

Análisis de texto y datamining con Pattern

Cómo realizar búsquedas en Flickr, Twitter, Wikipedia Análisis de texto automático, plural, singular, análisis de verbos

Sofía Martin

Noviembre 2018

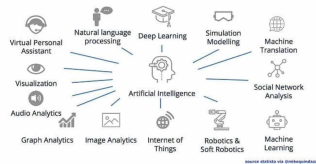


Contenido





- 1 Introducción
- 2 ¿Y cómo puedo manipular el texto?
- 3 ParseTree
- 4 Buscando en Web



- Pattern is a web mining module for the Python programming language.
- para Python 3 se debe instalar la versión de desarrollo.
- <https://www.clips.uantwerpen.be/pattern>



¿Qué puedo hacer?

Uso	Módulo
	pattern.en es de fr it nl
	pattern.web
	pattern.db
	pattern.search

¿Qué puedo hacer?

Procesar texto en varios idiomas. Clasificar. Conjuguar.

```
from pattern.en import parse
def procesol(s):
    """Muestra en crudo la info que muestra cuando procesa texto"""
    s = parse(s, relations=True, lemmata=True)
    print(s)
```

```
procesol(wuthering)
```

```
We/PRP/B-NP/0/NP-SBJ-1/we crowded/VBD/B-VP/0/VP-1/crowd round/NN/B-NP/0/NP-1
over/IN/B-PP/B-PNP/0/over Miss/NNP/B-NP/I-PNP/0/miss Cathy/NNP/I-NP/I-PNP/0
/s head/NN/I-NP/0/NP-SBJ-2/head I/PRP/I-NP/0/NP-SBJ-2/i had/VBD/B-VP/0/VP-2
P/O/NP-OBJ-2/peep at/IN/B-PP/B-PNP/0/at a/DT/B-NP/I-PNP/0/a dirty/JJ/I-NP/I
J/I-NP/I-PNP/0/ragged ,/,/I-NP/I-PNP/0/, black-haired/JJ/I-NP/I-PNP/0/black
0/0/0; big/JJ/B-ADJP/0/0/big enough/IN/B-PP/0/0/enough both/DT/0/0/both
and/CC/0/0/0/and talk/NN/B-NP/0/0/talk :/:0/0/0/: indeed/RB/B-ADVP/0/0/ind
```

- Verbos **VB** - Sustantivos **NN**
- Contexto: **SBJ**: sujeto de la oración,
- Conjugación
- Singular - Plural

¿Qué dice acá?

Parseo la salida

```
from pattern.en import parse
from pattern.en import pprint

pprint(parse('We crowded round, and over Miss Cathy's heads I had a peep at
```

WORD	TAG	CHUNK	ROLE	ID	PNP	LEMMA
We	PRP	NP	SBJ	1	-	we
crowded	VBD	VP	-	1	-	crowd
round	NN	NP	OBJ	1	-	round
,	-	-	-	-	-	,
and	CC	-	-	-	-	and
over	IN	PP	-	-	PNP	over
Miss	NNP	NP	-	-	PNP	miss
Cathy	NNP	NP	^	-	PNP	cathy
,	-	-	-	-	-	,
's	-	-	-	-	-	's
I	PRP	NP	SBJ	1	-	i
had	VBD	VP	-	1	-	have
a	DT	NP	-	1	-	a
peep	V	VP	-	1	-	peep
at	PP	PP	-	1	-	at

- Parse devuelve un string con tag unicode con la clasificación
- TAG indica el sentido en la oración: pronombre, verbo, sustantivo, etc
- CHUNK indica el contexto de la oración en que se encuentra la palabra.
- LEMMA forma simplificada de la palabra: Verbo en infinitivo, sustantivo singular

¿Y en castellano?

- pattern.es
- No es tan completo.
- Están documentados 600 verbos.
- La manipulación es más compleja por la estructura del idioma.

```
from pattern.es import parse
from pattern.es import pprint

pprint(parse(cumbres))
```

WORD	TAG	CHUNK	ROLE	ID
He	MD	VP	-	-
vuelto	VBN	VP ^	-	-
hace	VB	VP ^	-	-
unos	DT	NP	-	-
instantes	NNS	NP ^	-	-
de	IN	PP	-	-
visitar	VB	VP	-	-
a	IN	PP	-	-
mi	PRP\$	NP	-	-
casero	JJ	NP ^	-	-
v	CC	-	-	-

