

Procesamiento de Lenguaje Natural



Contenido resumido de la asignatura

Primer parcial:

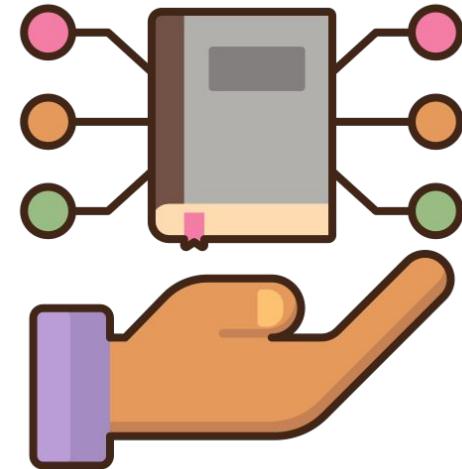
- Introducción al Procesamiento de Lenguaje Natural
- Procesamiento básico de texto
- Clasificación de texto y BoW

Segundo parcial:

- Redes neuronales y word Embeddings
- Desarrollo de Chatbots

Evaluación final:

- Modelos conversacionales
- Modelos Largos de Lenguaje (LLM)



Conocimientos previos

Programación y Lenguajes:

- Python
- Programación estructurada y orientada a objetos

Frameworks y librerías:

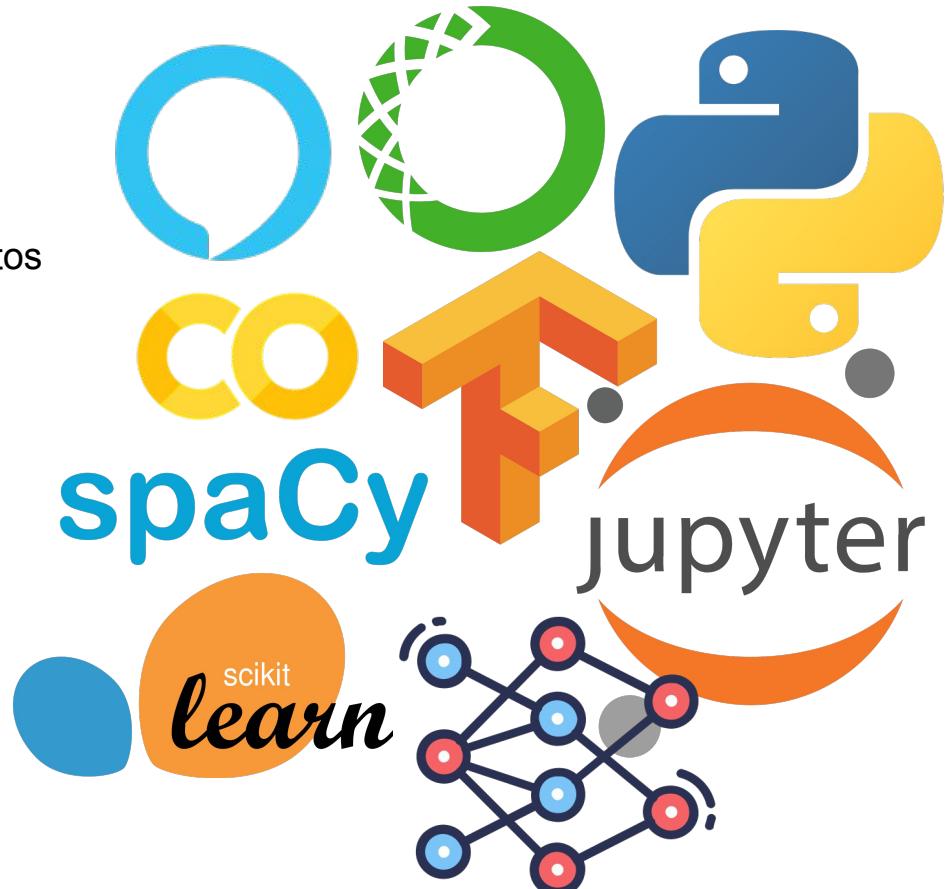
- Keras / Tensorflow
- Numpy, Pandas, Sklearn

Ambientes de trabajo:

- Jupyter Notebook
- Colab (Google)
- Anaconda / Miniconda

Conceptos básicos de IA:

- ML y modelos de clasificación
- Conceptos de redes neuronales



Ambiente de trabajo

1.- Descargar miniconda

<https://docs.conda.io/en/latest/miniconda.html>

Platform	Name	SHA256 hash
Windows	Miniconda3 Windows 64-bit	00e8370542836862d4c790aa8966f1d7344a8addd4b766004febcb23f40e2914
	Miniconda3 Windows 32-bit	4fb64e6c9c28b88beab16994bfba4829110ea3145baa60bda5344174ab65d462
macOS	Miniconda3 macOS Intel x86 64-bit bash	1622e7a0fa60a7d3d892c2d8153b54cd6ffe3e6b979d931320ba56bd52581d4b
	Miniconda3 macOS Intel x86 64-bit pkg	2236a243b6cbe6f16ec324ecc9e631102494c031d41791b44612bbb6a7a1a6b4
	Miniconda3 macOS Apple M1 64-bit bash	c8f436dbde130f171d39dd7b4fcfa669c223f130ba7789b83959adc1611a35644
	Miniconda3 macOS Apple M1 64-bit pkg	837371f3b6e8ae2b65bdfc8370e6be812b564ff9f40bcd4eb0b22f84bf94fe5
Linux	Miniconda3 Linux 64-bit	634d76df5e489c44ade4085552b97bebc786d49245ed1a830022b0b406de5817
	Miniconda3 Linux-aarch64 64-bit	3962738cfac270ae4ff30da0e382aecf6b3305a12064b196457747b157749a7a
	Miniconda3 Linux-ppc64le 64-bit	92237cb2a443dd15005ec004f2f744b14de02cd5513a00983c2f191eb43d1b29
	Miniconda3 Linux-s390x 64-bit	221a4cd7f0a9275c3263efa07fa37385746de884f4306bb5d1fe5733ca770550



(Seleccionar la versión adecuada según las especificaciones del equipo del estudiante)

Ambiente de trabajo

2.- Instalación de Miniconda

Windows	Linux / MAC OS
Ejecutar el archivo Miniconda3-latest-Windows-x86_64 descargado	Ir a la ruta de descarga y ejecutar bash <i>Miniconda3-latest-Linux-x86_64.sh</i> desde la terminal de Linux, o a bash Miniconda3-latest-MacOSX-x86_64.sh desde la terminal de Mac.



```
-pc:~$ cd Downloads/
-pc:~/Downloads$ bash Miniconda3-latest-Linux-x86_64.sh
```

Miniconda3 will now be installed into this location:
/home/[REDACTED]/miniconda3

- Press ENTER to confirm the location
- Press CTRL-C to abort the installation
- Or specify a different location below

```
[/home/[REDACTED]/miniconda3] >>>
```

Ambiente de trabajo

3.- Creación del ambiente virtual

Windows	Linux / MAC OS
Desde Anaconda prompt ejecutar:	Desde una nueva terminal ejecutar:

conda env list

Después, ejecutar la creación del ambiente virtual

conda create -n [env_name] python=3.9 tensorflow

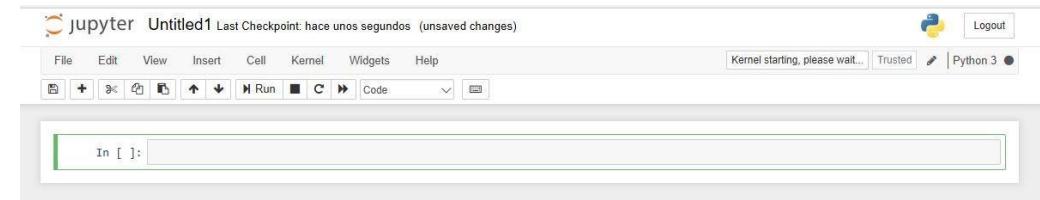
Al escribir nuevamente el comando para listar los ambientes, debería de ver el ambiente con el nombre seleccionado

A continuación ejecutar:

conda activate [env_name]

Ambiente de trabajo

4.- Instalación de dependencias



Ahora, con el ambiente activado, instalaremos las dependencias necesarias para trabajar en el curso:

```
conda install jupyter spacy numpy keras matplotlib pandas nltk scikit-learn gensim
```

Instalamos también el núcleo de trabajo de spacy:

```
conda install -c conda-forge spacy  
python -m spacy download es_core_news_sm
```

Finalmente, abrimos nuestro espacio de trabajo (jupyter) utilizando el siguiente comando:

```
jupyter notebook
```

Ejemplo de primer programa en Jupyter



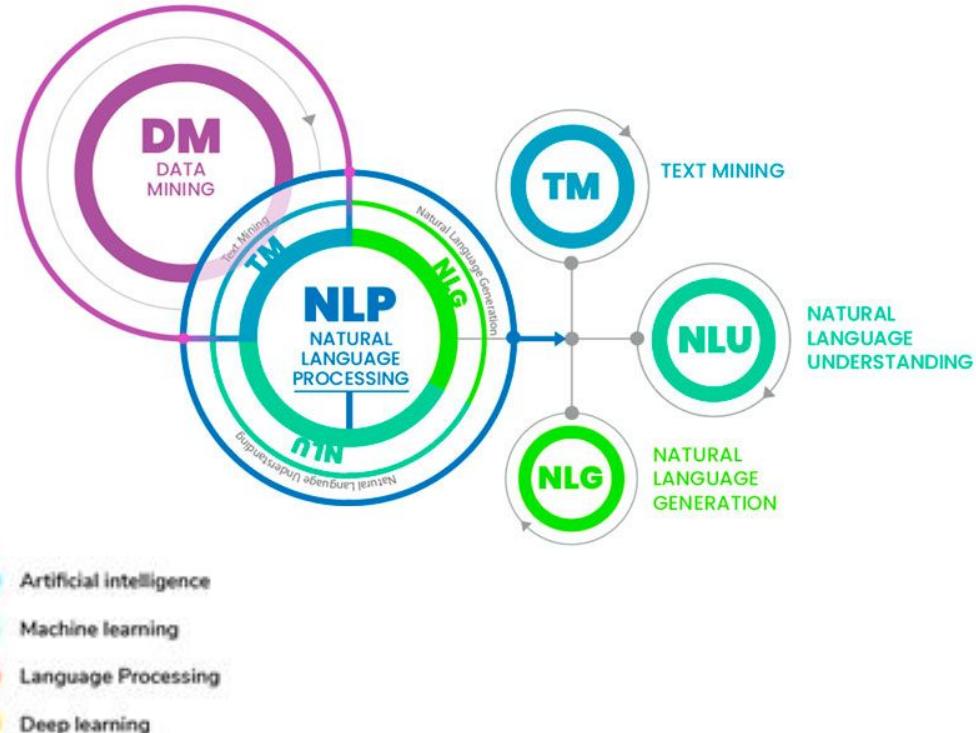
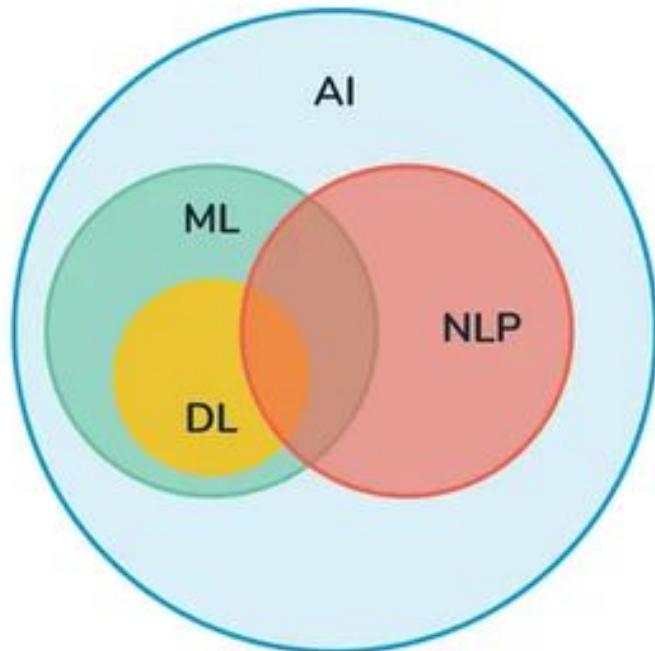
```
1 # Código para probar el ambiente de trabajo
2 from spacy import displacy
3 import spacy
4
5 pln_es = spacy.load("es_core_news_sm")
6
7 Texto = pln_es(u"¡Este es el hola mundo del Procesamiento de Lenguaje Natural. Universidad Panamericana")
8 displacy.render(Texto, style = 'ent', jupyter = True, options = {'distance': 110})
```

¡Este LOC es el hola mundo del Procesamiento de Lenguaje Natural MISC . Universidad Panamericana LOC

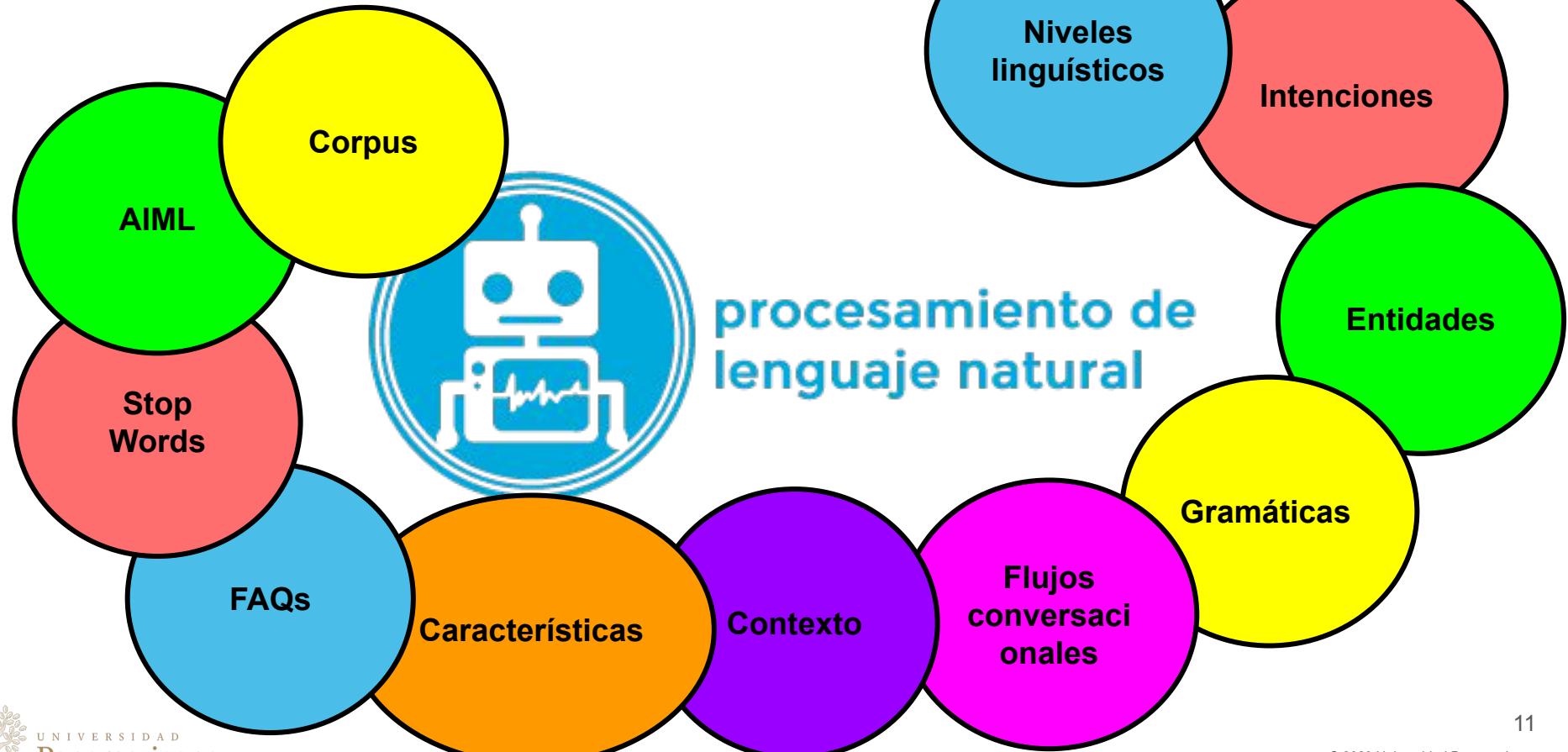
Introducción al Procesamiento del Lenguaje Natural



¿Qué es el procesamiento de lenguaje natural?



Conceptos básicos del PLN



El mundo y el PLN



Un poco de historia...

El procesamiento del lenguaje natural nace en la década de **1960**, como una subárea de la **Inteligencia Artificial** y la **Lingüística**, para estudiar problemas sobre la comprensión automática del **lenguaje natural**.

Década de los 50:

- Interés en traducción automática
- Poco interés en comprensión y significado de las frases



Década de los 60:

- Surge el procesamiento por frases
- Surgen las primeras y simples redes neuronales

Década de los 70:

- Técnicas para comprensión basadas en reglas del lenguaje
- Integración de PLN con aprendizaje automático

1966 (Primer chatbot):

Se crea “ELIZA”, chatbot desarrollado por el alemán Joseph Weizenbaum que representa a un psicoanalista e imita sus respuestas típicas.

Esta es una de las pocas situaciones humanas donde se puede responder con una pregunta sin dar ningún tipo de información al interlocutor

young woman: Men are all alike.

eliza: In what way?

young woman: They are always bugging us about something specific or other.

eliza: Can you think of a specific example?

young woman: Well, my boyfriend made me come here.

eliza: Your boyfriend made you come here?

young woman: He says I'm depressed much of the time.

eliza: I am sorry to hear you are depressed.

young woman: It's true. I'm unhappy.

eliza: Do you think coming here will help you not to be unhappy?

young woman: I need some help, that much seems certain.

eliza: What would it mean to you if you got some help?

young woman: Perhaps I could learn to get along with my mother.

