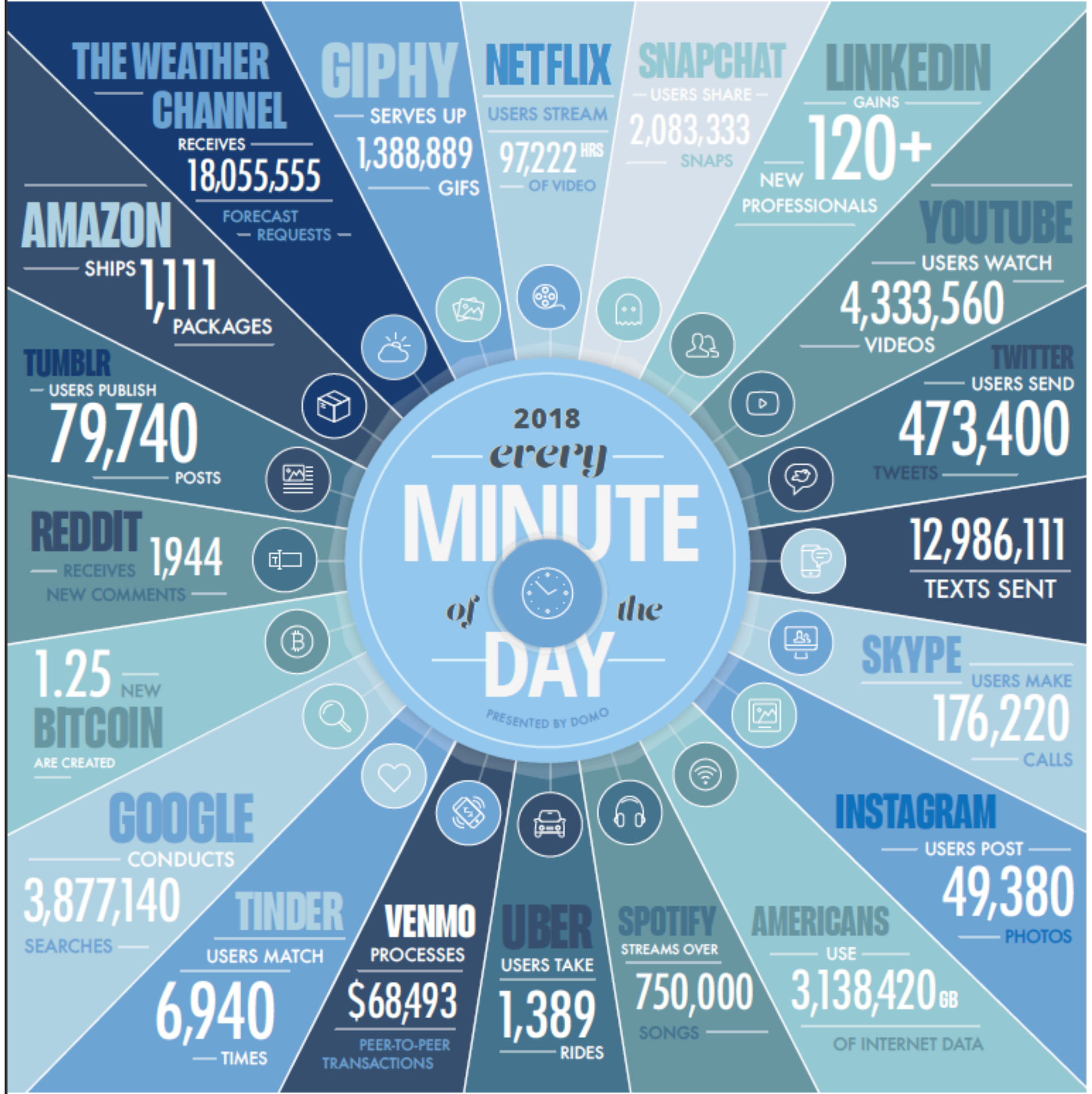
INTRODUCCIÓN AL PROCESAMIENTO DE LENGUAJE NATURAL (NLP)