Buscar sustantivos y agregar el artículo: en inglés no en castellano

```
from pattern.en import referenced, tag
def imprimo_sustantivos(s):
    s = tag(s)
    for w, pos in s:
        if pos == 'NN':
            print(referenced(w))
imprimo_sustantivos(wuthering)
```

```
a round
a head
a peep
a child
a talk
a face
a s
a round
a gibberish
a nobody
a fling
```


Comparativo y superlativo: en inglés no es castellano

```
from pattern.en import comparative, superlative
def superlativos(s):
    for w in s.split():
        print('N: {}, C: {}, S: {}'.format(w, comparative(w),
```

```
superlativos('bad nice tall')
```

N: bad, C: worse, S: worst

N: nice, C: nicer, S: nicest

N: tall, C: taller, S: tallest

ParseTree - ordenando la salida ordenada

Te genera una estructura de objetos que permite manipular mejor el análisis. Tipos de objetos:

- Text
- Sentence
 - Padre de cada oración
 - Objetos word de la oración
 - Lista de adjetivos, objeto, predicado, verbos
- Word
 - En que oración se encuentra
 - Lemma
- Chunk

El objeto Texto en un conjunto de oraciones (Sentence), que es un conjunto de palabras (Word), que están en un contexto(CHUNK). Puedo obtener información contextual de cada objeto.

```
from pattern.es import tag
from pattern.es import parsetree
def info_sentences(s):
```

Cambiar de plural a singular y viceversa

```
def colecto_sustantivos(s):
    """Arma una lista de los sustantivos en singular y plural """
    s = parsetree(s, relations=True, lemmata=True)
    list_palabras = [t for w in s.sentences for t in w.words if t.pos == 'NN' or t.pos == 'NNS']
    return list_palabras

def cambio_plu_sing(lista):
    """Invierte los sustantivos, de plural a singular y viceversa"""
    list_palabras = [pluralize(x.string) if x.pos == 'NN' else singularize(x.string) for x in lista]
    print(list_palabras)

l = colecto_sustantivos(wuthering)
cambio_plu_sing(l)

['rounds', 'heads', 'peeps', 'children', 'talks', 'faces', 'ss', 'foot', 'rounds', 'gibberishes', 'nobodi
gs', 'door', 'fashions', 'brats', 'houses', 'bairn', 'masters', 'matters', 'halves', 'fatigues', 'tales',
ses', 'street', 'owners', 'souls', 'monies', 'times', 'homes', 'expense', 'conclusions', 'mistresses', 't
ild']
```

¿Qué cambia en estas dos salidas?

```
def coleccion_sustantivos_es(s):
    """Arma una lista de los sustantivos en singular y plural """
    s = parsetree(s, relations=True, lemmata=True)
    list_palabras = [t for w in s.sentences for t in w.words if t.pos == 'NN' or t.pos == 'NNS']
    return list_palabras

def cambio_plu_sing_es(lista):
    """Invierte los sustantivos, de plural a singular y viceversa"""
    list_palabras = [pluralize(x.string) if x.pos == 'NN' else singularize(x.string) for x in lista]
    print(list_palabras)

l = coleccion_sustantivos_es(cumbres)
cambio_plu_sing_es(l)

['haces', 'uno', 'visitares', 'mis', 'caseros', 'yas', 'figuras', 'solitarios', 'vecinos',
'causas', 'estes', 'bellos', 'países', 'podidos', 'res', 'mases', 'agradables', 'enes',
'habriamos', 'unas', 'parejas', 'compañero', 'parecidos', 'extraos', 'rdinarios', 'yes',
'contrarios', 'simpatias', 'los', 'contrarios', 'motivos', 'los', 'dedos', 'mases', 'nos']

from pattern.es import pluralize, singularize
from pattern.es import parsetree
def coleccion_sustantivos_es(s):
    """Arma una lista de los sustantivos en singular y plural """
    s = parsetree(s, relations=True, lemmata=True)
    list_palabras = [t for w in s.sentences for t in w.words if t.pos == 'NN' or t.pos == 'NNS']
    return list_palabras

def cambio_plu_sing_es(lista):
    """Invierte los sustantivos, de plural a singular y viceversa"""
    list_palabras = [pluralize(x.string) if x.pos == 'NN' else singularize(x.string) for x in lista]
    print(list_palabras)

l = coleccion_sustantivos_es(cumbres)
cambio_plu_sing_es(l)

['instante', 'figuras', 'vecinos', 'inquietarmes', 'países', 'misántropos', 'encontramos',
'compañero', 'hombres', 'hombres', 'extraos', 'rdinarios', 'simpatias', 'contrarios', 'dedo',
'párpado', 'nunciare', 'nombres', 'preguntarles']
```