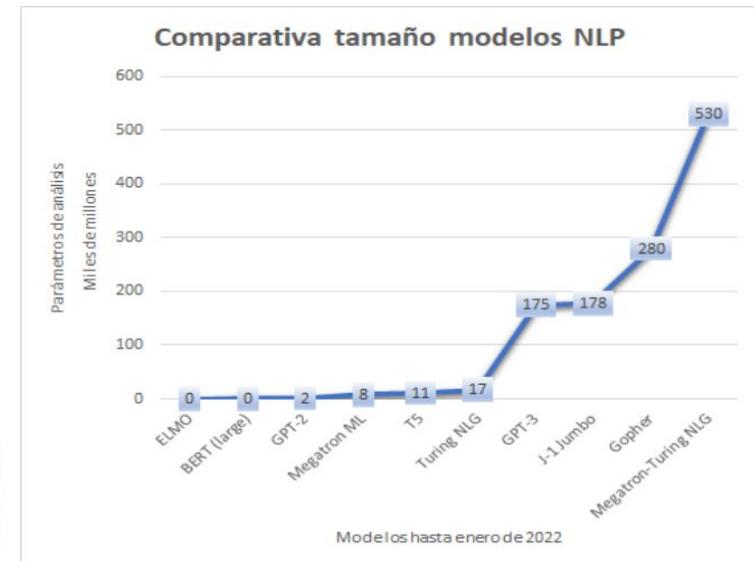
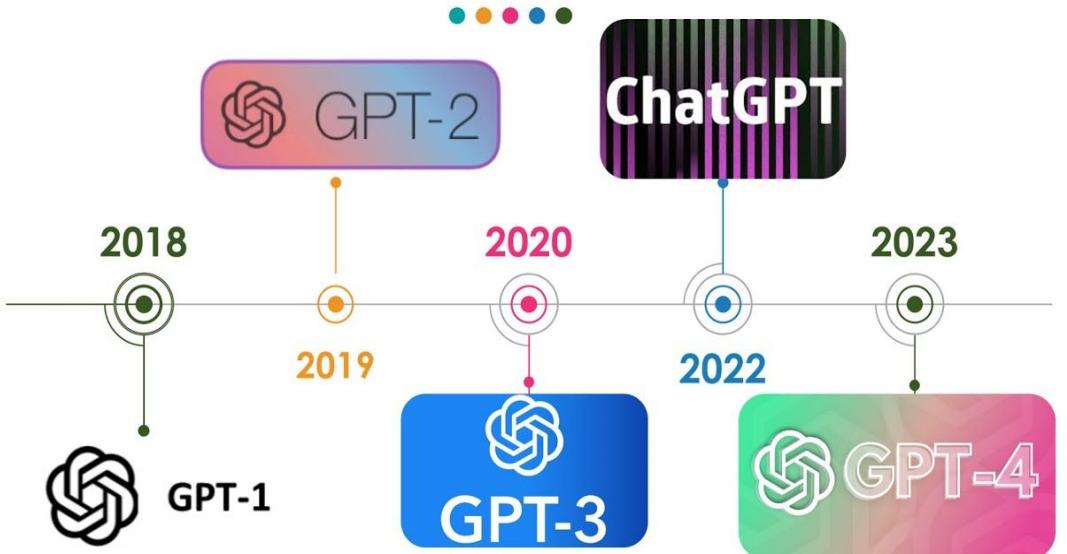
eliza: Tell me more about your family.

young woman: My mother takes care of me.

eliza: Who else in your family takes care of you?

2018 (Se lanza GPT-1):

Con el modelo GPT-1 comienza la era formal de los LLMs o Modelos Largos de Lenguaje, los cuales son modelos de PLN a base de transformers y que tienen miles de millones de parámetros de entrenamiento



¿Para qué se utiliza el PLN? - Aplicaciones comunes

C clientes y opiniones

- Análisis de quejas y calidad del servicio
- Análisis de sentimientos y opiniones
- Clasificación de atención a clientes



Análisis de documentos

- Resumen de textos
- Clasificación de documentos

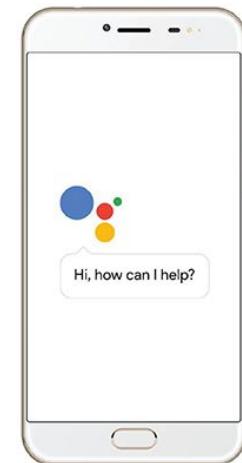


Publicidad y mercadotecnia

- Sistemas de recomendación

Herramientas

- Correctores de texto
- Predictores de texto
- Traductores
- Chatbots
- Asistentes virtuales



Modelado del Lenguaje



Tony: “Shit!”

Steve: “Language!”

Tokenización

Proceso que consiste en la división de elementos **ESCRITOS**, en instancias **NO SEPARABLES**:

Texto: *Si la montaña viene hacia ti... corre, porque se está derrumbando.*

Tokens: | *Si* | *la* | *montaña* | *viene* | *hacia* | *ti* | ... | *corre* | , | *porque* | *se* | *está* | *derrumbando* | . |

Texto: **¿Por qué nos caemos? Para que podamos aprender a recuperarnos (BATMAN)**

Tokens: | **¿** | **Por** | **qué** | **nos** | **caemos** | **?** | **Para** | **que** | **podamos** | **aprender** | **a** | **recuperarnos** | (| **BATMAN** |) |

Texto: **Op. cit. “Todos flotan... y tú también flotarás cuando estés aquí”**

Tokens: | **Op** | . | **cit** | . | “ | **Todos** | **flotan** | ... | **y** | **tú** | **también** | **flotarás** | **cuando** | **estés** | **aquí** | “ |

Parts of Speech (PoS)



Los tokens se clasifican según su uso como elementos del discurso. Dependen de su posición en la frase, y relación con otras palabras.

Ejemplo:

La familia de mi esposa **vino** muy rápido y desde muy lejos para Navidad, tomamos **vino** y comimos pavo

Lematización

Lematizar una palabra consiste en llevar las palabras **a su forma raíz**, es decir, su forma base:
Sin conjugaciones, plurales, géneros, adverbios, etc

Ejemplos:

- **Plurales** - Se transforman en singular
Los asistentes virtuales -> El asistente virtual
- **Conjugaciones verbales** - Se convierten todos a infinitivo
“Yo... Soy IronMan” -> “Yo... Ser IronMan”
- **Adverbios** - Se convierten en sustantivos
Pensé rápidamente... -> Pensar rapidez...
- **Adjetivos** - Se convierten en la palabra de la que derivan (Sustantivo)
Es un bello atardecer -> Ser un belleza atardecer
- **Géneros** - Las palabras con género femenino (En español) se convierten en masculino
La casa roja -> El casa rojo

Nota: Para un sistema de NLP generalmente **no importa que no tenga sentido gramatical la oración**

¿Por qué la lematización ayuda en el Español?

¿Es más difícil el inglés o el español?



Do, did, done, doing



Hacer, hecho, haciendo, hago, haces, hace, hacemos, hacen, hacia, haciais, hacia, haciamos, haciais, hacian, haciais, hice, hiciste, hizo, hicimos, hicisteis, hicieron, hiciste, haga, hagas, hagamos, hagais, hiciere, hicieres, hiciere, hiciereis, hicieres, haz, hagamos, haced, hagan, etc...



Go, goes, went,
gone, going.

Ir, yendo, ido, voy, vas, va, vamos,
vais, van, iba, ibas, íbamos, ibais,
iban, fui, fuiste, fue, fuimos, fuisteis,
fueron, iré, irás, irá, iremos, iréis, irán,
iría, irías, iríamos, iríais, irían, vaya,
vayas, vayamos, vayáis, vayan,
fuere, fueres, fuéremos, fuereis,
fueren, fuera, fueras, fuéramos,
fuerais, fueran, fuese, fueses,
fuésemos, fueseis, fuesen.

Artículo	Inglés	Español
indefinido	a	un / una
	an	unos / unas
definido	the	el / la los / las



Ejemplo de uso

```
1 # Importamos Spacy y descargamos el nucleo de trabajo en Español
2 import spacy
3
4 core_esp = spacy.load("es_core_news_sm")
5
6 # core_esp se convierte en un método que Tokeniza el texto dentro
7 Texto = core_esp(u"Si no sabes exactamente a dónde quieres ir, entonces, \
8                 realmente no importa cuál camino elijas")
9 for token in Texto:
10    print("Texto:", token.text) #.text extrae el texto del token
11    print("PoS:", token.pos_)  #.pos_ extrae el part of speech
12    print("Lemma:", token.lemma_) #.lemma_ extrae la lematización
13    print("\n")
```

Texto: Si
PoS: SCONJ
Lemma: si

Texto: no
PoS: ADV
Lemma: no

Texto: sabes
PoS: VERB
Lemma: saber

Texto: exactamente
PoS: ADV
Lemma: exactamente

Texto: a
PoS: ADP
Lemma: a

https://colab.research.google.com/drive/1oW3-G1Lb_eCArC98woPbnUNLC3jgioi?usp=sharing

Ejercicio:

De los trabalenguas siguientes, determinar e imprimir cuál de ellos es más difícil de pronunciar (basado en el siguiente criterio **Dificultad = Cantidad_de_Lemmas_Diferentes / Cantidad_de_palabras_en_la_frase**)
El valor más bajo de este resultado, representará una dificultad más alta, y el valor más alto, representará una má baja.

Pista: Puedes obtener la cantidad de lemmas diferentes si los escribes en una lista y evalúas la cantidad de elementos diferentes, o si preguntas si el elemento ya existe, antes de escribirlo.

Trab1: *Cuando cuentos cuentos cuenta cuantos cuentos cuentas, así, sabrás cuantos cuentos contaste*

Trab2: *Un trabajenguista muy trabajenguado creó un trabajenguas muy trabajenguoso ni el mejor trabajenguista ni el más trabajenguado pudo trabajengüear aquel trabajenguas tan trabajenguoso*

Trab3: *Hay suecos en Suiza y hay suizos en Suecia, pero hay más suizos en Suiza que suizos en Suecia, y más suecos en suecos en Suecia que suecos en Suiza*

Trab4: *Si mi gusto gustara del gusto que gusta a mi gusto. Tu gusto gustaría del gusto que gusta mi gusto. Pero como tu gusto no gusta del gusto que gusta a mi gusto. Mi gusto no gusta del gusto que gusta a tu gusto*

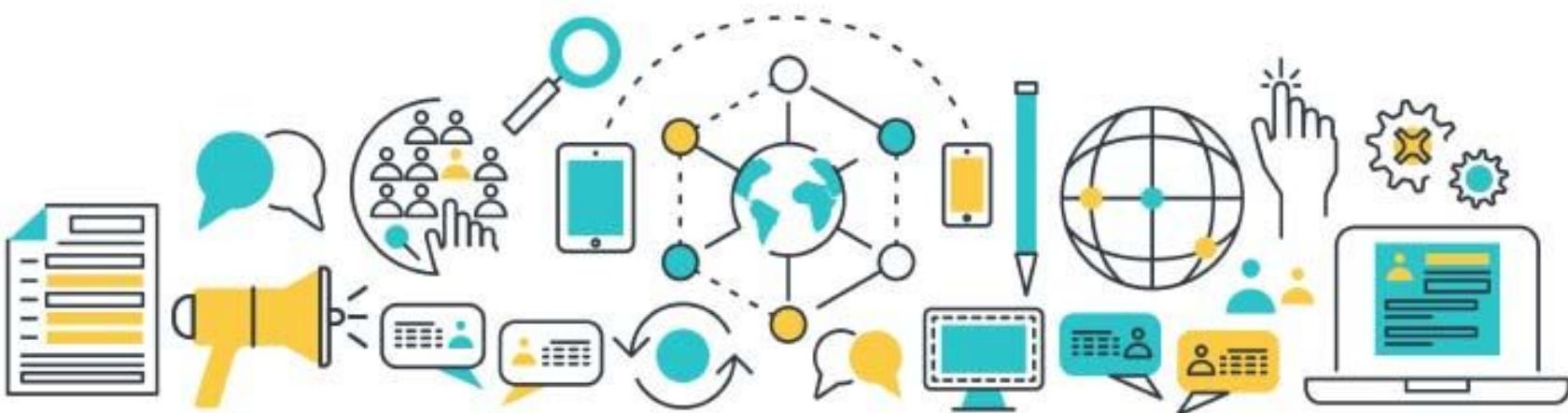
Modelado del Lenguaje (II)

Name **Date** **Designation** **Subject**

Named Entity Recognition

John McCarthy who was born on September 4, 1927 was an American computer scientist and cognitive scientist. He was one of the founders of the discipline of artificial intelligence. He co-authored the document that coined the term "Artificial intelligence" (AI), developed the programming language family Lisp, significantly influenced the design of the language ALGOL

Forma de los tokens



Forma de los tokens

Representa la **cantidad y tipo** de caracteres que conforman un token, utilizando las siguientes claves:

- x = Caracter alfabético (Minúsculas)
- X = Caracter alfabético (Mayúsculas)
- d = Número / dígito

Token original	Forma (Token Shape)
Palabra	Xxxxx
1994	ddd
6 de Julio	d xx Xxxxx
23 veces	dd xxxx
CDMX	XXXX
Actumlogos@gmail.com	Xxxxx@xxxx.xxx

Nota 01: Cada clave (x, X ó d) solo es escrita un máximo de 4 veces consecutivas. Si el token tiene más de 4 claves iguales consecutivas, son ignoradas.

Nota 02: Los caracteres que no son alfanuméricos como los signos especiales, son conservados sin cambios

```

1 # Importación de Librerías
2 import spacy
3
4 # Cargamos núcleo de trabajo (Español)
5 pln_es = spacy.load("es_core_news_md")
6
7 # Imprimimos la forma de los tokens de diferentes textos:
8 Frases = [pln_es(u'Mi celular es 55 12 34 56 78'),
9             pln_es(u'El descubrimiento de América fue el 12/10/1492'),
10            pln_es(u'Voy a P. Sherman, calle Wallaby 42 Sydney')]
11
12 # Separamos el token por medio de | (Utilizando el comando end)
13 for frase in Frases:
14     for token in frase:
15         # Imprimimos la forma de cada token
16         print(token.shape_, end='| ')
17     print("\n")

```

<https://drive.google.com/file/d/14aEmPhWrZQLs-iWStMxVi4CC7WVxBJY/view?usp=sharing>

Xx | xxxx | xx | dd | dd | dd | dd | dd | dd |

Xx | xxxx | xx | Xxxxxx | xxx | xx | dd / dd / dddd |

Xxx | x | X. | Xxxxxx | , | xxxx | Xxxxxx | dd | XXXXX |

Detección de entidades



Detección automática de entidades

Las entidades son palabras o grupos de palabras que pertenecen a grupos específicos, similar a campos semánticos, por ejemplo, nombres propios, lugares, fechas, etc.

Date - Fechas

Isaac Asimov:

Loc - Lugares

“20 de diciembre de 1919 a 2 de enero de 1929, Nueva York, Estados Unidos, fue un escritor y profesor de bioquímica en la facultad de medicina de la Universidad de Boston de origen ruso, nacionalizado estadounidense, conocido por ser un prolífico autor de obras de ciencia ficción, historia y divulgación científica. Asimov, asimismo, tenía un dilatado conocimiento sobre las ciencias naturales en todo su conjunto.

Sign - Materia

Litt - Obras

La obra más famosa de Asimov es la *Saga de la Fundación*, también conocida como *Trilogía o Ciclo de Trántor*, que forma parte de la serie del *Imperio Galáctico* y que más tarde combinó con su otra gran serie sobre los robots. También escribió obras de misterio y fantasía, así como una gran cantidad de textos de no ficción. En total, firmó más de 500 volúmenes y unas 9000 cartas o postales. Sus trabajos han sido publicados en 9 de las 10 categorías del Sistema Dewey de clasificación.”

Numb - Núm.

Misc - Otros

NOTA: Son uno de los elementos más importantes en los ChatBots, tanto automáticas como customizadas.

```
1 # Importación de la librería de SpaCy y displacy
2 import spacy
3
4 # Cargamos núcleo de trabajo
5 pln_es = spacy.load("es_core_news_md")
6
7 Ejemplos = [pln_es(u'Este año, nos vamos a mudar a la CDMX!'),
8             pln_es(u'Para más información, enviar un correo a la dirección actumlogo@gmail.com')
9             pln_es(u'Viaje de 5km hasta CDMX, Costos de viaje en Uber $125.50')]
10
11 for ejemplo in Ejemplos:
12     for entidad in ejemplo.ents:
13         print("Texto:", entidad)
14         print("Tipo:", entidad.label_)
15         print('\n')
```

https://drive.google.com/file/d/1BNvoDmHrDsoOnP4qjmOIIAai_bYgW4tUI/view?usp=sharing

Texto: CDMX
Tipo: LOC

Texto: CDMX
Tipo: LOC

Texto: Costos de viaje en Uber
Tipo: MISC

Repaso de atributos del lenguaje

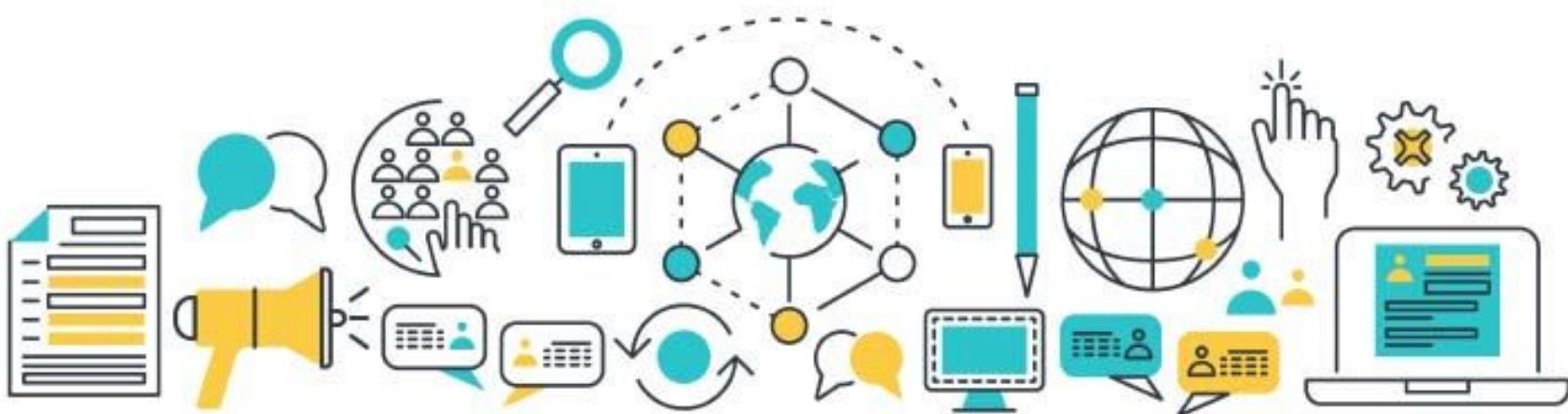


Tabla de atributos (SpaCy)

Elemento	Atributo en SpaCy	¿Qué devuelve?	Ejemplo: “Programa”
Texto	<code>.text</code>	El token convertido en <i>string</i>	Programa
Forma	<code>.shape_</code>	Código del token con las letras X, x y d	Xxxxx
Parte del discurso	<code>.pos_</code>	Parte del discurso, por ejemplo un verbo, adjetivo, o pronombre	‘NOUN’ / ‘VERB’
Lemma	<code>.lemma_</code>	La palabra raíz o primitiva del token	Programa / Programar
Entidades	<code>.ents</code>	Todas las entidades que reconozca	----
Entidades	<code>[entidad].label_</code>	Tipo de entidad reconocida	----

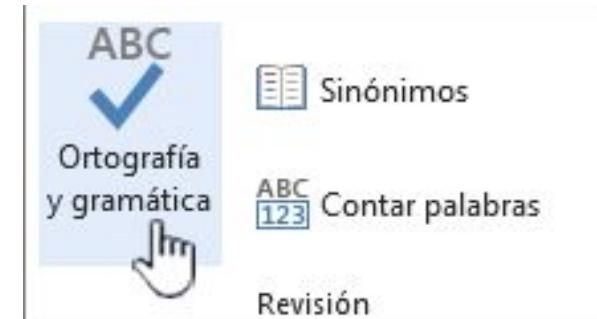
¿Qué es un corrector gramatical?