Universidad del Valle de Guatemala


2018
every
MINUTE
of the
DAY
PRESENTED BY DOMO
THE WEATHER CHANNEL
RECEIVES
18,055,555
FORECAST REQUESTS
GIPHY
SERVES UP
1,388,889
GIFS
NETFLIX
USERS STREAM
97,222 HRS
OF VIDEO
SNAPCHAT
USERS SHARE
2,083,333
SNAPS
LINKEDIN
GAINS
NEW 120+
PROFESSIONALS
AMAZON
SHIPS 1,111
PACKAGES
YOUTUBE
USERS WATCH
4,333,560
VIDEOS
TUMBLR
USERS PUBLISH
79,740
POSTS
TWITTER
USERS SEND
473,400
TWEETS
REDDIT
RECEIVES 1,944
NEW COMMENTS
12,986,111
TEXTS SENT
1.25 NEW
BITCOIN
ARE CREATED
SKYPE
USERS MAKE
176,220
CALLS
GOOGLE
CONDUCTS
3,877,140
SEARCHES
INSTAGRAM
USERS POST
49,380
PHOTOS
TINDER
USERS MATCH
6,940
TIMES
VENMO
PROCESSES
$68,493
PEER-TO-PEER TRANSACTIONS
UBER
USERS TAKE
1,389
RIDES
SPOTIFY
STREAMS OVER
750,000
SONGS
AMERICANS
USE
3,138,420 GB
OF INTERNET DATA

¿QUÉ PODEMOS HACER CON TEXTO?

- *Formatear texto, tratar de entender lo que dice,*
- *Encontrar, identificar o extraer información relevante,*
- *Clasificar documentos,*
- *Encontrar documentos relevantes,*
- *Análisis de sentimientos,*
- *Modelos de tópicos*

MANEJAR TEXTO EN PYTHON

Construcciones primitivas:

- *oraciones, frases,*
- *palabras o tokens,*
- *caracteres,*
- *documentos*

```
>>> text1 = 'Un intelectual es el que dice una cosa simple de un modo
complicado. Un artista es el que dice una cosa complicada de un modo simple.'
>>> len(text1)
133
>>> text2 = text1.split(' ')
>>> len(text2)
26
>>> text2
['Un', 'intelectual', 'es', 'el', 'que', 'dice', 'una', 'cosa', 'simple',
'de', 'un', 'modo', 'complicado.', 'Un', 'artista', 'es', 'el', 'que',
'dice', 'una', 'cosa', 'complicada', 'de', 'un', 'modo', 'simple.']
```

Buscando palabras específicas

Palabras largas. Palabras que tengan más de 3 letras, por ejemplo:

```
>>> [w for w in text2 if len(w)>3]
['intelectual', 'dice', 'cosa', 'simple', 'modo', 'complicado.', 'artista',
'dice', 'cosa', 'complicada', 'modo', ‘simple.']
```

Palabras que inicien con mayúscula:

```
>>> [w for w in text2 if w.istitle()]
['Un', ‘Un']
```

Palabras que terminen con a:

```
[w for w in text2 if w.endswith('a')]
['una', 'cosa', 'artista', 'una', 'cosa', 'complicada']
```

Palabras únicas: usando `set()`

```
>>> text3 = 'Ser o no ser'
>>> text4 = text3.split(' ')
>>> len(text4)
4
>>> len(set(text4))
4
>>> set(text4)
{'Ser', 'o', 'no', 'ser'}
>>> len(set([w.lower() for w in text4]))
3
>>> set([w.lower() for w in text4])
{'o', 'no', 'ser'}
```

Algunas funciones para comparar palabras

```
➤ s.startswith(t)

➤ s.endswith(t)

➤ t in s

➤ s.isupper(); s.islower(); s.istitle()

➤ s.isalpha(); s.isdigit(); s.isalnum()
```

Operaciones sobre `strings`

```
➤ s.lower(); s.upper(); s.titlecase()

➤ s.split(t)

➤ s.splitlines()

➤ s.join(t)

➤ s.strip(); s.rstrip(); s.lstrip()

➤ s.find(t); s.rfind(t)

➤ s.replace(u, v)
```

Ejemplo

```
>>> text5 = 'ouagadougou'
>>> text6 = text5.split('ou')
>>> text6
['', 'agad', 'g', '']
>>> 'ou'.join(text6)
'ouagadougou'
>>> text5.split('')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: empty separator
>>> list(text5)
['o', 'u', 'a', 'g', 'a', 'd', 'o', 'u', 'g', 'o', 'u']
>>> [c for c in text5]
['o', 'u', 'a', 'g', 'a', 'd', 'o', 'u', 'g', 'o', 'u']
```