Ser cuidadoso con los módulos que estamos trabajando

Buscar adjetivos

```
from pattern.es import tag
from pattern.es import parsetree

def colecto_adjetivos(lista):
    """Arma una lista de los adjetivos"""

    s = parsetree(lista, relations=True, lemmata=True)
    list_palabras = [t for w in s.sentences for t in w.words if t.pos == 'JJ']
    return list_palabras
colecto_adjetivos(cumbres)

[Word('casero/JJ'),
 Word('solitario/JJ'),
 Word('bello/JJ'),
 Word('agradable/JJ'),
 Word('ideal/JJ'),
 Word('espontánea/JJ')]
```

Nuestras frases tienen sentimientos

```
from pattern.en import sentiment
print(sentiment(wuthering))

(0.06290726817042605, 0.562781954887218)
```

- Sólo en inglés.
- Análisis de los adjetivos.
- (Polaridad -1.0, +1.0 - Subjectividad 0.0, +1.0)
- Considera positivo mayor +0.1.

Otras funcionalidades

- `pattern.vector`: se utiliza para machine learning
- `pattern.search`: búsqueda de palabras por tag

```
from pattern.es import parsetree
from pattern.search import search
busco_adjetivos(cumbres)
```

```
[Match(words=[Word('casero/JJ')]), Match(words=[Word('solitario/JJ')]), Match(words=[Word('agradable/JJ')]), Match(words=[Word('ideal/JJ')]), Match(words=[Word('...')])]
```

- `pattern.graph`: representación de la estructura con las facilidades que se necesitan de un grafo, relaciones semánticas

¿Dónde puedo buscar?

- Twitter.
- Buscadores
- Flickr
- Wikipedia
- Facebook

```
def trending_topics():  
    print (Twitter().trends(cached=False))  
trending_topics()
```

```
['Eibar', '#SmallBusinessSaturday', '#Amici18', '#FelizSábado', '#FreddieMercury', '가갸갸갸',  
'Nicolas Roeg', 'Mustafa Cengiz', 'TEAM NEWS', 'syamuさん', '갸갸갸갸', 'Champs-Élysées',  
'#NousToutes', '#ÖnceMilletÖnceMemleket', '#24novembre', '#AbuDhabiGP', 'لمستقبل في جملة',  
'かすぎで伝わらないモノマネ', '#갸갸갸갸갸갸갸', '#صباح الفصحى المفاالى', '#갸갸갸갸갸갸', '#Misi', '#  
#UdineseRoma', '#TKO2018', '#UFCBeijing', '#OTDirecto24NOV', '#محمد بن سلمان في الاشارات',  
'IV', '#M05BYB', '#TimbreoNacional', '#DUP18', 'الاتفاق التعاون', '#ObrigadoRenato', '#MUN  
KeepAwayFrom', '#nonunadimeno', '#SabadoDetremuraSDV', '#ST0PSánchez']
```


Buscando en Twitter

```
from pattern.web import Twitter

def busco_en_twitter(s):
    t = Twitter()
    i = None
    for j in range(3):
        for tweet in t.search(s, start=i, count=10):
            print(tweet.text)
            print("-----")
            i = tweet.id

busco_en_twitter('river')
```

RT @Fercaveoficial: Mas unidos que nunca !!! En las buenas y en las malas! Vamos River bertadores

RT @robertoayala10: ES HOY VAMOS RIVER DE MI VIDA

RT @FOXSportsArg: "PITY MARTÍNEZ ES EL MESSI DE RIVER"

#90MinutosFOX | La llamativa comparación de @PolloVignolo de la figura de Riv... <https://>

@benitez98_u Si gana river te caigo con un vino

Buscando en Flickr

```
def buscar(quebusco):  
    engine = Flickr(license=None)  
  
    q = quebusco  
    results = engine.search(q, size=MEDIUM, sort=RELEVANCY, cached=False)  
    for img in results:  
        #print(img.url) # Retrieving the actual image URL executes a query.  
        print(img.text)  
        print(img.author)  
        print('\n')
```

Sitios Web restricciones

- Con License Key paga
 - Google (search)
 - Bing (search, news, image)
 - Yahoo (search, news, image)
- Cantidad de usos por tiempo
 - Twitter
 - Facebook

¿preguntas?

- @entrerrianas
- Sofía Martin