

# DATA SCIENCE



---

Federico Baiocco  
[baioccofede@gmail.com](mailto:baioccofede@gmail.com)  
3512075440



# Clase 28 - Agenda

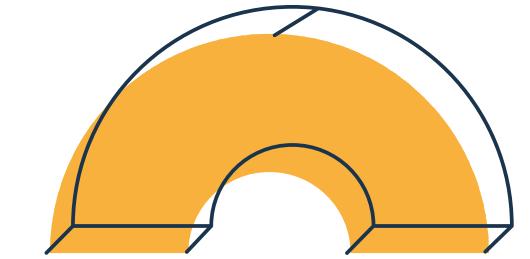
## NATURAL LANGUAGE PROCESSING

---



NLP

---



Hasta ahora venimos trabajando con datos "estructurados" como por ejemplo:

	ID	Label	House	Year	Month	Temperature	Daylight	EnergyProduction
0	0	0	1	2011	7	26.2	178.9	740
1	1	1	1	2011	8	25.8	169.7	731
2	2	2	1	2011	9	22.8	170.2	694
3	3	3	1	2011	10	16.4	169.1	688

Sin embargo, hay muchísimos datos disponibles que no vienen en este formato. Por ejemplo: Texto

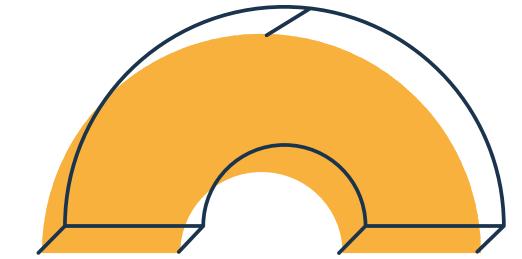


¿Dónde podemos encontrar texto?

- Noticias
- Comentarios
- Reviews de una película
- Libros
- Twitter
- Encuestas
- Emails

...

# NLP



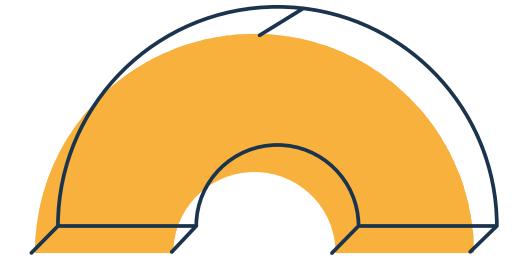
NLP = Natural language processing = Procesamiento del lenguaje natural.

Es una rama de la inteligencia artificial que se enfoca en permitirle a las computadoras entender y procesar lenguaje natural.

La idea es aplicar distintas técnicas de preprocesamiento y modelado para darle "estructura" a un texto.

Algunos ejemplos:

- Análisis de sentimiento (por ejemplo de tweets, comentarios, encuestas)
- Extracción de palabras clave de un texto
- Extracción de tópicos de un texto
- Clasificación de texto
- Chat bots



Al momento de trabajar con texto, es muy importante el preprocesamiento.

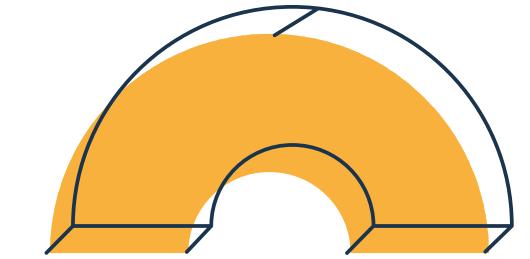
Imaginen los siguientes tweets:

- Quiero ir a COMER Hamburguesas a #Palermo
- Quisiera ir a comer una hamburguesa en palermo

Nosotros podemos darnos cuenta que ambas oraciones son similares, que por ejemplo "COMER" es equivalente a "comer" y "Hamburguesas" es el plural de "hamburguesa".

Para una computadora, no es tan simple.

# Normalización



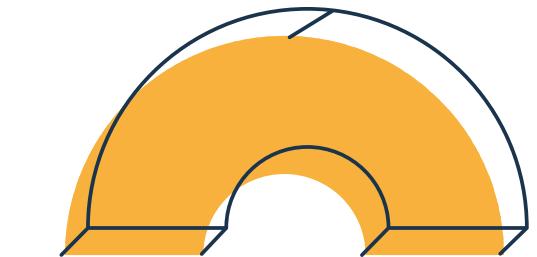
Entonces, un paso importante en el preprocesamiento de texto es la normalización.