Limpiando texto

```
>>> text8 = '           Un intelectual es el que dice una cosa simple de
un modo complicado. Un artista es el que dice una cosa complicada de un
modo simple.   '
>>> text8.split(' ')
['', '', '\t\tUn', 'intelectual', 'es', 'el', 'que', 'dice', 'una',
'cosa', 'simple', 'de', 'un', 'modo', 'complicado.', 'Un', 'artista',
'es', 'el', 'que', 'dice', 'una', 'cosa', 'complicada', 'de', 'un',
'modo', 'simple.', '', '', '']
>>> text9 = text8.strip()
>>> text9
'Un intelectual es el que dice una cosa simple de un modo complicado. Un
artista es el que dice una cosa complicada de un modo simple.'
>>> text9.split(' ')
['Un', 'intelectual', 'es', 'el', 'que', 'dice', 'una', 'cosa', 'simple',
'de', 'un', 'modo', 'complicado.', 'Un', 'artista', 'es', 'el', 'que',
'dice', 'una', 'cosa', 'complicada', 'de', 'un', 'modo', 'simple.']
```

Modificando texto

```
>>> text9
'Un intelectual es el que dice una cosa simple de un modo complicado. Un
artista es el que dice una cosa complicada de un modo simple.'
>>> text9.find('o')
35
>>> text9.rfind('o')
124
>>> text9.replace('o', 'O')
'Un intelectual es el que dice una cOsa simple de un mOdO cOmplicadO. Un
artista es el que dice una cOsa cOmplicada de un mOdO simple.'
```

Textos mas extensos

Leyendo archivos línea por línea

```
>>> f = open('cabeza_de_vaca.txt', 'r')
>>> f.readline()
'The Project Gutenberg EBook of Naufragios de Alvar Nunez Cabeza de
Vaca\n'
>>> f.readline()
'by Alvar Nunez Cabeza de Vaca\n'
```

Leyendo el archivo completo

```
>>> f.seek(0)
0
>>> text12 = f.read()
>>> len(text12)
434426
>>> text13 = text12.splitlines()
>>> len(text13)
7044
>>> text13[0]
'The Project Gutenberg EBook of Naufragios de Alvar Nunez Cabeza de Vaca'
```

Operaciones con archivos

```
➤ f = open(filename, mode)

➤ f.readline(); f.read(); f.read(n)

➤ for line in f: do_something(line)

➤ f.seek(n)

➤ f.write(message)

➤ f.close()

➤ f.closed()
```

Algunos detalles sobre leer archivos de texto

```
>>> f = open('cabeza_de_vaca.txt')
>>> text14 = f.readline()
>>> text14
'The Project Gutenberg EBook of Naufragios de Alvar Nunez
Cabeza de Vaca\n’
```

Para eliminar al caracter de nueva línea

```
>>> text14.rstrip()
'The Project Gutenberg EBook of Naufragios de Alvar Nunez
Cabeza de Vaca'
```

EXPRESIONES REGULARES

```
tweet = "@nltk Text analysis is awesome! #regex #pandas #python"
```

¿Cómo extraemos los hashtags?

EXPRESIONES REGULARES

```
tweet = "@nltk Text analysis is awesome! #regex #pandas #python"
```

¿Cómo extraemos los hashtags?

```
print([word for word in tweet.split(' ') if word.startswith('#')])
```

```
>>> text10 = 'Going to play golf right now with @AbeShinzo.
Japan loves the game. Tremendous fans of @JackNicklaus,
@TigerWoods, and @PhilMickelson — I said what about @GaryPlayer,
they said we love Gary too!'
>>> text11 = text10.split(' ')
>>> [w for w in text11 if w.startswith('@')]
['@AbeShinzo.', '@JackNicklaus,', '@TigerWoods,',
'@PhilMickelson', '@GaryPlayer,']
```

Encontrando patrones con expresiones regulares

```
@AbeShinzo, @JackNicklaus, @ Guatemala
```

Queremos comparar con algo después de @.