Es un sistema de revisión de textos, capaz de implementar reglas de análisis y corrección de textos basados en el idioma específico con el que se está trabajando.

Capacidad de corrección:

Entre otras cosas, un correcto gramatical puede detectar lo siguiente:

- Faltas de ortografía en palabras comunes (Por fabor → Por favor)
- Combinación incorrecta de singulares y plurales (Mi casas → Mi casa)
- Palabras repetidas (Le dije eso eso ayer → Le dije eso ayer)
- Mayúsculas en lugares incorrectos (LA tarDe → La tarde)
- Problemas con verbos auxiliares (Debes pagado → Debes pagar)
- Entre otras aplicaciones



TAREA 01: Construir un corrector gramatical que revise en un texto errores como los que se detallan a continuación. Imprimir el texto corregido marcando los cambios entre corchetes “[]”
Mostrar la cantidad y tipo de correcciones realizadas:

- **Caso 1:** Implementar un diccionario con palabras comúnmente mal escritas para revisar su ortografía
- **Caso 2:** Verificar que no exista una palabra en singular, seguida de una en plural y al revés
- **Caso 3:** Verificar que no exista la misma palabra dos veces seguidas
- **Caso 4:** Verificar que la forma de un token no mezcle mayúsculas, minúsculas y dígitos, a menos que se trate de todo en mayúsculas para marcar siglas
- **Caso 5:** Identificar cuando se escriben 2 “NOUNS” seguidos, pero no se trata de un NOMBRE propio

Tips:

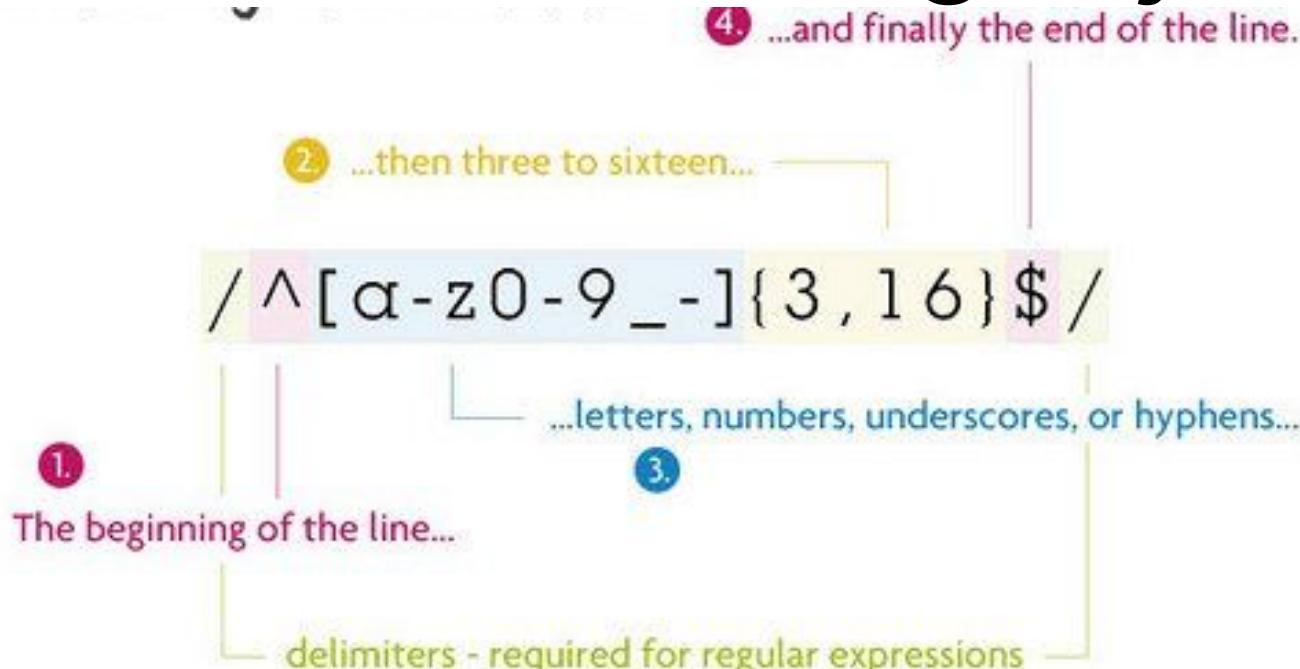
- Se debe utilizar un método por cada caso
- La tarea se trabajará directamente en un Jupyter Notebook y se deberá de poder ejecutar con la instalación realizada en clase, y en caso de usar alguna librería adicional, especificarlo en el mismo notebook

Resultado esperado:

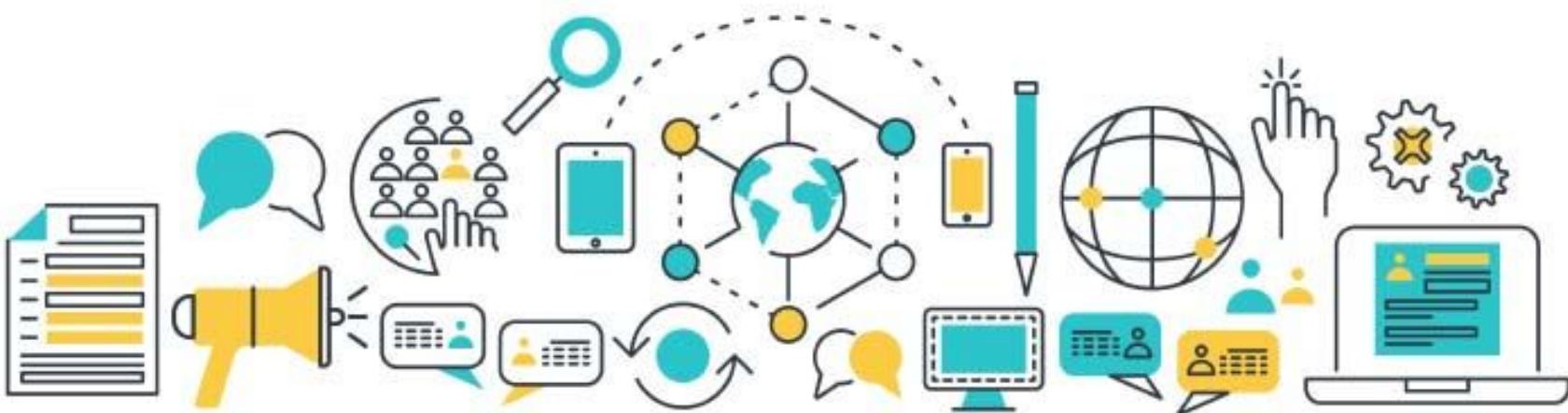
Texto para revisión: ola, me das mi saldo por fabor, grasics
4 errores de ortografía encontrados

Texto corregido: [hola], me das mi [saldo] por [favor], [gracias]

Modelado del Lenguaje (III)



Expresiones regulares



Expresiones regulares

- Es una **notación algebraica** para representar una cadena de caracteres (string)
- Necesita un **patrón** para buscar en un **texto**
- En python, el patrón se escribe `r"pattern"`

Enlace a probador de expresiones regulares:

[Regex101.com](https://regex101.com)

¿Por qué no coincidió en la primer frase las letras UP?

The screenshot shows the Regex101.com interface. The regular expression input field contains `: r" up`. The test string input field contains the following text:
Yo•estudio•en•la•UP
Once•upon•a•time
Los•humanos•tienden•a•supone•muchas•cosas
Antes•de•criticarme•intenta•superarme

On the right side of the interface, there are several buttons: "3 matches (15 steps, 7.9ms)", "v1", "gm", and a copy icon.

Definiciones de expresiones regulares

- Usamos [] para indicar “cualquiera de”
- Busca coincidencias con **un carácter** dentro de los corchetes

Ejemplos:

[A-z] cualquier letra

\d [0-9] cualquier dígito

\D [^0-9] cualquier no-dígito

\w [A-z0-9_] cualquier alfanumérico

\W [^w] cualquier no-alfanumérico

\s [\r\t\n\f] espacio en blanco, tabulador

\S [^s] un no-espacio en blanco

REGULAR EXPRESSION : r" [A-z]erro " gm ⚒

TEST STRING

El perro erro en el cerro.
Durante el encierro adopte un becerro.
El ferrocarril es de hierro.

REGULAR EXPRESSION : r" [^A-z]erro " gm ⚒

TEST STRING

El perro erro en el cerro.
Durante el encierro adopte un becerro.
El ferrocarril es de hierro.

- ? El carácter anterior aparece una o ninguna vez
- *
- + Al menos una coincidencia
- . Cualquier carácter
- | Puede ser una u otra coincidencia

REGULAR EXPRESSION

```
:r" cuen.
```

" gm

TEST STRING

cuando•cuentes•cuentos, •cuenta•cuantos•cuentos•cuentas⁴
así, •cuando•cuentes•cuentos•sabrás•cuántos•cuentos•contaste

REGULAR EXPRESSION

```
:r" Irving?
```

3 matches (21 steps, 0.0ms)

v1 ▾

TEST STRING

Hola, •mi•nombre•es•Irving⁴
a•veces•la•gente•lo•escribe•como•Irvin⁴
pero•a•veces•como•Irvink⁴
y•algunas•veces•mas•como•Erving

REGULAR EXPRESSION

```
:r" [0-9]+
```

" gm

TEST STRING

Mi•código•postal•es•34567⁴
Vivo•en•P.◦•Sherman•calle•Wallaby•42•Sidney⁴
Si•tienes•problemas•llama•al•911⁴
Tenemos•un•33-12, •repito, •33-12



REGULAR EXPRESSION

6 matches (198 steps, 0.1ms)

v1 ▾

```
:r" [^A-z][Ee][A-z]*
```

" gm



TEST STRING

En este ejemplo buscamos palabras
que empiezen con la misma letra, por ejemplo,
la cuál es la letra E, y además, se demuestra
cómo manejar cualquier carácter de forma controlada.

Anclas para RegEx

Buscan coincidencias en lugares específicos

- `^` al principio de la línea
- `\$` al final de la línea
- `\b` límite de palabra
- `\B` límite de una no-palabra

{n} número de coincidencias

{n,m} de m a n ocurrencias

{n,} al menos n ocurrencias

{,m} máximo m ocurrencias

REGULAR EXPRESSION

```
: / \b[A-z]*\d{4}\b
```

TEST STRING

hoy • es • 17 • de • mayo • de • 2022 • ↵

2021 • fue • un • año • horrible • ↵

el • ID • es • perro1234 • ↵

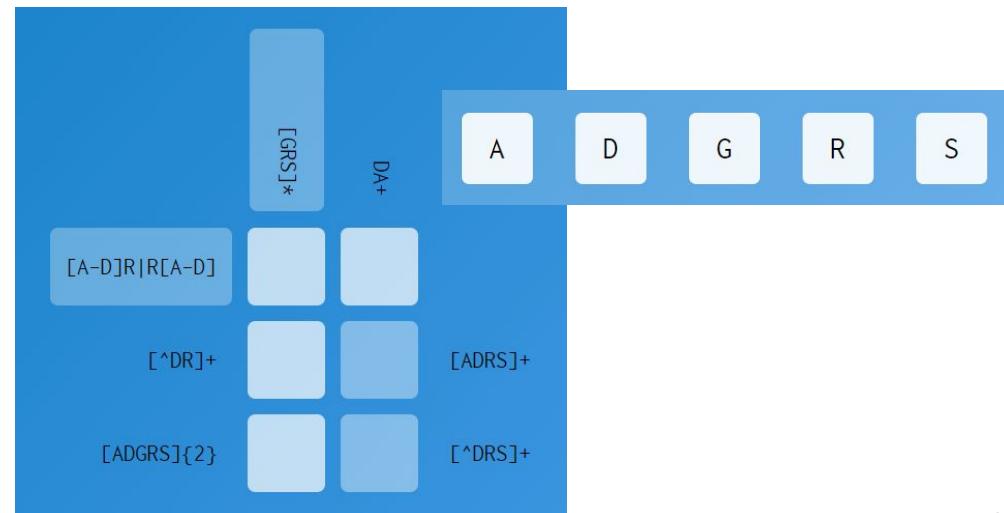
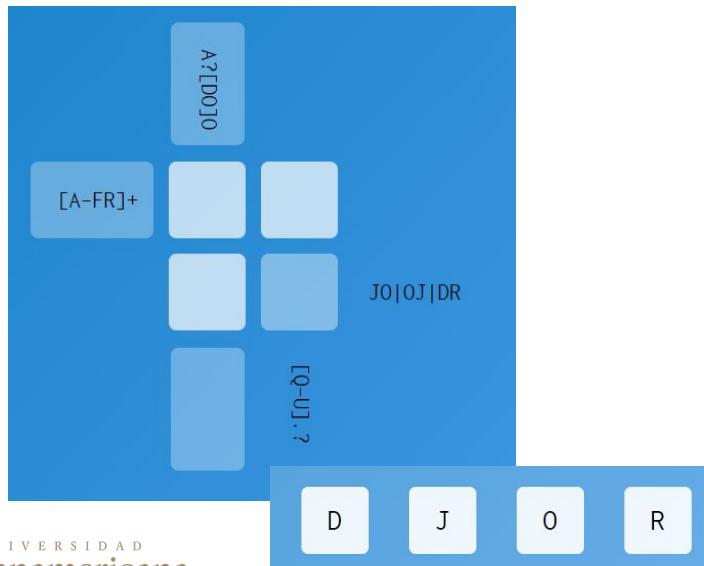
el • precio • es • de • \$1200 • ↵

con • desc12345 • hay • un • descuento • ↵

Ejercicios: RegEx

- 1) Escribir la expresión regular que pueda recuperar una fecha que tenga el formato “dd-mm-aaaa”, “dd/mm/aaaa”, o “dd mm aaaa”, y además el año puede contener 2 o 4 dígitos y debe de ser válido

- 2) Resuelve el siguiente crucigrama de RegEx



Funciones RegEx

En python puedes usar

- **[findall]** Regresa TODAS las coincidencias de una expresión regular en una lista
- **[search]** Regresa un valor booleano si encuentra el pattern dentro del texto

```
1 # Importar re
2 import re
```

```
1 # Program to extract numbers from a string
2
3 import re
4
5 string = 'Voy a P. Sherman, Calle Wallaby 42 Sidney, está en Australia\
6     en el código postal 3456'
7 pattern = '\d+'
8
9 result = re.findall(pattern, string)
10 print(result)
```

['42', '3456']

```
1 import re
2
3 string = "Yo nací en el año de 1998"
4
5 # Verificamos si 'Python' está al inicio
6 match = re.search('\d{4}', string)
7
8 if match:
9     print("Se encontró el formato buscado")
10 else:
11     print("NO se encontró el formato buscado")
```

Se encontró el formato buscado

Ejercicios: RegEx - Python

<https://colab.research.google.com/drive/1CF-ueBqQ5Shnt845566XmC9moOt6vul?usp=sharing>

- 1) **[Números telefónicos]** Escribe una expresión regular que valide un número de teléfono en los formatos (123)456-7890 o 123-456-7890.

"123-456-7890" -> Válido
"(123)456-7890" -> No válido

Nota: Solo debe aceptar números y no caracteres alfabéticos u otros caracteres especiales (Además de los paréntesis y el guión medio).

- 2) **[Dirección de correo electrónico]** Escribe una expresión regular que valide direcciones de correo electrónico donde el nombre de usuario puede contener letras, números, puntos y guiones bajos y el dominio puede contener letras y puntos. Por ejemplo:

nombre.apellido_123@ejemplo.com es válido, pero
nombre.apellido@123.com no lo es.

"nombre.apellido@ejemplo.co.mx"
-> Válido
"nombre.apellido@ejem_plo.com"
-> No válido

- 3) **[Fechas]** Escribe una expresión regular que valide fechas en el formato dd/mm/aaaa, donde el día puede variar de 01 a 31, el mes de 01 a 12, y el año debe ser de 4 dígitos.

"16/08/2023" -> Válido
"05/13/2023" -> No válido (No hay un mes 13)
"05/12/23" -> No válido (El año no tiene 4 dígitos)

N-Gramas (Fraseado)



Uso de n-gramas

https://colab.research.google.com/drive/1vy3nmQQmOMnQQMIEVUIP_oxXMiFRSgoS?usp=sharing

Consiste en unir grupos de 2 o más palabras que, debido a su naturaleza, adquieren mucho más valor cuando están juntas (como si se tratara de una sola palabra), generalmente son usadas en **negaciones**, o palabras que **siempre van juntas** para expresar una idea concreta.