Para esto, podemos aplicar:

- Llevar todo a minúsculas
- Limpiar caracteres especiales
- Llevar todas las palabras a su raíz (Hamburguesas = hamburguesa)

Además, en un texto vamos a encontrar muchas palabras como "el", "la", "y", etc. Estas palabras se conocen como "stop words" y suelen eliminarse del texto a analizar.

# Tokenización



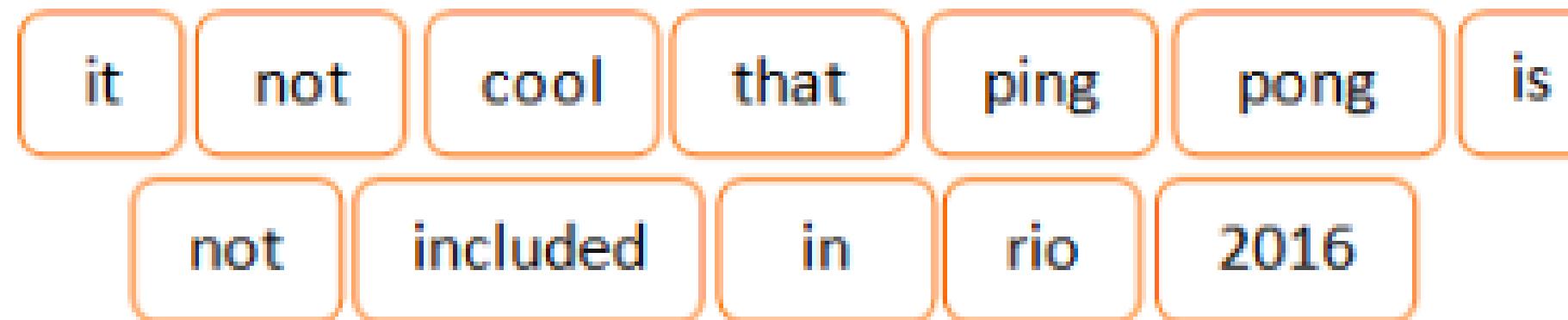
Tokenización es el proceso de separar el texto original en partes llamadas "tokens".

Los tokens pueden ser por ejemplo oraciones, palabras, conjuntos de 2 palabras, conjuntos de 3 palabras, etc.

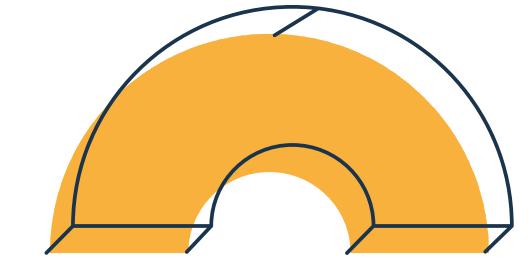
it not cool that ping pong is not included in rio 2016



Tokenization



# Vectorizar

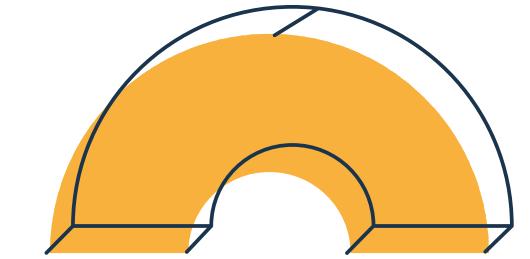


Cuando trabajábamos con datos estructurados, a nuestros modelos de machine learning los entrenábamos con vectores de features (que llamamos X).