- *alfabetos,*
- *números,*
- *símbolos especiales (como '_').*

```
@[A-Za-z0-9_]+
```

```
>>> text10 = 'Going to play golf right now with @AbeShinzo.
Japan loves the game. Tremendous fans of @JackNicklaus,
@TigerWoods, and @PhilMickelson — I said what about @GaryPlayer,
they said we love Gary too!'
>>> text11 = text10.split(' ')
>>> [w for w in text11 if w.startswith('@')]
['@AbeShinzo.', '@JackNicklaus,', '@TigerWoods,',
'@PhilMickelson', ‘@GaryPlayer,']
```

Importamos primero la librería `re`

```
>>> import re
>>> [w for w in text11 if re.search('@[A-Za-z0-9_]+',w)]
['@AbeShinzo.', '@JackNicklaus,', '@TigerWoods,',
'@PhilMickelson', '@GaryPlayer,']
```

Entendiendo la expresión regular `@[A-Za-z0-9_]+`

- `@` — inicia con @
- `[A-Za-z0-9_]` — seguido de cualquier alfabeto (en minúsculas o mayúsculas), cualquier dígito o guión bajo
- `+` — que se repite una o varias veces

Metacaracteres: coincidencias

`.` : comodín, vale por cualquier caracter ,
`^` : inicio de un `string`,
`$` : fin de un `string`,
`[]` : coincide con uno de los caracteres `a,…,z`
`[^abc]` : coincide con un carácter que no es `a,b` o `c`,
`a|b` : coincide con `a` o `b`, donde `a` o `b` son strings,
`\` : coincide con alguna de los caracteres especiales,
`\b` : frontera de una palabra,
`\d` : cualquier dígito, equivalente a `[0-9]`,
`\D` : cualquier no dígito, equivalente a `[^0-9]`,
`\s` : cualquier `whitespace`, equivalente a `[ \t\n\r\f\v]`,
`\S` : cualquier no `whitespace`, equivalente a `[^ \t\n\r\f\v]`,
`\w` : caracter alfanumérico, equivalente a `[a-zA-z0-9_]`,
`\W` : caracter no alfanumérico, equivalente a `[^a-zA-z0-9_]`.

Metacaracteres: repeticiones

`*` : comodín por cero o más caracteres,
`+` : comodín por uno o más caracteres,
`?` : comodín por cero o un caracter,
`{n}` : exactamente `n` repeticiones, `n>=0`,
`{n,}` : por lo menos `n` veces,
`{,n}` : a lo más `n` veces,
`{m,n}` : por lo menos `m` veces, a lo más `n` veces.

```
>>> text10 = 'Going to play golf right now with @AbeShinzo.
Japan loves the game. Tremendous fans of @JackNicklaus,
@TigerWoods, and @PhilMickelson — I said what about
@GaryPlayer, they said we love Gary too!'
>>> text11 = text10.split(' ')

>>> [w for w in text11 if re.search('@[A-Za-z0-9_]+',w)]
['@AbeShinzo.', '@JackNicklaus,', '@TigerWoods,',
'@PhilMickelson', ‘@GaryPlayer,']

>>> [w for w in text11 if re.search('@\w+', w)]
['@AbeShinzo.', '@JackNicklaus,', '@TigerWoods,',
'@PhilMickelson', '@GaryPlayer,']
```

Otros ejemplos

```
>>> text12 = 'ouagadougou'

>>> re.findall(r'[aeiou]', text12)
['o', 'u', 'a', 'a', 'o', 'u', 'o', 'u']

>>> re.findall(r'[^aeiou]', text12)
['g', 'd', 'g']
```

PROCESAMIENTO DE LENGUAJE NATURAL

El lenguaje natural es el lenguaje usado por los humanos.

El language natural evoluciona:

- *nuevas palabras se agregan (tuitear),*
- *palabras viejas pierden popularidad,*
- *el significado de las palabras cambia,*
- *las reglas de lenguaje cambian.*

Algunas tareas de procesamiento de lenguaje natural:

- *contar palabras, contar frecuencia de palabras,*
- *encontrar fronteras de oraciones,*
- *identificar los elementos de oraciones,*
- *identificar roles semánticos (María ama a José),*
- *identificar entidades en oraciones,*
- *encontrar que pronombres se refieren a qué entidades,*
- *etc.*