Quiero realizar un retiro sin tarjeta



Quiero realizar un retiro **sin-tarjeta**

¿Cuál es el saldo en mi tarjeta oro?



Cuál es el saldo en mi **tarjeta-oro?**

No quiero realizar la operación



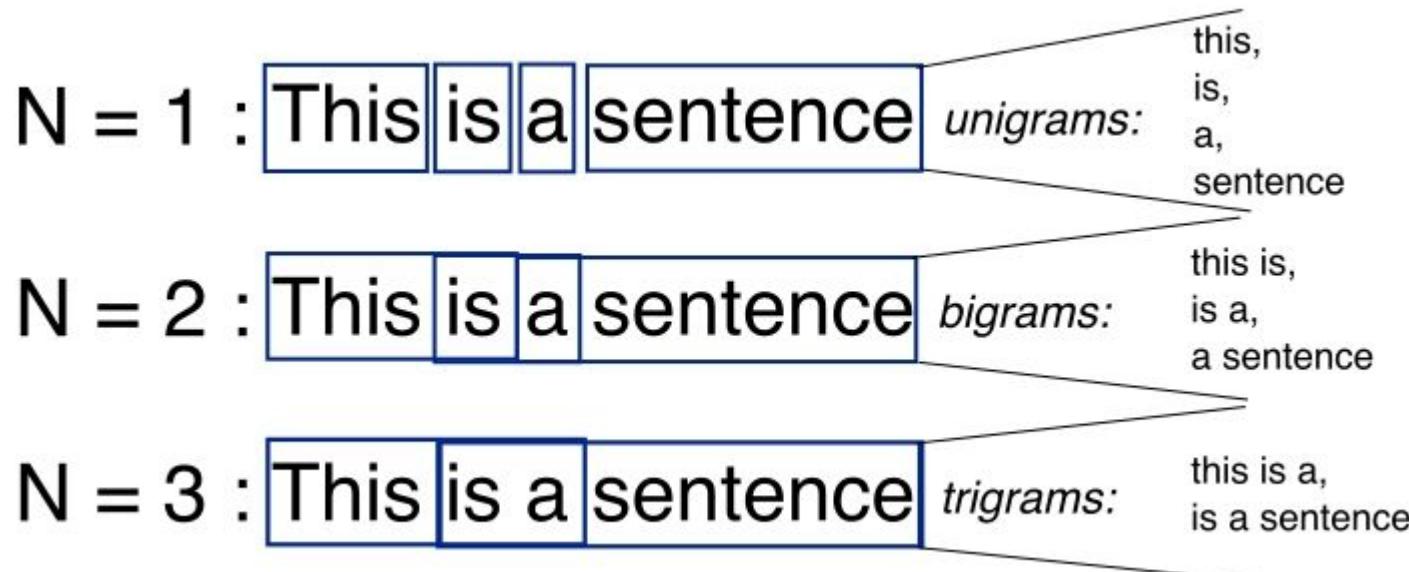
No-quiero realizar la operación

¿Qué otros ejemplos se te ocurren de posible n-gramas?

Negación

Producto específico

N-Gramas (Fraseado)



Uso de n-gramas

Consiste en unir grupos de 2 o más palabras que, debido a su naturaleza, adquieren mucho más valor cuando están juntas (como si se tratara de una sola palabra), generalmente son usadas en **negaciones**, o palabras que **siempre van juntas** para expresar una idea concreta.

Quiero realizar un retiro sin tarjeta



Quiero realizar un retiro **sin-tarjeta**

¿Cuál es el saldo en mi tarjeta oro?



Cuál es el saldo en mi **tarjeta-oro**?

No quiero realizar la operación



No-quiero realizar la operación

¿Qué otros ejemplos se te ocurren de posible n-gramas?

Negación

Producto específico

Determinación de N-gramas

$$\text{N-Grams}_K = K - (N - 1)$$

Donde:

N-Grams: Cantidad de n-gramas posibles

K: Cantidad de tokens en el texto

N: Tamaño del n-grama

Ejemplo: Calcular la cantidad de 3-gramas en el texto:

“La vida es un regalo y no pienso desperdiciarla. Nunca se sabe qué cartas repartirá la próxima vez.”

$$\text{N-Grams} = 20 - (3 - 1) = 18$$

(La vida es), (vida es un), (es un regalo), (un regalo y)
(regalo y no), (y no pienso), (no pienso desperdiciarla), ...

Ejemplo y ejercicio:

<https://colab.research.google.com/drive/1c8w46awtxzrpDDmODE0KocMuGQcpJKVC?usp=sharing>

Stopwords (Palabras vacías)

[“**This**”, “**is**”, “**a**”, “**test**”]



X

X



StopWords

Las StopWords (**Palabras vacías**) son palabras que vuelven difícil el análisis para un sistema de PLN. Pueden ser palabras **muy poco comunes o demasiado comunes**, que generen **ambigüedad**.

Por definición, siempre son StopWords:

- Artículos definidos (El, La, Los, Las)
- Artículos indefinidos (Un, Una, Unos, Unas)
- Adj. Posesivos (Mi, Tu, Su, Nuestro, Nuestra)
- Preposiciones (a, con, de, en, para, por, etc)
- Pronombres demostrativos (este, esa, aquel, etc)

Son las decisiones las que nos hacen ser quienes somos, y siempre podemos optar por hacer lo correcto" (Spiderman 3).



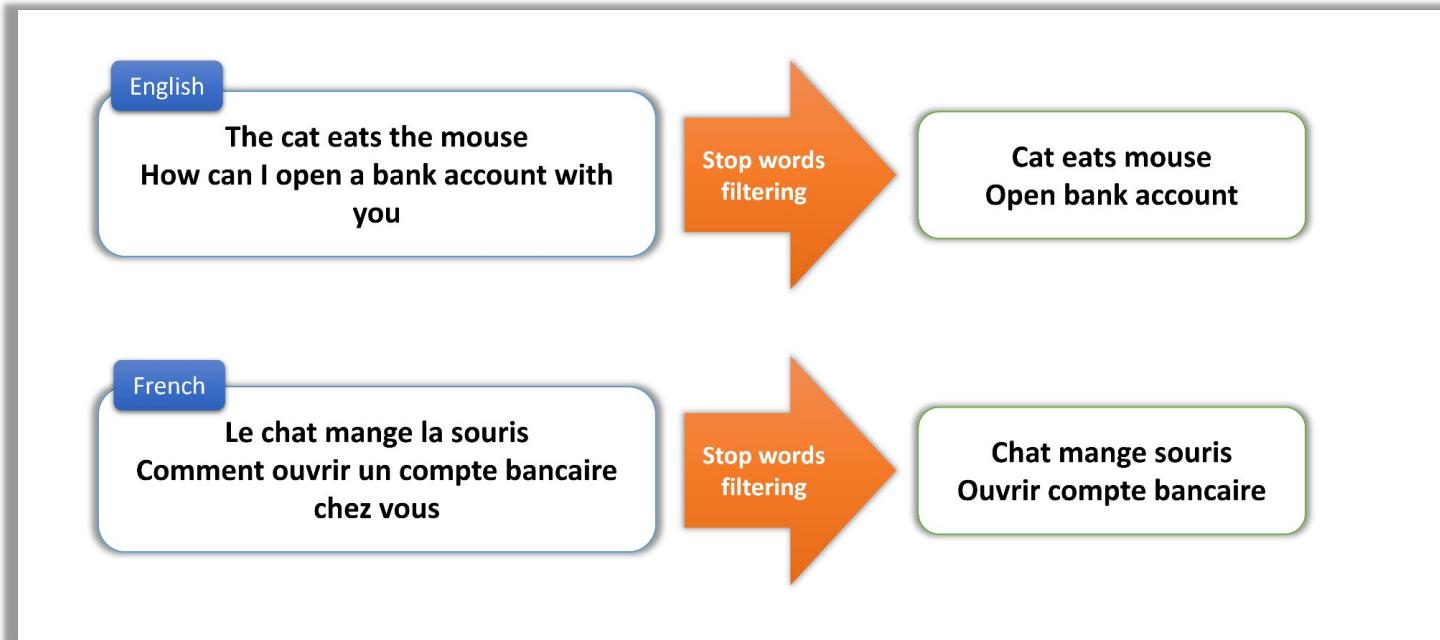
"Decisiones nos hacen ser quienes somos, siempre podemos hacer correcto" (Spiderman 3).

DATO: En el idioma japonés algunas de estas palabras no las consideran necesarias en la comprensión de ideas



Stopwords

Dependiendo del idioma, la cantidad de StopWords por defecto puede ser menor o mayor.



Ejercicio para determinar StopWords:

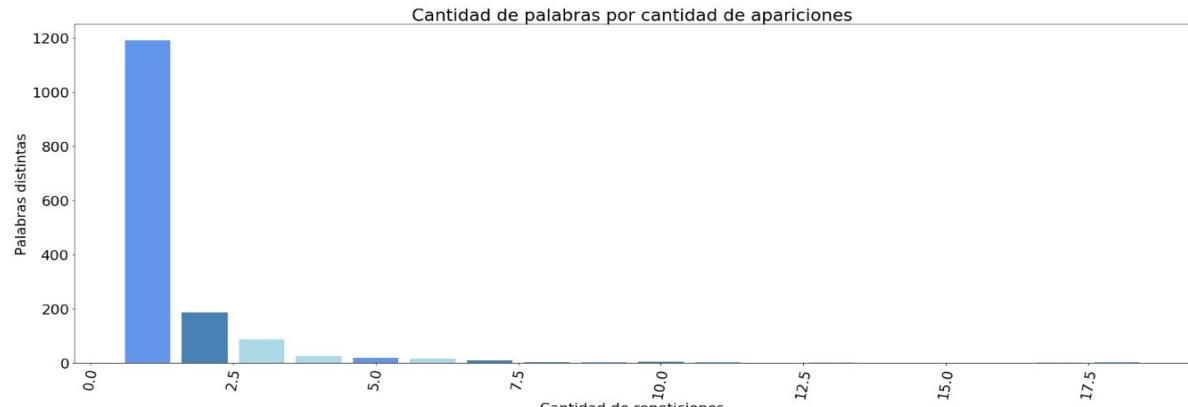
- A) De los textos de libros, en la carpeta “**Textos_Stopwords**”, encontrar todas las palabras diferentes y eliminar aquellas que la librería NLTK maneje como *stopwords*. Quitar signos de puntuación y aplicar el método *lower()* para pasar todo a minúsculas.
- B) Con las palabras únicas, realizar una gráfica de barras (Histograma) mostrando cuántas palabras existen que aparezcan solo una vez, cuantas aparecen 2 veces, cuantas 3, etc...
- C) Imprimir cuales son las palabras correspondientes a cada frecuencia y analizar los resultados para determinar hasta qué grado las palabras son relevantes
- D) Añadir StopWords personalizadas por el usuario y repetir el proceso de graficado. Observar diferencias

Palabras que se repiten 5 veces:

```
{'cualquier', 'lado', 'mañana', 'luego',  
o', 'segura', 'sabía', 'hacía', 'piel',
```

Palabras que se repiten 6 veces:

```
{'casi', 'abuela', 'bien', 'momento', 'ai  
ía', 'fuego'}
```



Tips:

- De la librería *nltk.corpus*, importar *stopwords*, ver documentación de NLTK en: <https://www.nltk.org/>
- Crear un diccionario en el que se almacenen todas las palabras diferentes contenidas en los textos, con base en su frecuencia de aparición.

Solución (Parte 1):

```
1 # Importar librerías de NLTK
2 from nltk.tokenize import RegexpTokenizer
3 from nltk.tokenize.treebank import TreebankWordDetokenizer
4 from nltk.corpus import stopwords
```

Uso de la librería NLTK para invocar un diccionario de Stopwords en Español predefinido

```
1 # Asignación de StopWords predefinidas para idioma Español
2 import nltk
3 stop_words = nltk.corpus.stopwords.words('spanish')
4 print(stop_words)
```

Aprendizaje Automático y Características



La manzana y el Aprendizaje automático

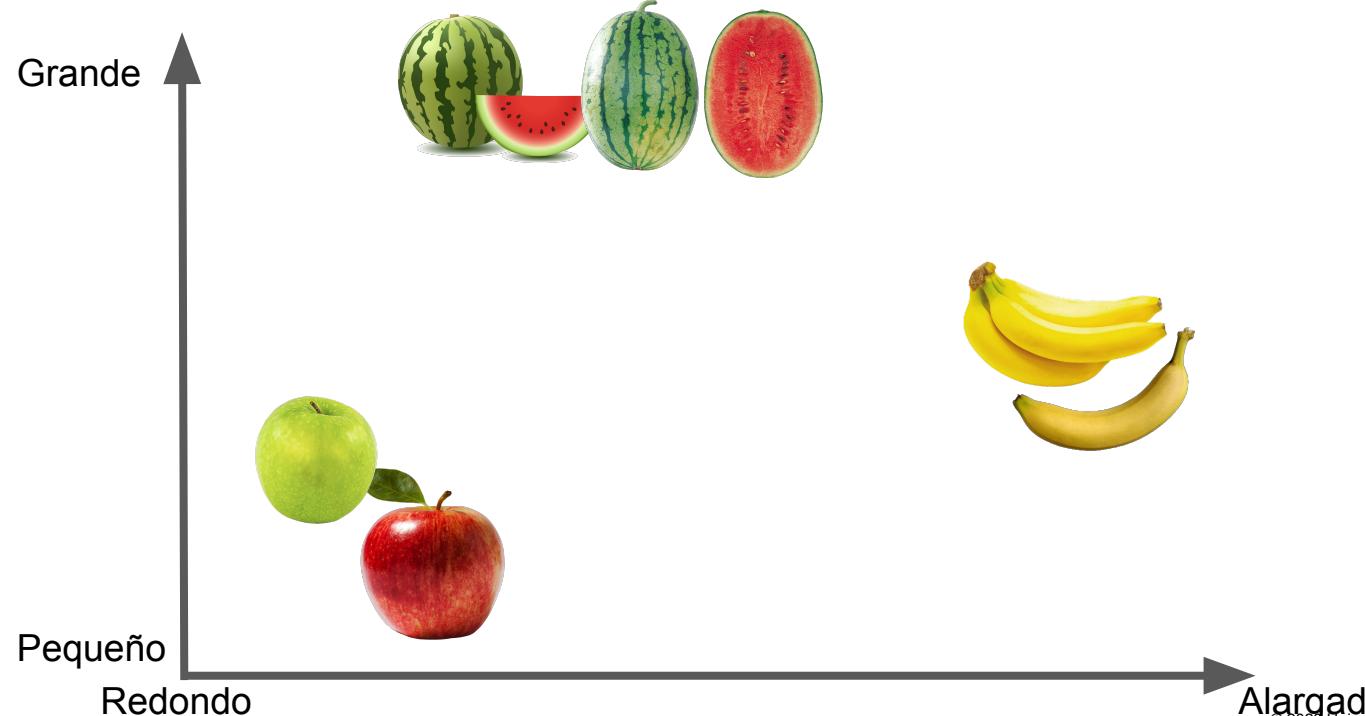
Tú como humano, ¿Cómo sabes que esto es una manzana? Eso es lo que le debes de **ENSEÑAR** a una máquina.



¿Qué hace que esto sea una manzana?

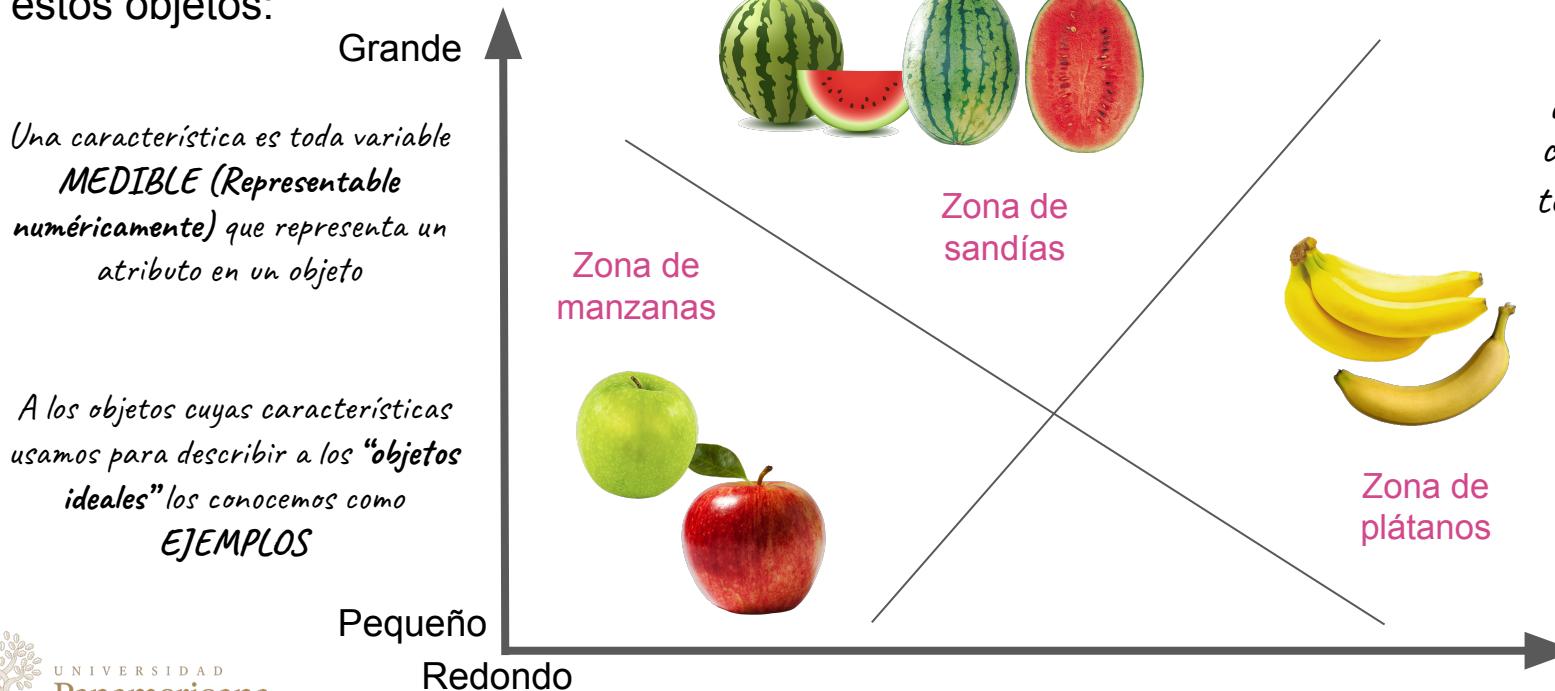
Características

Las características son los **atributos** físicos, abstractos, relacionales, etc., que nos permiten **catalogar o diferenciar objetos** claramente



Características

Podemos tener N características para describir objetos como forma, color, tamaño, redondez, cantidad de palabras en un texto, cantidad de caracteres, frecuencias de audio, entre un sin fin más. Gracias a las características podemos generar fronteras que nos ayuden a **CLASIFICAR** estos objetos:



¿Qué ejemplos de características se te ocurren para... ?