Por ejemplo:

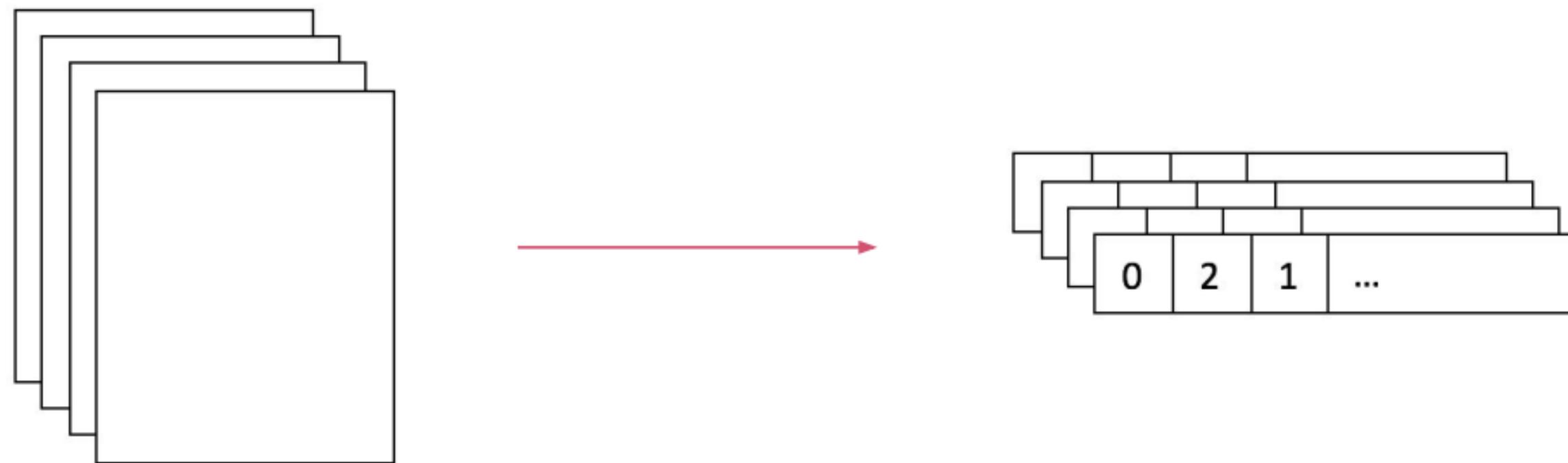
ID	Label	House	Year	Month	Temperature	Daylight	EnergyProduction
0	0	1	2011	7	26.2	178.9	740

# Vectorizar

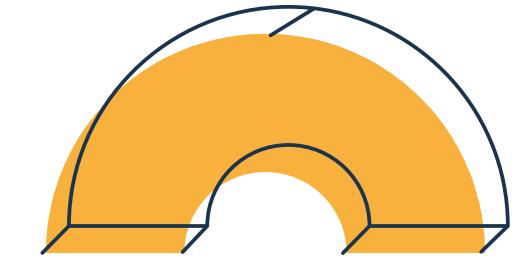


Ahora que tenemos texto, tenemos que buscar la manera de representarlo como un vector de features.

El objetivo de la vectorización, es representar a cada texto como un vector de features que nos sirva para entrenar un modelo de machine learning



# Vectorizar



Existen distintas técnicas para vectorizar. NLP es un área que está constantemente progresando (cada mes aparecen nuevas cosas).

Una de las técnicas más conocidas (y simples) es Bag of Words.

Para vectorizar utilizando Bag of Words, lo primero que debemos hacer es identificar nuestro "corpus".