`input = “Lista listado listas”`

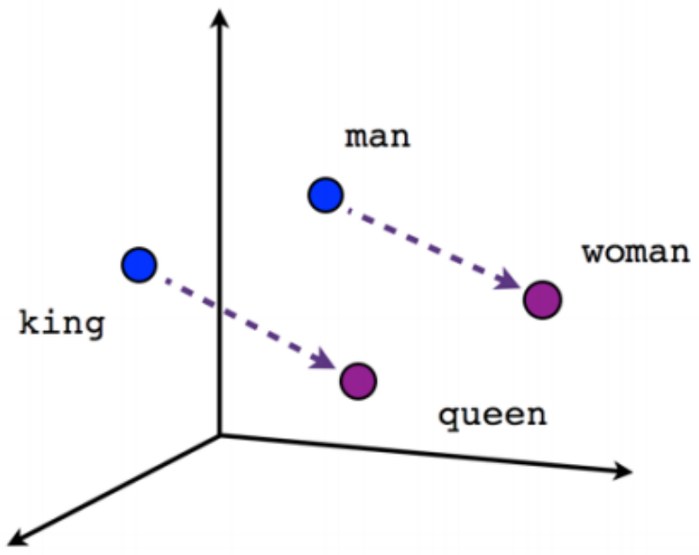
Normalización

[lista, listado, listas]