- Imágenes
- Textos
- Audio

¿Cómo funciona el Aprendizaje Automático?

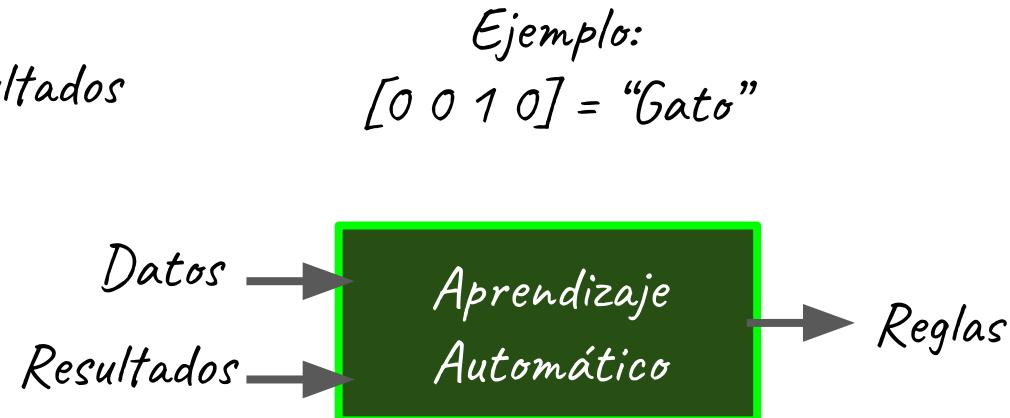
Es una **metodología de programación** que, a diferencia de la tradicional, adapta un **modelo** a un conjunto de entradas y sus respectivos **resultados deseados**.

Programación Tradicional:



Ejemplo:
 $D = A + B - C$

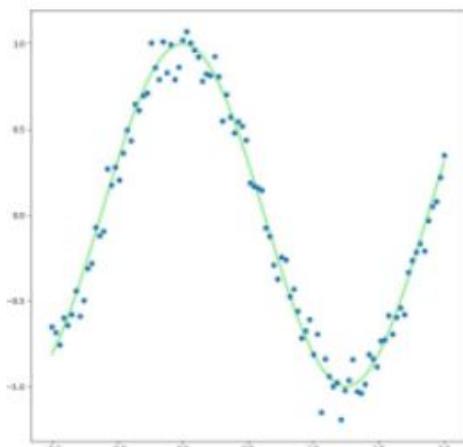
Aprendizaje Automático:



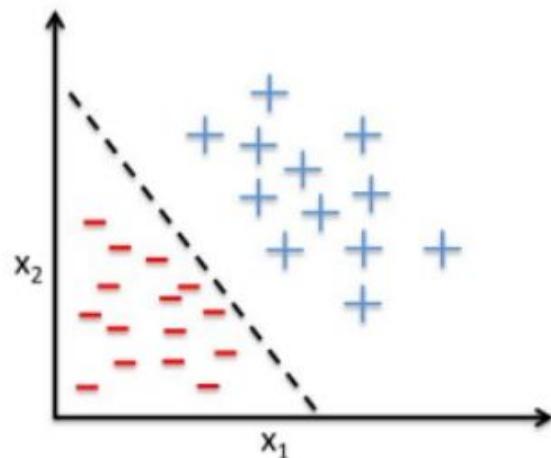
Ejemplo:
[0 0 1 0] = "Gato"

¿Cómo funcionan estos algoritmos?

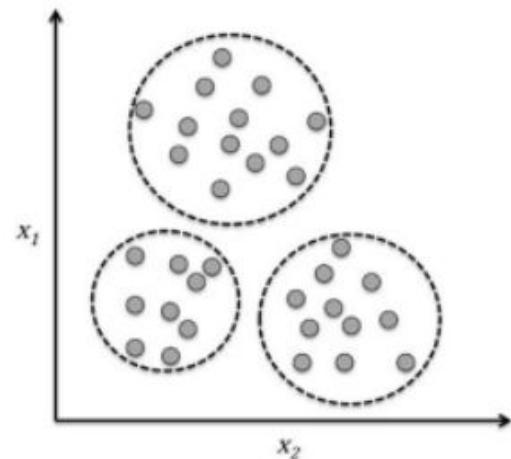
Regresión



Clasificación



Agrupamiento



Tomamos el conjunto de **entrenamiento** para tratar de generar una función que se ajuste a los datos (predecir un valor numérico)

Separa en clases, esto es cuando la variable de salida es una categoría, como como rojo o azul o enfermedad y sin enfermedad.

Es la división de los datos en grupos con **rastgos** similares (sin conocer las clases).



¿Qué es un sistema de PLN de aprendizaje automático?

La terminología o elementos de análisis de un sistema de PLN basado en elementos del lenguaje puede variar con respecto a un sistema basado en aprendizaje automático.

	Elementos del lenguaje	Aprendizaje automático
Elemento de análisis	Token	Característica
Algoritmos de análisis	<i>Basados en modelado del lenguaje</i>	<i>Machine Learning</i>
Dependencias	<i>Diccionarios de modelado</i>	<i>Algoritmos de ML</i>
Análisis gramatical	Si	<i>No tan común*</i>
Eficiente para textos largos	No	Si
Afecta idioma y ortografía	Si	<i>No*</i>

Características en textos

Las características en los textos pueden ser todos aquellos **parámetros medibles** dentro de un conjunto de textos, una de las características más comunes es la activación de palabras en un diccionario global como el que se muestra a continuación.

Frase1: *El dinero nunca duerme* (**El lobo de Wall Street**)

Frase2: *Veo gente muerta* (**Sexto sentido**)

Frase3: *Al infinito... ¡y más allá!* (**Toy Story**)

Frase 4: *Tonto es el que hace tonterías* (**Forrest Gump**)

NOTA: CUALQUIER cosa que se pueda medir en un texto puede ser considerada una característica

Tokens	el	dinero	nunca	duerme	veo	gente	muerta	al	infinito	...	i	y	más	allá	!	tonto	es	que	hace	tonterías
Frase1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Frase2	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
Frase3	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	
Frase4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	

Ejercicio

Texto	C1	C2	C3	C4	C5
<i>Algo malo debe tener el trabajo o los ricos ya lo habrían acaparado</i>					
<i>Yo amo, tú amas, él ama, nosotros amamos, ustedes aman, ellos aman. ¡Ojalá no fuese conjugación sino realidad!</i>					
<i>La primera obligación de todo ser humano es ser feliz, la segunda hacer feliz a los demás.</i>					
<i>El humor es cosa seria y la seriedad es una cosa que hay que tomar con humor</i>					
<i>Seres luminosos somos nosotros</i>					
<i>Debes desaprender lo aprendido</i>					
<i>Hazlo o no lo hagas, pero no lo intentes</i>					
<i>Entrénate para dejar ir todo lo que temes perder</i>					

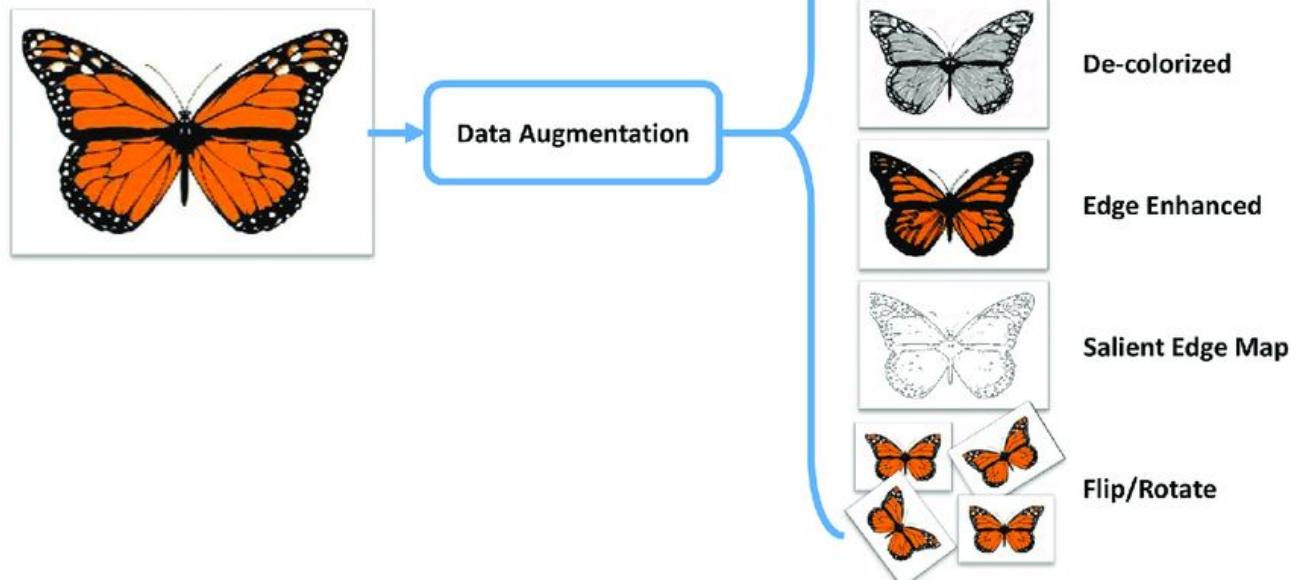
Si el idioma NO importa, ¿Puedo hablar marciano?



Basado en el uso de características, podrías trabajar con lenguajes mezclados, lenguajes inventados, e incluso en marciano

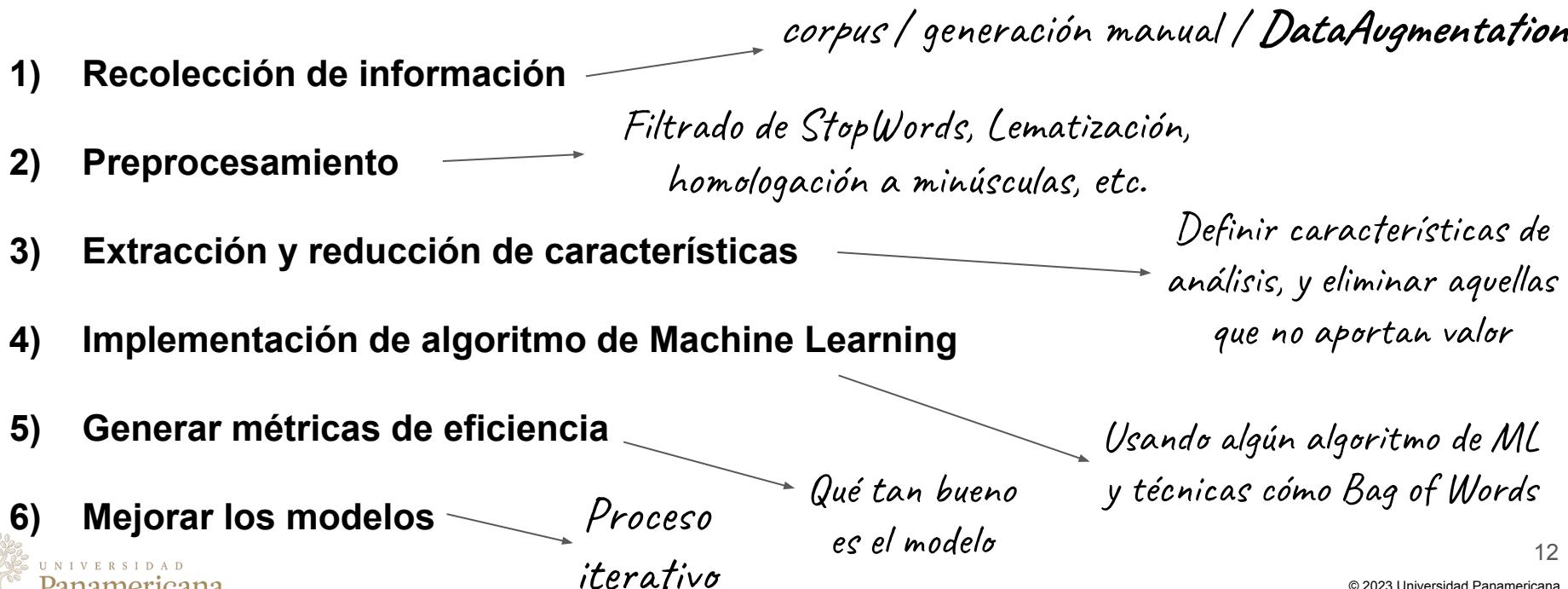
↳ VYSHIP hcfHLcP dT fHdec
yc hc VLcPf iHfc. 2c yc
hc M9PdT vtc9cLr hc
v5L5cPc ↳

Data Augmentation (Aumento de datos)



Pipeline de sistema de PLN y Aprendizaje Automático

El pipeline común para el desarrollo de modelos de PLN basado en Aprendizaje Automático suele incluir los siguientes pasos:



Obtener y generar información

En Inteligencia Artificial, a un conjunto de información clasificada y etiquetada, que se usará para entrenar un sistema, se le conoce como **corpus**.

El **corpus** puede obtenerse de información ya existente (Datos reales recopilados) o generada de manera artificial.

¿Y qué ocurre si no tengo información suficiente para comenzar a trabajar?

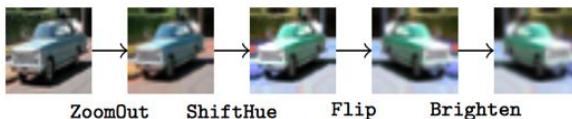
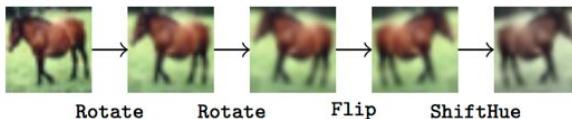
La respuesta es:
Data Augmentation



¿Cómo funciona el aumento de datos?

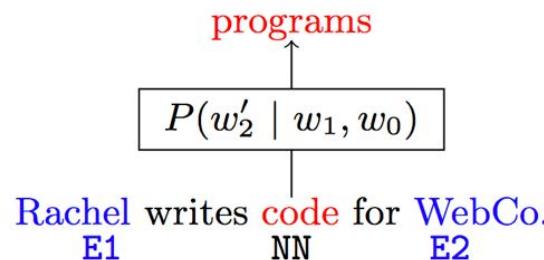
Consiste en tomar un **ejemplo base**, y aplicarle **transformaciones** para generar nuevos ejemplos con **variaciones significativas**.

Images



- Rotations
- Scaling / Zooms
- Brightness
- Color Shifts
- Etc...

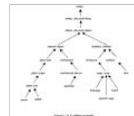
Text



NOTA:

Su uso en exceso puede conducir a problemas de reconocimiento.

Lo que conoceremos posteriormente como sobreentrenamiento



- Synonymy
- Positional Swaps
- Etc...

“Swap” posicional

Consiste en tomar ejemplos distintos (de la misma clase) e intercambiar palabras “equivalentes” en ambos.

Verbos de sus respectivas oraciones

El perro está sentado en el sillón

El gato está acostado sobre la escalera

El perro está **acostado** en la escalera

El gato está **sentado** sobre el sillón

NOTA:

En este tipo de algoritmos se pueden intercambiar, por ejemplo, verbos, preposiciones, sustantivos y otros elementos del discurso (*PoS*), o intercambiar palabras basadas en su posición.

Variaciones de escritura

Consiste en insertar / reemplazar / eliminar caracteres en palabras para generar variaciones, generalmente faltas de ortografía comunes, o sinónimos.

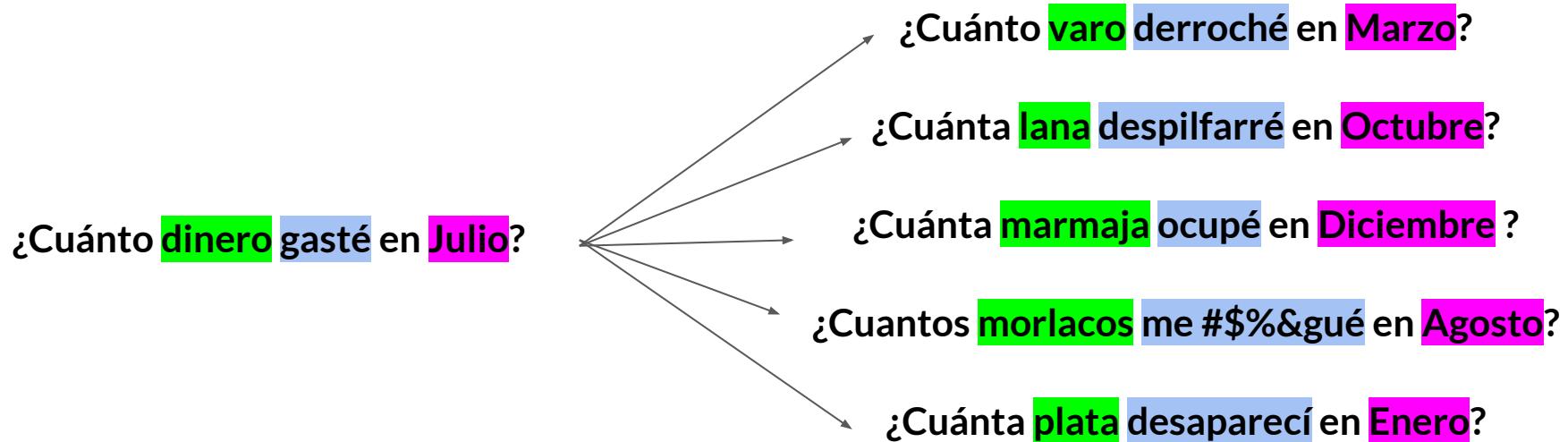


NOTA:

Esto resulta útil para chatbots en los que usualmente el usuario tiene faltas de ortografía

Diccionarios de campos semánticos

Consiste en definir diccionarios de palabras que pueden ser reemplazadas por otras dentro de un mismo campo semántico.



