada en Python para encontrar patrones en un texto y realizar operaciones sobre el mismo.
- Usos:
 - If - then: se cumplen condiciones, realizar una operación
 - Data Cleaning
 - Reemplazo de caracteres
 - Ajustes según necesidad

```
# Probar con expresiones regulares
import re
#Solo quiero Letras
letras = re.sub("[^a-zA-ZáóéíúñÑ]", " ", desc[1] )
print (letras)
```

```
Principales Funciones Responsable de los procesos de selección de mandos medios y gerenciales Responsable de la mejora co
ntinua en los procesos de reclutamiento administrativo Gestionar los procesos de desarrollo organizacional y líneas de carre
ra Requisitos Bachiller Titulado en psicología Mandatorio Experiencia mínima de años en selección especialmente g
erencialy mandos medios Experiencia en headhunting Experiencia en evaluación de candidatos psicológicas y por competenci
as El contenido de este aviso es de propiedad del anunciante Los requisitos de la posición son definidos y administrados po
r el anunciante sin que Bumeran sea responsable por ello
```



Scikit-learn

- Librería de machine learning para Python.
- Incluye todos los algoritmos principales de regresión, clasificación y clustering.
- <https://scikit-learn.org/stable/>



TF-IDF en Scikit-learn

- **Term Frequency:** a más frecuencia de términos, mayor el score.
- **Inverse Document Frequency:** cuanto más raro el término, mayor el score.

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(desc_limpio)
print (tfidf_matrix.shape)
```

(12659, 32303)

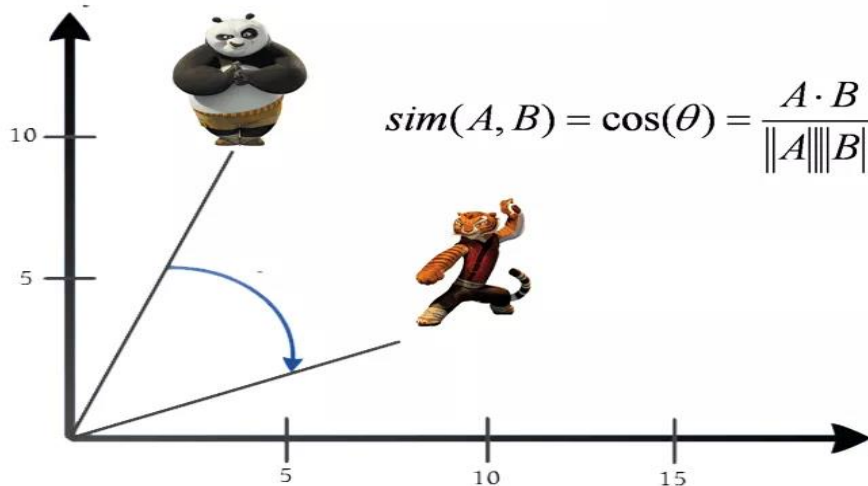
Entrenamiento del modelo a partir de las palabras en la colección de documentos usando TF-IDF

<https://www.kdnuggets.com/2018/08/wtf-tf-idf.html>



Similitud de documentos usando Distancia Coseno

Cosine Similarity



```
#obtener el nombre del documento y crear un dataframe
res = cosine_similarity(tfidf_matrix2, tfidf_matrix, True)
res = res[0]
```

res
array([0.01664071, 0.05082784, 0.02006109, ..., 0.06865837, 0.01540713, 0.04398107])

- <http://techinpink.com/2017/08/04/implementing-similarity-measures-cosine-similarity-versus-jaccard-similarity/>
- <http://blog.christianperone.com/2011/09/machine-learning-text-feature-extraction->



Ejercicio N° 3.2: NLTK y scikit-learn

Al finalizar el laboratorio, el alumno logrará:

- Ejecutar un caso de NLP centrado en encontrar la similitud entre un CV y perfil laboral usando NLTK y scikit-learn.



Lecturas Adicionales

Para obtener información adicional, puede consultar los siguientes enlaces:

- <https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72>
- <https://towardsdatascience.com/how-to-get-started-in-nlp-6a62aa4eaeff>



Resumen

En este capítulo, usted aprendió que:

- NLP es una rama de inteligencia artificial, que le da a una máquina la capacidad de entender el lenguaje humano.
- Existen distintas aplicaciones en la industria de NLP. Además, se ha profundizado la clasificación de documentos usando librerías para Python: NLTK y scikit-learn.



Tarea N° 3: NLP – casos de uso

Investigar otro caso de uso de NLP y enviar un informe de dos hojas máximo.

- Elegir un caso (no similitud de documentos)
 - Asistente chatbot
 - Filtro spam
 - Buscador de lenguaje natural
 - Otro
- Contexto del caso
- Aplicación en la industria
- Técnicas de NLP usadas

Enviar por correo al instructor.