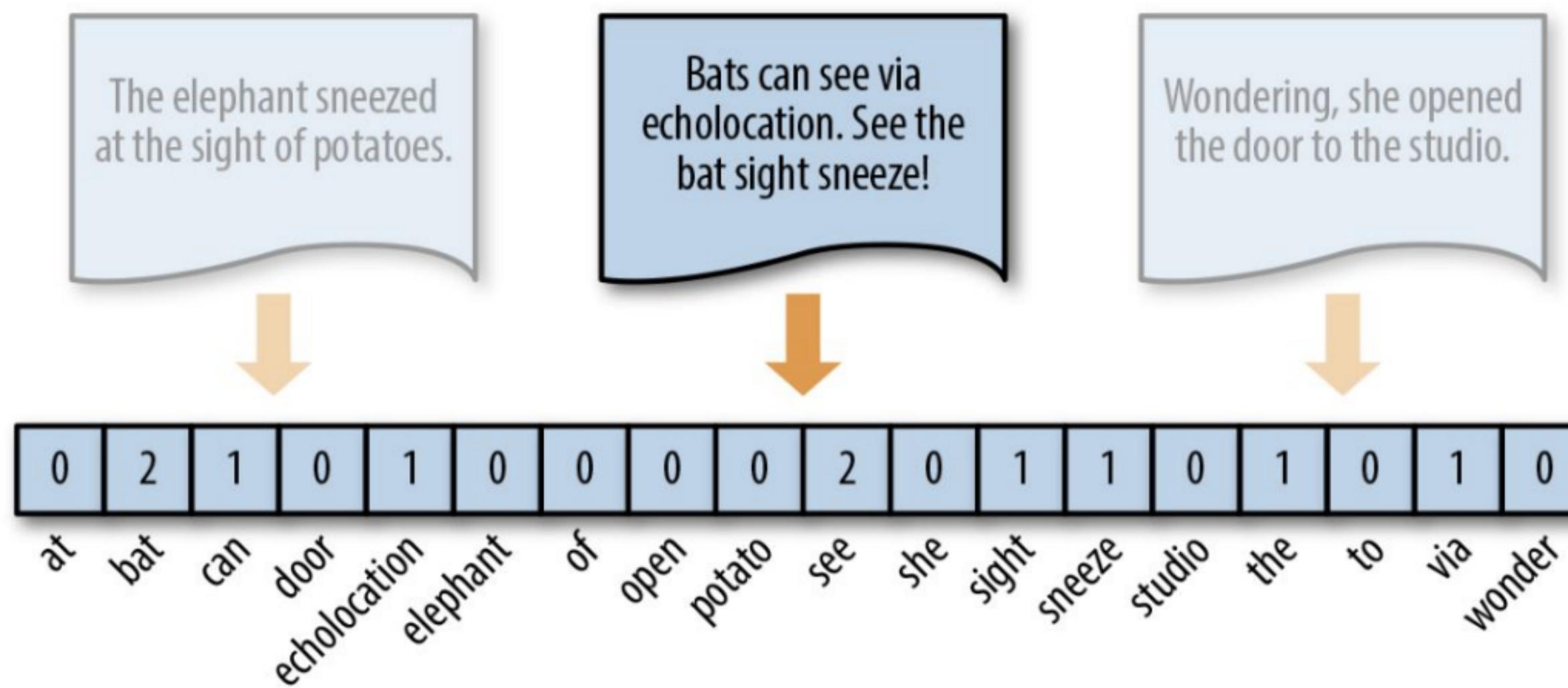
El "corpus" son todas las palabras (tokens) únicos que existen en nuestro texto.

# Vectorizar

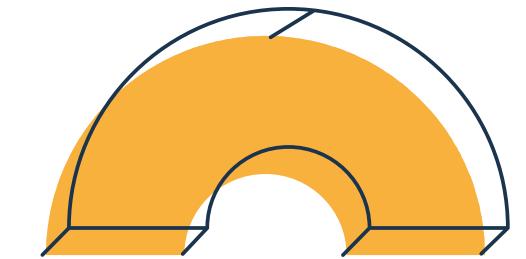


Una vez que tenemos nuestro corpus, vamos a tomar a cada palabra como una "feature".

Para cada texto, contaremos cuantas veces aparece cada palabra



# Vectorizar



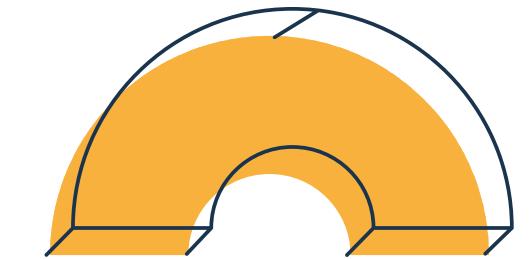
El principal problema de Bag of Words es la cantidad de features que generamos.

Imaginen en un texto grande, hay miles de palabras.

Por esto, es muy importante antes de hacer bag of words normalizar el texto y eliminar "stop words" o palabras que no aporten nada a nuestro modelo.

- Llevar todas las palabras a su raíz
- Llevar todo a minúsculas
- Eliminar caracteres especiales (salvo que nos sirvan en nuestro problema)
- Eliminar stop words

# Vectorizar



Entonces, luego de aplicar bag of words, podríamos tener algo así:

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good
Review 1	1	1	1	1	1	1	1	0	0	0	0
Review 2	1	1	2	0	0	1	1	0	1	0	0
Review 3	1	1	1	0	0	0	1	0	0	1	1

Y entrenar un modelo como veníamos haciendo hasta ahora.