NOTA:

Esta técnica es especialmente útil para ChatBots, para el entrenamiento con diferentes entidades.

Fraseado (Uso de n-gramas)*

Consiste en unir grupos de 2 o más palabras (n-gramas) que, **adquieren mucho más valor cuando están juntas** (como si fuesen una sola palabra).

Contiene vitaminas y minerales



Contiene **vitaminas-y-minerales**

siempre juntos

¿Tienen servicio de caja rápida?



¿Tienen servicio de **caja-rápida**?

No quiero realizar contratar nada



No-quiero contratar nada

Producto específico

NOTA:

El proceso de Data Augmentation con n-gramas solo es útil si la aplicación podrá trabajar con los n-gramas durante la ejecución.*

Preprocesamiento de Datos



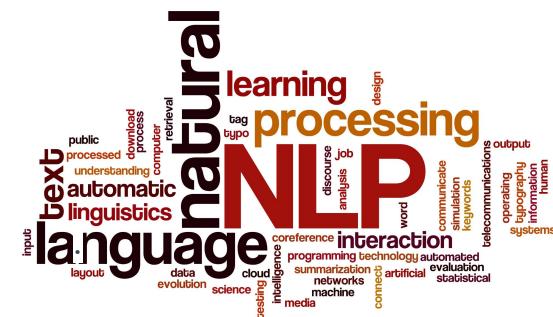
¿Qué es el preprocessamiento de datos?

Tratar la información, para que ésta sea **más útil para el problema específico a resolver**. (Quitar palabras poco útiles, resaltar palabras importantes, lematizar, etc.)

Para el PLN algunas de las técnicas de preprocessamiento más comunes son:

- **Lematización** - Convertir las palabras a su forma raíz
- **Filtrado de StopWords (Palabras vacías)** - Quitar palabras que generen ruido
- **Tratamiento de signos** - Eliminar signos de puntuación
- **Eliminar mayúsculas** - Convertir todo el texto a minúsculas

"Algunas palabras suelen ser más importantes que otras"



Lematización como técnica de preprocesamiento

La lematización como técnica de preprocesamiento es bastante útil para reconocer **Acciones**, sin importar la persona, el tiempo, o la conjugación de los verbos, adjetivos o sustantivos usados.

Por ejemplo:

Si se desean clasificar textos que hablen de estudiar, contra textos que hablen de trabajar:

Trabajar

- Trabajar puede ser estresante para la mayoría de las personas...
- Yo trabajo todos los días, y mi rutina consiste en...
- Actualmente estoy trabajando en una oficina, y mi opinión es...

- En mi opinión, es más sencillo estudiar, todos los días...
- Por más que estudio, no logro aprobar mi examen...
- Solo estudiando lograrás cumplir todas tus metas...

Estudiar

StopWords

Por definición, siempre son StopWords:

- **Artículos definidos** (El, La, Los, Las)
- **Artículos indefinidos** (Un, Una, Unos, Unas)
- **Posesivos** (Mi, Tu, Su, Nuestro, Nuestra)
- **Preposiciones** (a, con, de, en, para, por, etc)
- **Pronombres demostrativos** (este, esa, aquel, etc)
- **Palabras muy repetidas dentro de un grupo**

“Son las decisiones las que nos hacen ser quienes somos, y siempre podemos optar por hacer lo correcto” (Spiderman 3).

“Decisiones nos hacen ser quienes somos, siempre podemos hacer correcto” (Spiderman 3).

Reducción de características

Algunas de las características NO aportan valor al análisis dado que **se repiten mucho** en todo el conjunto de Frases / documentos a analizar, por ello, se suelen **eliminar** las características que no aporten valor.

Frase1: Los **animales** terrestres suelen ser mamíferos.

Frase2: Aquellos **animales** catalogados como terrestres, suelen tener un mayor tamaño.

Frase3: Aquellos **animales** que pueden volar, son generalmente aves.

Frase4: La capacidad de volar solo la contienen los **animales** del tipo “aéreo”

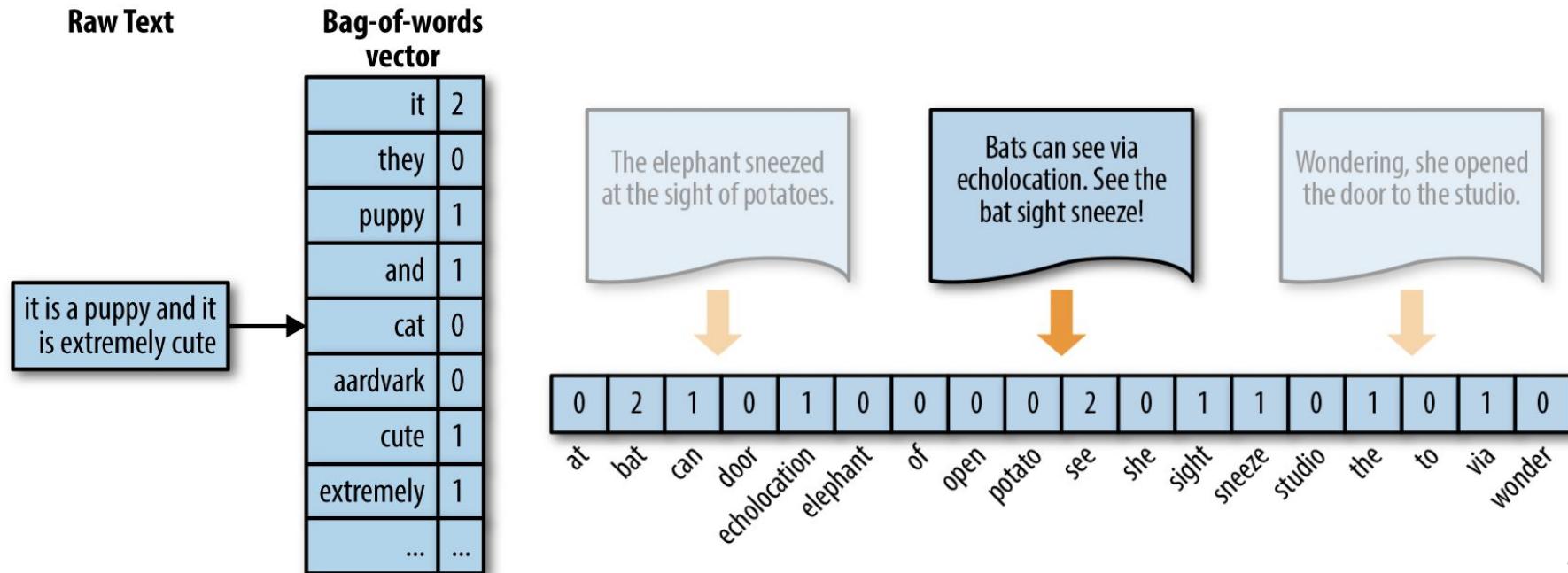
La característica “animales” es común para todas las frases, a diferencia de las características “terrestres” y “volar”, las cuales se presentan sólo en su propia categoría.

Bag of Words y tf-idf

	the	red	dog	cat	eats	food
1. the red dog →	1	1	1	0	0	0
2. cat eats dog →	0	0	1	1	1	0
3. dog eats food →	0	0	1	0	1	1
4. red cat eats →	0	1	0	1	1	0

Algoritmos Bag of Words (Bolsa de palabras)

Los algoritmos BoW (Bag of Words) tienen como base la **extracción de características**. Su manera de operar es considerando un diccionario con las diferentes características en el modelo, y después aplicando alguna técnica de Machine Learning para procesar la información.



Term Frequency - Inverse Document Frequency

Term Frequency: Es la cantidad de veces que un token (*term*) se encuentra en un texto / frase / documento (*document*).

Inverse Document Frequency: Se calcula como el logaritmo del inverso de la frecuencia de documentos (*document*) que contienen un token (*term*). Se obtiene dividiendo el total de los documentos entre el número de documentos en el que aparece un *term*, y luego, calculando el logaritmo del resultado

$$\text{tf-idf}(t,d) = \text{tf}(t,d) \times \text{idf}(t)$$

tf(t,d): *Term frequency* del término **t** en el documento **d**

idf(t): *Inverse document frequency* del término **t**

n: Cantidad de documentos **d**

df(t): Frecuencia de aparición de **t** en los documentos **d**

$$\text{idf}(t) = \log \frac{n}{\text{df}(t)} + 1$$

[Información del algoritmo utilizando sklearn](#)

Ejemplo: Para clasificador de Saludos (Según la hora)

n = 3

D-Día: Hola muy buenos día

df(Hola) = 3

D-Tar: Hola muy buenas tardes

df(buenas) = 2

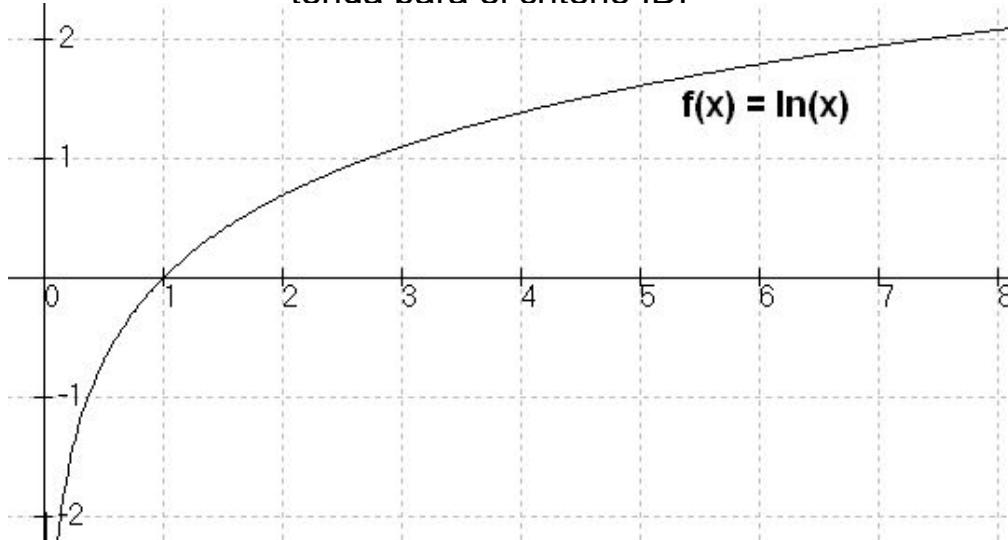
D-Noc: Hola muy buenas noches

Ejercicio, calcular el IDF para “Hola” y para “buenas”

$$idf_{hola} = \ln\left(\frac{3}{3}\right) + 1 = \ln(1) + 1 = 0 + 1 = 1$$

$$idf_{buenas} = \ln\left(\frac{3}{2}\right) + 1 = \ln(1.5) + 1 = 0.4054 + 1 = 1.4054$$

Mientras la fracción dentro de un logaritmo es más pequeña, su valor absoluto es más grande, por lo cual, mientras menor sea la frecuencia de aparición de una palabra, mayor será el peso que tendrá para el criterio IDF



Pesos-IDF

hola	1.000000
muy	1.000000
buenas	1.405465
buenos	2.098612
días	2.098612
noches	2.098612
tardes	2.098612

Token muy relevante

Así podemos **conocer qué tan relevante es cada palabra para el conjunto de textos**, y así, se pueden descartar aquellas con pesos muy bajos en relación a las demás.

Si multiplicamos el ***idf*** por el ***tf*** de cada token por cada texto se obtendrá la relevancia de ese token en ese texto (Será de cero si el token en cuestión no aparece en el texto, ya que el ***tf*** será también de cero)

Así se puede determinar que palabras son las más representativas o importantes de cada texto (Las que deberían definir la clase de ese texto). Si un token tiene un *idf* bajo pero un ***tf*** muy alto, se volverá relevante.

	TfIdf 1	TfIdf 2	TfIdf 3
buenos	0.638341	0.000000	0.000000
días	0.638341	0.000000	0.000000
hola	0.304173	0.345454	0.345454
muy	0.304173	0.345454	0.345454
tardes	0.000000	0.724975	0.000000
buenas	0.000000	0.485524	0.485524
noches	0.000000	0.000000	0.724975

Tokens más relevantes para cada frase

Tokens que definen a las clases

[Día] buenos, días

[Tar] tardes

[Noc] noches

Nota, el valor final de tf - idf se calcula utilizando el **logaritmo natural** para la versión de **Sklearn**, esta librería también aplica la **norma Euclíadiana** para escalar todos los resultados entre **0 y 1** como se ve en el siguiente cálculo de ejemplo para el término “buenas” en el **texto 2**.

La librería Sklearn, utiliza logaritmo natural, aunque otras versiones de tf-idf utilizan log base 10.

$$t (\text{término}) = 'buenas'$$

$$n (\text{documentos}) = 3$$

$$df (\text{documentos en que aparece } t) = 2$$

$$idf_{buenas} = \ln\left(\frac{n}{df}\right) + 1 = \ln\left(\frac{3}{2}\right) + 1 = 0.4054 + 1 = 1.4054$$

$$tf_{buenas} = \text{Veces que aparece } t \text{ en el documento} = 1$$

$$tf - idf_{buenas} = tf_{buenas} * idf_{buenas} = 1 * 1.4054 = 1.4054$$

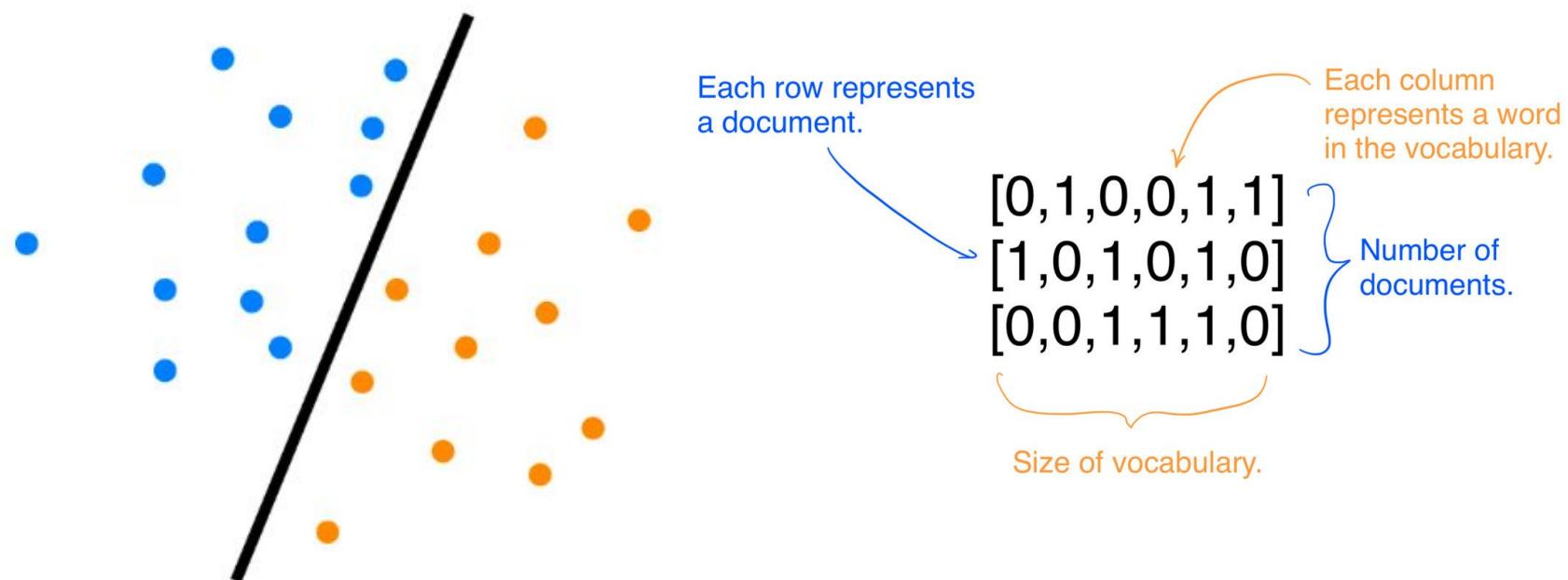
tf-idf puro VS tf-idf normalizado

$$tf - idf_{buenas} (\text{Escalado}) = \frac{v}{\sqrt{v_1^2 + v_2^2 + v_3^2 + \dots}} = \frac{1.4054}{\sqrt{1^2 + 1^2 + 1.4054^2 + 2.0986^2}} = \frac{1.4054}{\sqrt{8.3792}} = \frac{1.4054}{2.8946} = 0.4855$$

Norma Euclíadiana



Desarrollo de un modelo de clasificación Binaria con BoW



¿Qué es un clasificador *Binario*?

Un clasificador binario es un modelo capaz de diferenciar entre **SOLO 2 categorías** (Encendido y Apagado) por ejemplo, un clasificador Spamless o anti spam

Spam: Aquellos elementos que no resultan de interés (No solicitados)

- ¡Ofertón!, Solo por hoy, esto no se va a repetir...
- Mira este video, no creerás lo que pasó...
- Fulano de tal vio tu perfil en Facebook, dile hola...
- etc...



Ham: Elementos relevantes (Solicitados, o enviados de manera PERSONALIZADA, No genérica)

- Se ha actualizado la contraseña de tu cuenta...
- Información solicitada sobre el esquema de pagos...
- Agenda para la reunión de mañana...
- etc...



Paso 1 - Obtener un *corpus* o conjunto de información

Un DataSet o ***corpus***, consiste en una cantidad **N** de ejemplos (En este caso **Textos**) que sirven para **entrenar o alimentar** un modelo de IA:

Clasificación de textos

- Textos Ham: “Contraseña Recuperada...”
- Textos Spam: “Aprovecha este ofertón...”