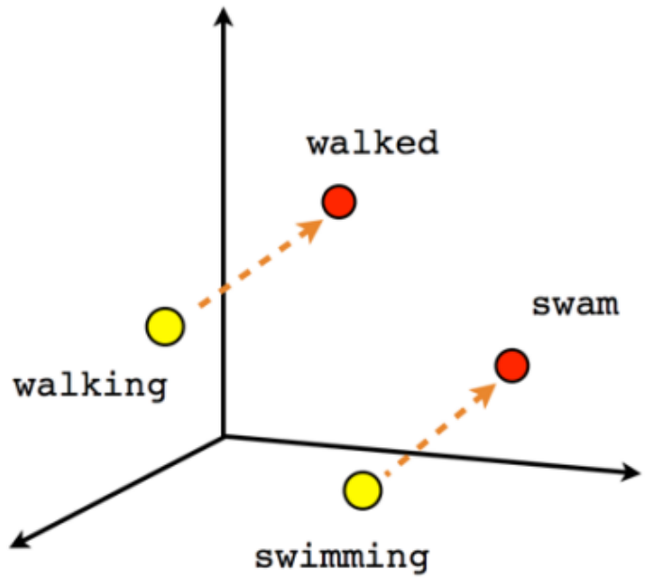
Lematización

Todas las palabras anteriores tienen una raíz común

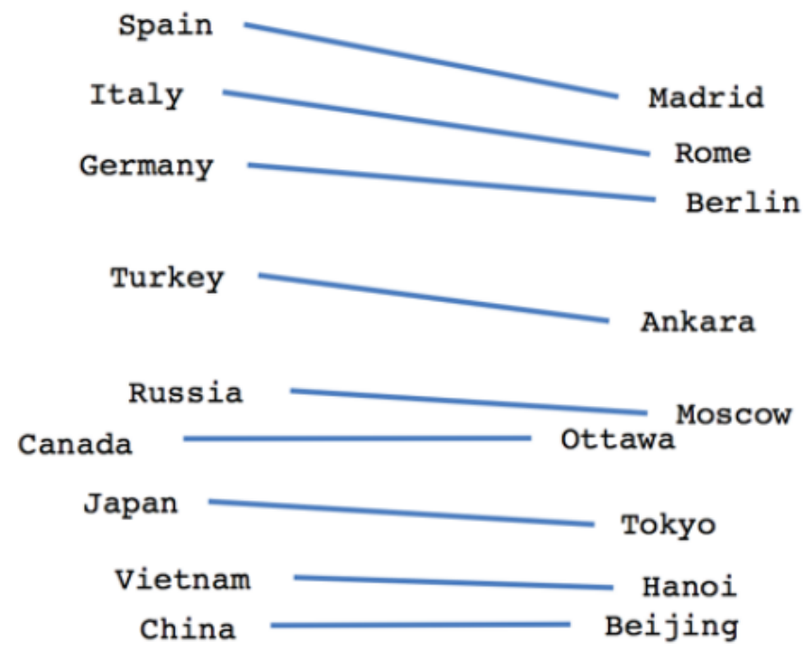
WORD EMBEDDINGS



Male-Female

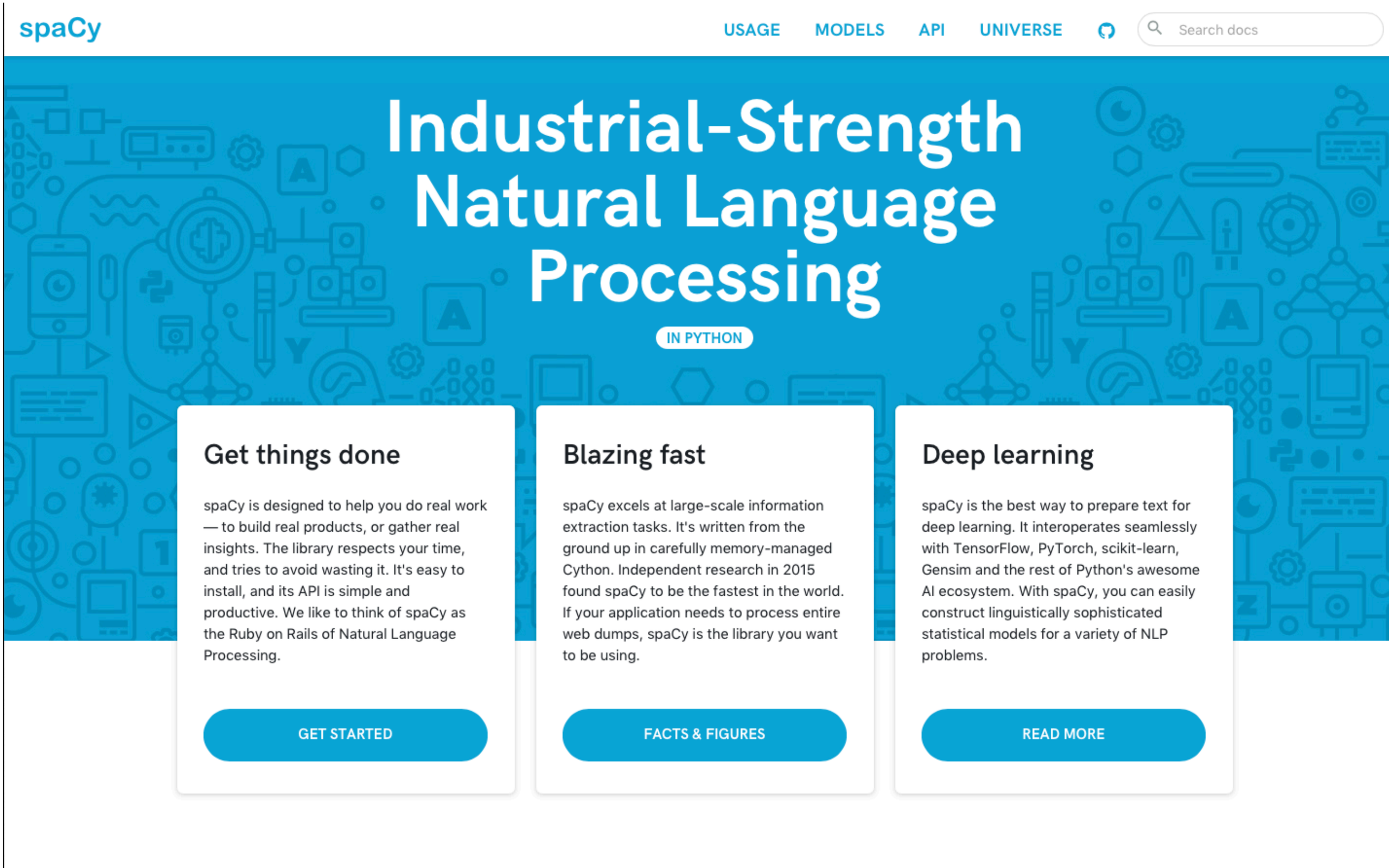


Verb tense



Country-Capital

NLP CON SPACY



Instalar

```
pip install -U spacy
python -m spacy download es
python -m spacy download es_core_news_md
python -m spacy download en
```

Actualizar

```
pip install -U spacy
python -m spacy validate
```