Análisis de sentimientos

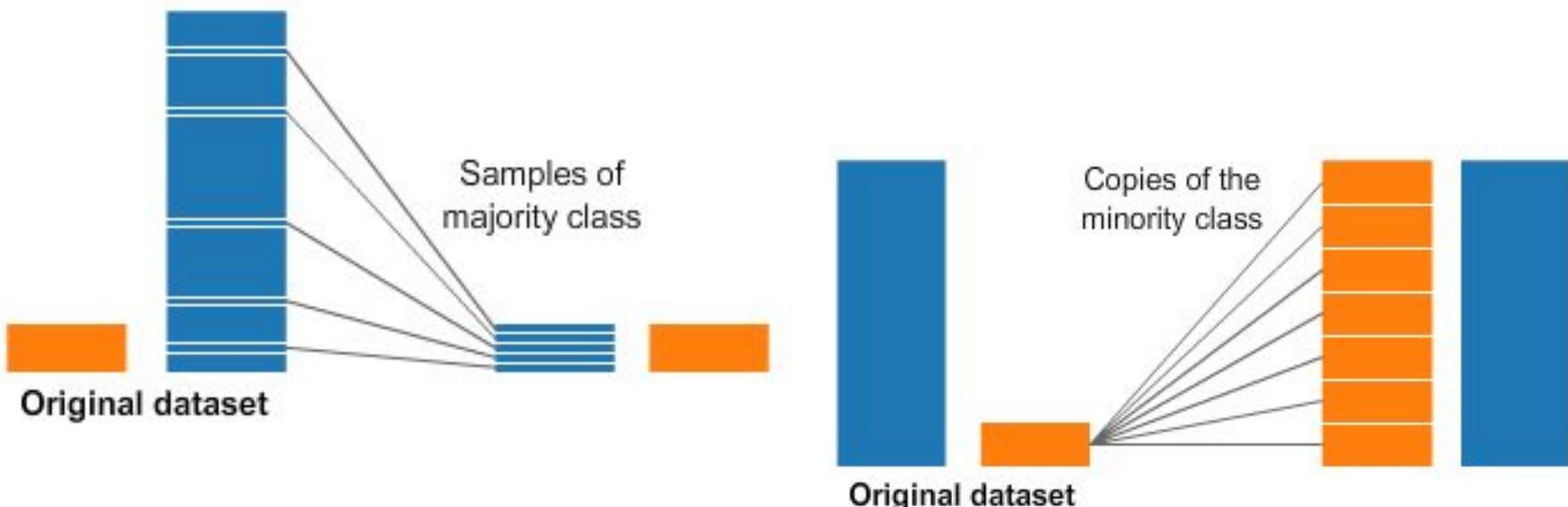
- Texto positivo: “La película fue excelente...”
- Texto negativo: “Su actuación fue pésima...”
- Texto neutro: “Buena obra, pero mal servicio...”



Paso 2 - Verificar el balance y la validez de los datos

Es importante verificar que no se tengan ejemplos **No válidos**, y que la información no esté demasiado **desbalanceada** (Dentro de los posibles)

Si están muy desproporcionados, **podemos Eliminar / Agregar** ejemplos para tener un mejor resultado.



Paso 3 - Preprocesamiento

Consiste en **mejorar** los ejemplos y **eliminar elementos** que no sean relevantes para el modelo de NLP.

NOTA: Preprocesar la información *en exceso* puede derivar en generalizaciones y clasificadores pobres.

“La semana pasada ví una obra de teatro, la cuál resultó ser excelente. La actuación fue buena, aunque, el lugar era algo pequeño y el servicio era lento.”

Positivo:

- excelente
- buena



Las mismas palabras pueden tener diferentes significados según el contexto

“La actuación no fue buena, y está muy lejos de ser excelente. Solo me gustó porque el lugar no es precisamente pequeño, y el servicio para nada es lento”

Negativo:

- pequeño
- lento



Paso 4 - Feature engineering

Es muy bueno **analizar previamente** la información para encontrar relaciones, patrones, información poco útil, entre otros, por ejemplo:

Cantidad de caracteres

- Palabras más usadas
- Cantidad de signos
- PoS específicas
- Entidades
- Etc...

Escritura correcta:

- *Hola, buenas tardes. Quisiera realizar una pregunta.
¿Pueden realizar mi entrega el dia de mañana?*

Signos

Escritura incorrecta:

- *ola buenas tardes. pueden hacer mi entrega mañana?*

Describir un animal:

- *El cocodrilo habita en algunas zonas pantanosas del Amazonas, es un animal agresivo y cauteloso que...*

Entidades

Describir un lugar:

- *México tiene actualmente más de 7,000 millones de habitantes, su extensión es de más de 40 mil kilómetros...*

Respuesta cálida:

Si por favor, muchas gracias (24)
No por ahora, te lo agradezco (24)
Me parece perfecto :) (18)

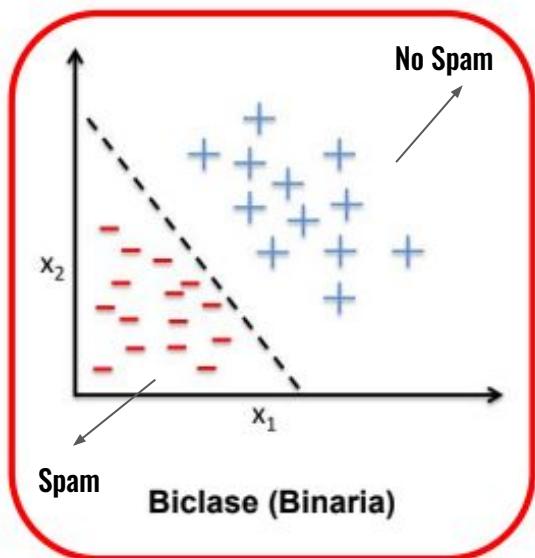
Respuesta fría:

Si (2)
No (2)
Ajá (3)

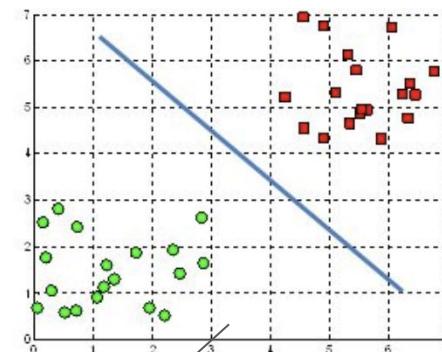
Caracteres

Paso 5 - Construcción del modelo de clasificación

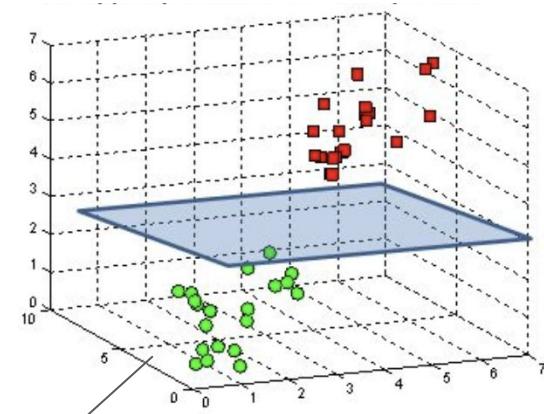
Para un algoritmo BoW (Bag of Words) se puede usar algún modelo de Machine Learning que permita clasificar características, com un SVC (Clasificador de soporte vectorial), LR (Regresión Logística), DT (Árbol de decisiones) entre otros.



Un clasificador binario, puede tener N características de análisis aunque solo sean dos clases a separar



Modelo de 2 características

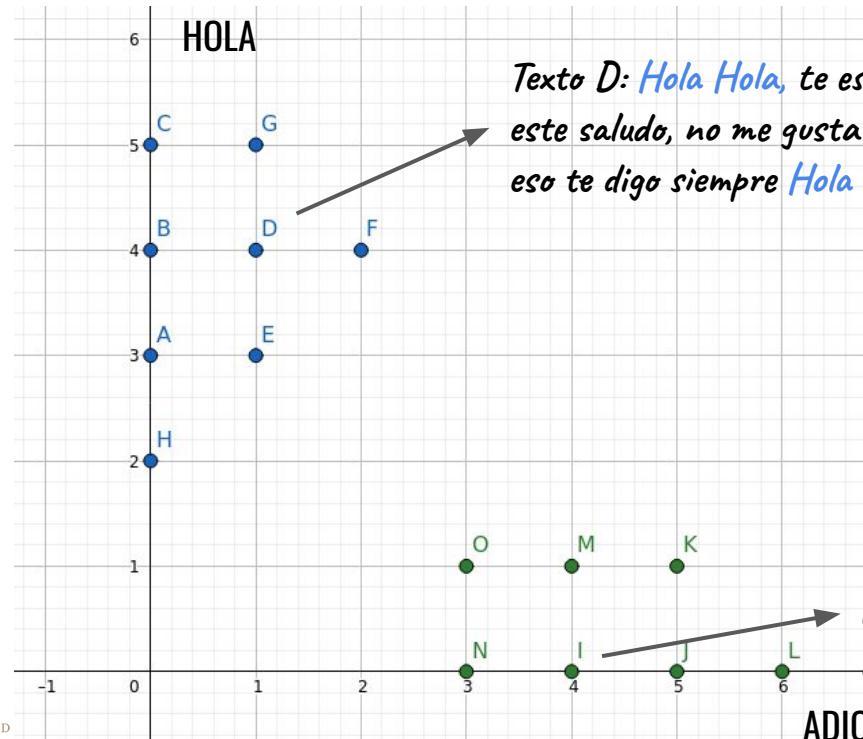


Modelo de 3 características

NOTA: Cada característica diferente aumentará la dimensionalidad del modelo

Ejemplo de modelado basado en características:

Modelo para clasificar “Textos de saludos” VS “Textos de despedidas” basados en 2 características, la frecuencia de la palabra **hola**, y de la palabra **adios**:

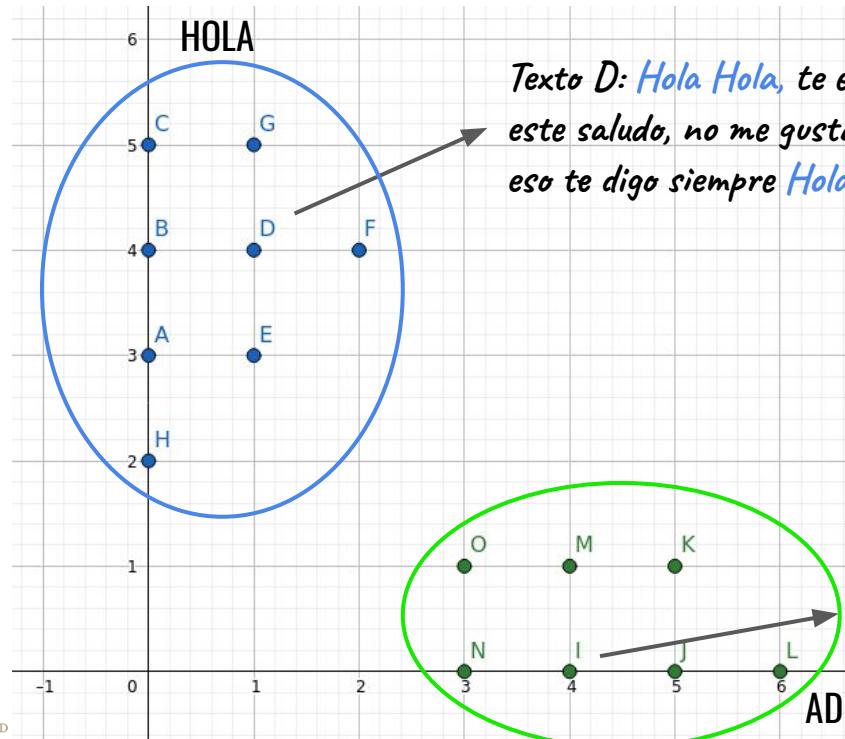


Texto D: *Hola Hola, te estoy mandando este saludo, no me gusta decir adiós, por eso te digo siempre Hola Hola.*

Texto I: *Hoy tengo que decir adiós, y despedirme con esa palabra. Adiós no es para siempre, adiós, a veces es solo por un instante, adiós.*

Ejemplo de modelado basado en características :

Identificamos 2 grupos de puntos para los dos tipos de textos. Cada uno representa una región donde debería de caer un tipo de texto en específico:

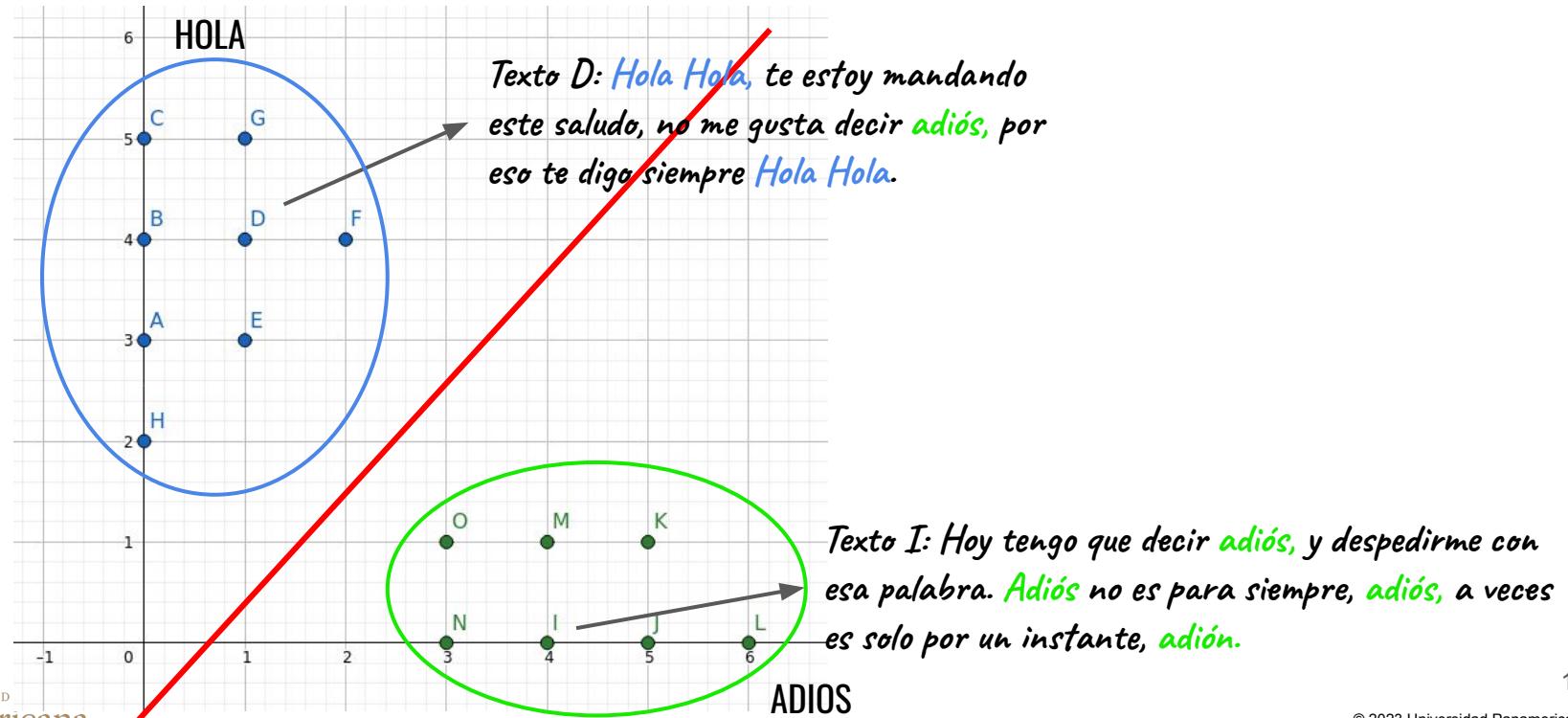


Texto D: *Hola Hola, te estoy mandando este saludo, no me gusta decir adiós, por eso te digo siempre Hola Hola.*

Texto I: *Hoy tengo que decir adiós, y despedirme con esa palabra. Adiós no es para siempre, adiós, a veces es solo por un instante, adión.*

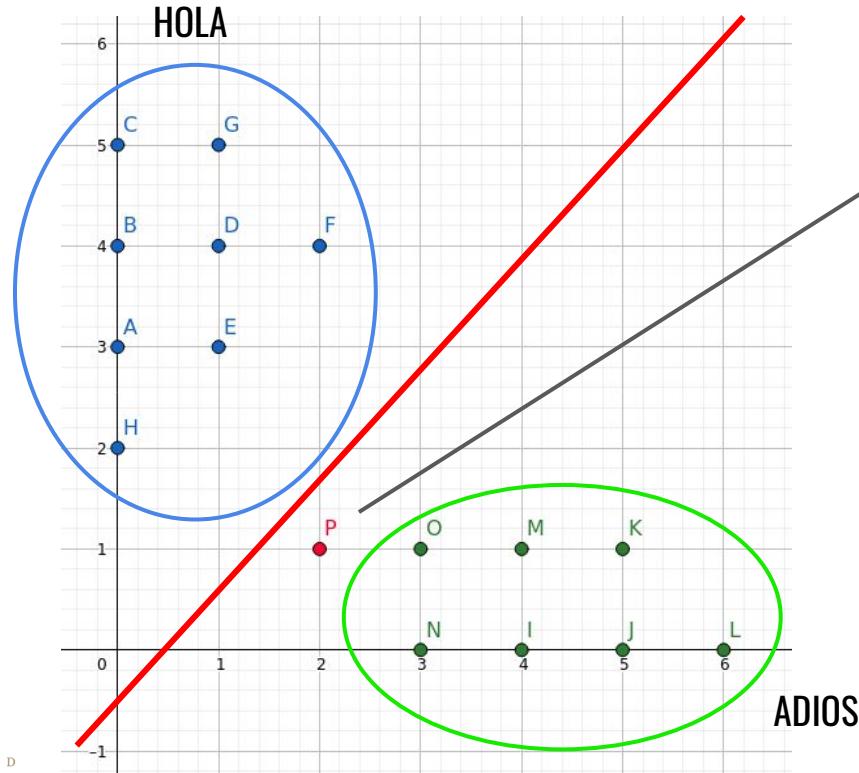
Ejemplo de modelado basado en características :

El objetivo de un modelo de ML es **encontrar relación en la información** y así poder separarla. Por ejemplo, un SVC intentaría encontrar una línea que separe la info en dos categorías:



Ejemplo de modelado basado en características :

Una vez que se genere esa línea conocida como **hiperplano**, cualquier ejemplo nuevo puede ser clasificado usando dicha línea como base:

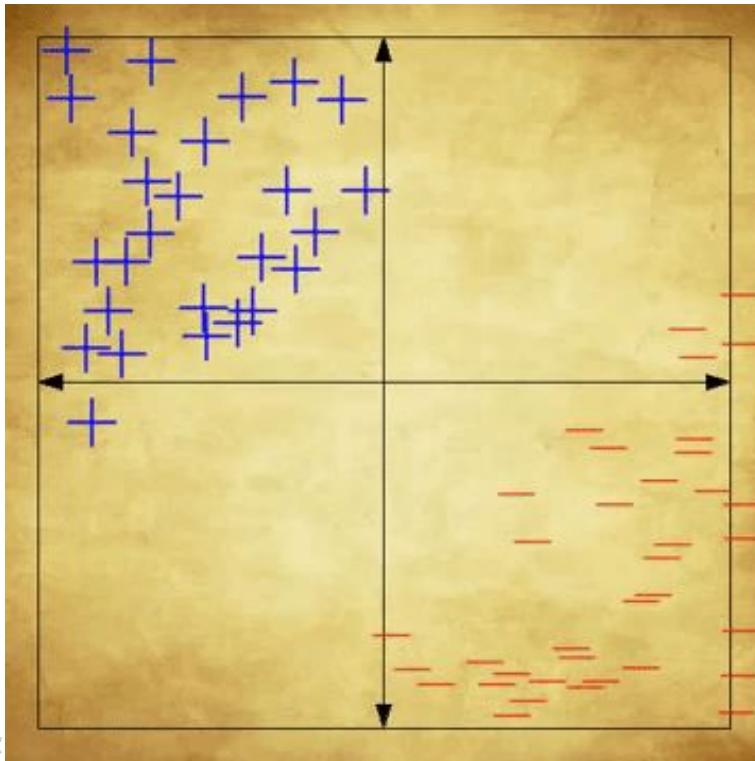


Texto P: *Adiós, esta es una despedida, fue grandioso cuando te dije **hola** por primera vez, pero el día de hoy es el **adiós** que necesitamos para ser felices.*

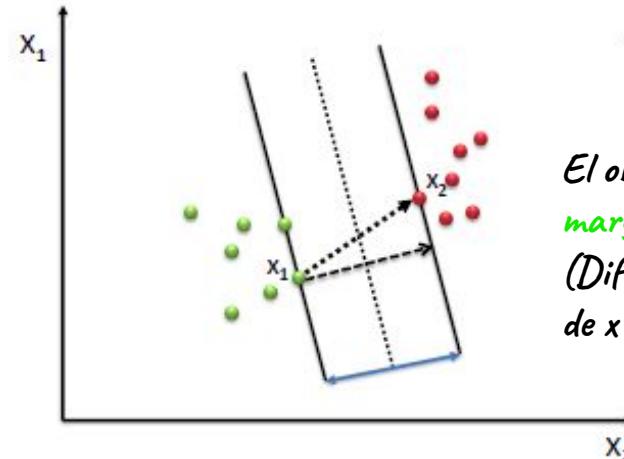
Si un nuevo texto “P” cae en la región de Despedidas, por sus características [Adiós (2 Veces) y Hola (1 Vez)] pertenece a esa categoría

Máquina de soporte vectorial (SVC)

Es un algoritmo de ML que **busca obtener el mejor hiperplano** de separación entre las categorías, basándose en un margen máximo de separación entre ambas categorías.



El algoritmo es un proceso con iteraciones llamadas **épocas**, en las cuales se corrige o modifica la inclinación del hiperplano según el **error** del mismo.

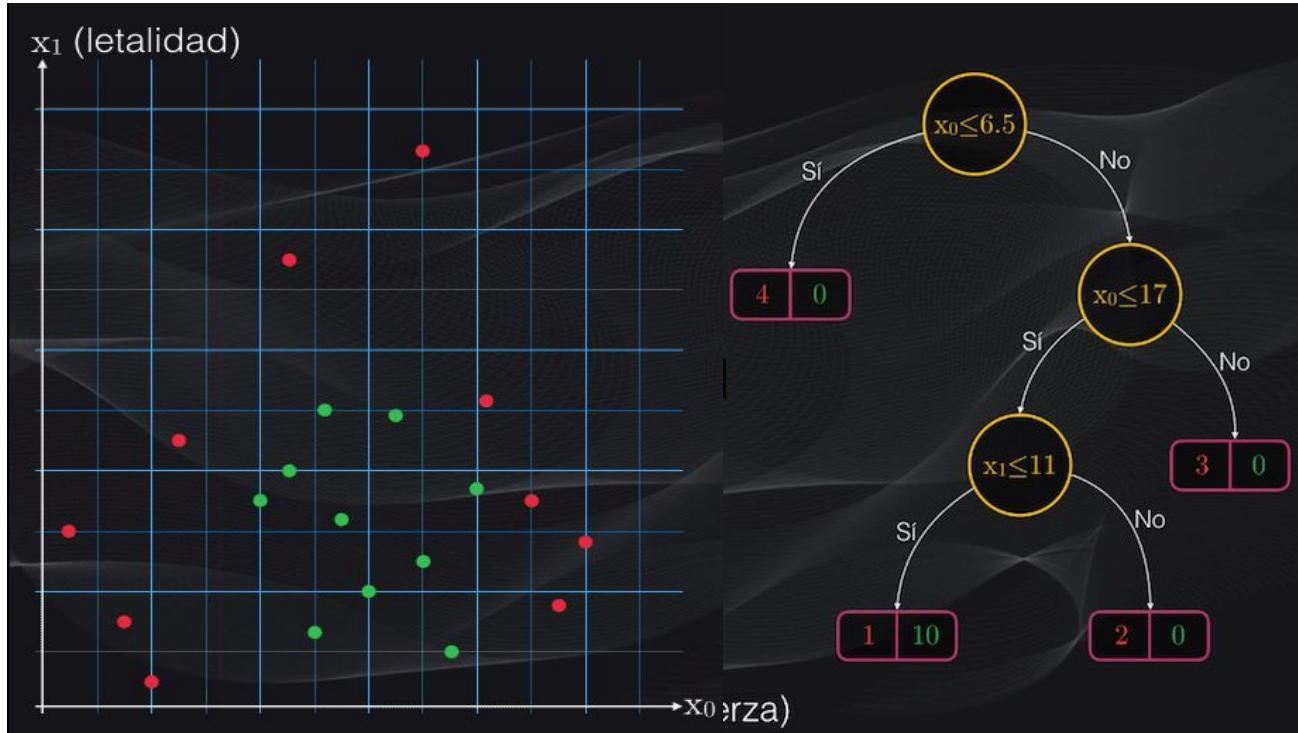


$$\frac{w}{\|w\|} \cdot (x_2 - x_1) = \text{width} = \frac{2}{\|w\|}$$

El objetivo es **maximizar el margen**, al **minimizar w**
(Diferencia entre las distancias de x_1 al margen y a x_2)

Árboles de decisión (DT)

Es un algoritmo de ML que **busca dividir el espacio en regiones perpendiculares a las características**. Estos algoritmos toman decisiones secuenciales para generar regiones **anidadas**.

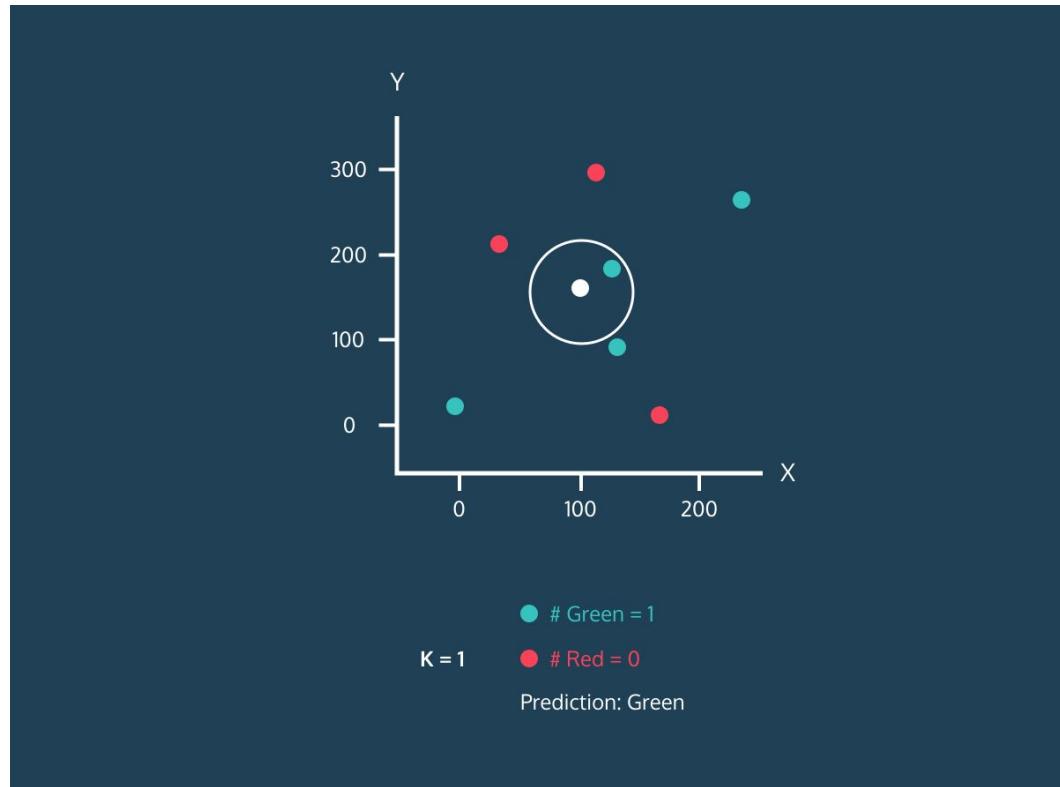


Estos algoritmos miden su error basados en la cantidad de elementos no clasificados o mal clasificados después de cada decisión

Vecinos mas cercanos (KNN)

Es un algoritmo que toma su decisión basado en la cercanía de un nuevo objeto a los **K ejemplos mas cercanos** pertenecientes a una clase u otra.

En este tipo de algoritmos se elige el valor de K, y dependiendo de la cantidad de ejemplos pertenecientes a cada categoría, se determina una decisión o la otra.



Naive Bayes

Algoritmo basado en probabilidades condicionales

$$R = P(\text{Llueve} \mid \text{Nublado}, \text{H_Alta}, \text{T_Baja})$$

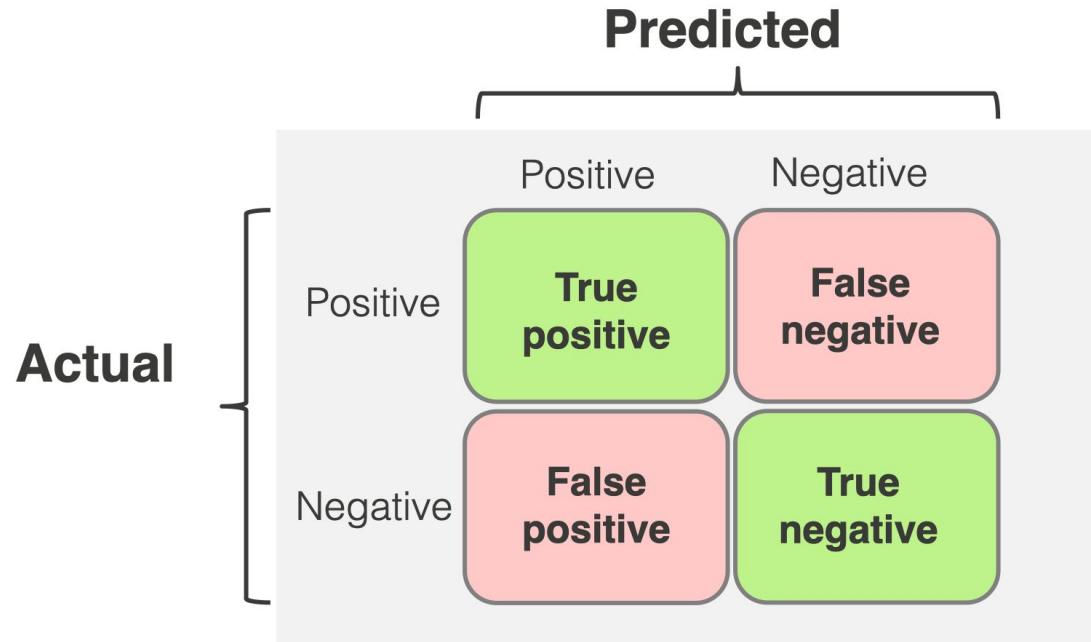
$$\text{NR} = P(\text{NoLlueve} \mid \text{Nublado}, \text{H_Alta}, \text{T_Baja})$$

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}$$

THE PROBABILITY OF "B" BEING TRUE GIVEN THAT "A" IS TRUE
↓
↑ THE PROBABILITY OF "A" BEING TRUE GIVEN THAT "B" IS TRUE

THE PROBABILITY OF "A" BEING TRUE
↓
↑ THE PROBABILITY OF "B" BEING TRUE

Paso 6.- Revisión de modelo



Matriz de confusión

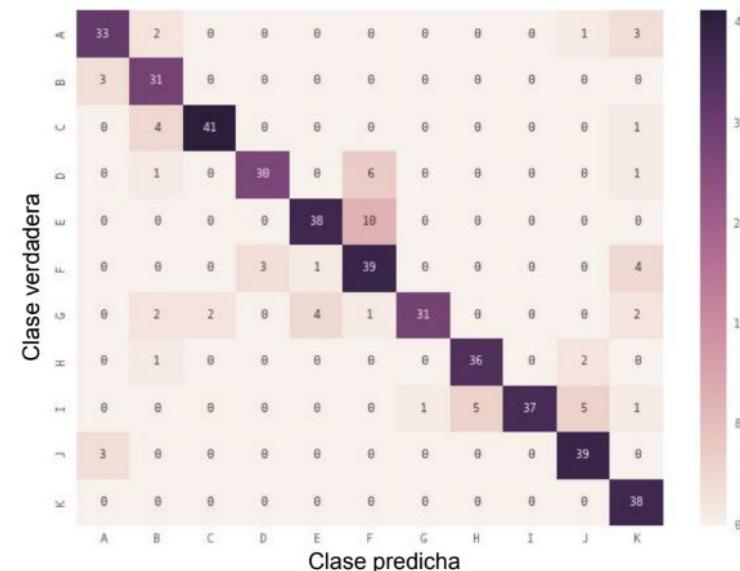
Una matriz de confusión es una matriz de NxN, donde N es el **número de clases** del modelo.

Expresa la **cantidad de veces que una clase A fue clasificada como una clase B y viceversa**, también nos indica la relación de falsos positivos y falsos negativos

¿Qué es **peor**, un falso positivo o un falso negativo?

		Datos predichos	
		Rechazar	Aceptar
Datos reales	Rechazar	No error	Falso positivo
	Aceptar	Falso negativo	No error

Biclase



Multiclas

Métricas obtenidas de la matriz de confusión

Precisión y sensibilidad

- **Precisión (PPV):** Proporción de las predicciones positivas que son correctas
- **Sensibilidad (Recall o TPR):** Proporción de las predicciones correctas para la clase positiva

		Dato predicho	
		Rechazar	Aceptar
Dato real	Rechazar	TN	FP
	Aceptar	FN	TP

$$\text{Sensibilidad} = \frac{TP}{TP + FN}$$

$$\text{Precisión} = \frac{TP}{TP + FP}$$

TPR: True Positive Rate
NPV: Negative Predictive Value
PPV: Positive Predictive Value
TN: True Negative
TP: True Positive
FP: False Positive
FN: False Negative

Precisión y sensibilidad

- F_1 : Es conveniente combinar precisión y recall en una métrica llamada F1-score (Como resultado, el clasificador solo obtendrá un puntaje alto si tanto el recall como la precisión son altos)

		Dato predicho	
		Rechazar	Aceptar
Dato real	Rechazar	TN	FP
	Aceptar	FN	TP

$$F_1 = \frac{2}{\frac{1}{pres} + \frac{1}{recall}} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

Ejercicio: Calcular la precisión, la sensibilidad y el F1-score para las siguientes matrices de confusión (Biclase), y de los parámetros en color azul

	C1	C2
C1	1	3
C2	2	8

TP = 1

$$\text{Precisión}_{C1} = \frac{TP_{C1}}{TP_{C1} + FP_{C1}} = \frac{1}{1+2} = \frac{1}{3} = 0.333\dots$$

FP = 2

$$\text{Sensibilidad}_{C1} = \frac{TP_{C1}}{TP_{C1} + FN_{C1}} = \frac{1}{1+3} = \frac{1}{4} = 0.25$$

FN = 3

$$F1_{C1} = \frac{2}{\frac{1}{pres_{C1}} + \frac{1}{sens_{C1}}} = \frac{2}{\frac{1}{3.333} + \frac{1}{0.24}} = \frac{2}{\frac{2}{7}} = 0.2857$$

TP = 9

$$\text{Precisión}_{C2} = \frac{TP_{C2}}{TP_{C2} + FP_{C2}} = \frac{9}{9+4} = \frac{9}{13} = 0.6923$$

FP = 4

$$\text{Sensibilidad}_{C2} = \frac{TP_{C2}}{TP_{C2} + FN_{C2}} = \frac{9}{9+0} = \frac{9}{9} = 1.0$$

FN = 0

$$F1_{C2} = \frac{2}{\frac{1}{pres_{C2}} + \frac{1}{sens_{C2}}} = \frac{2}{\frac{1}{0.6923} + \frac{1}{1}} = \frac{2}{\frac{2}{2.444\dots}} = \frac{2}{2.444\dots} = 0.818$$