# N-Grams



Cuando hablamos de tokenizar, decíamos que un token puede ser una palabra o un conjunto de palabras.

Hay palabras, que solo cobran sentido cuando están agrupadas con otras (o cambian de significado). Por ejemplo:

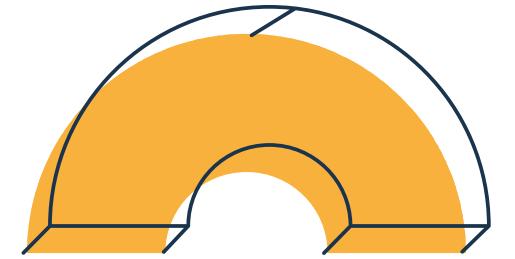
- Control remoto

No es lo mismo que

- Control
- Remoto

Por separado

# N-Grams



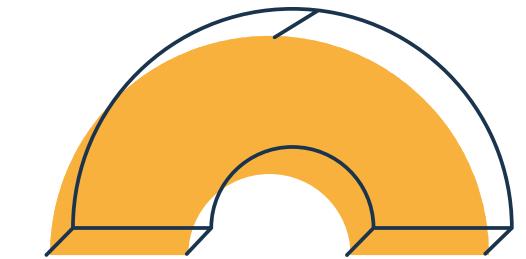
Entonces, al momento de aplicar Bag of Words, tenemos la opción de utilizar N-Gramas.

De esta forma cada "feature" será un conjunto de palabras.

Problema:

- Vamos a tener más features que antes

# TF-IDF

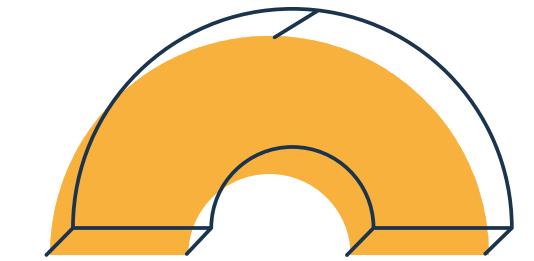


De todas formas, por más que saquemos stop words, llevemos las palabras a su raíz, etc etc, vamos a seguir teniendo palabras que tienen más peso o importancia que otras.

Si tenemos palabras que aparecen en muchos documentos, estas palabras no nos aportan información.

Entonces, parecería bueno darle un peso a cada palabra dependiendo de su importancia.

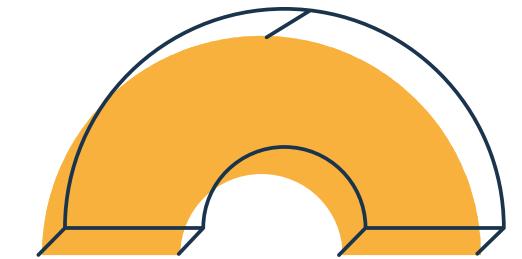
# TF-IDF



TF-IDF = Term frequency - inverse document frequency

La idea de TF-IDF es medir no sólo cuánto aparece una palabra en un documento, sino también qué tan frecuente es esa palabra en todo el corpus.

# TF-IDF



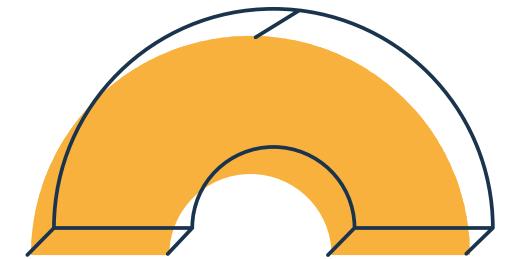
## Term frequency (TF):

Frecuencia de una palabra (term) en una instancia o documento (doc).

$\text{TF}(\text{term}, \text{doc}) = \frac{\# \text{ de veces que aparece el term en el doc}}{\# \text{ de terms diferentes en el doc}}$

0.125      0.125      0.375  
Hello, my name is Brandon. Brandon Brandon. The elephant jumps over the moon.

# TF-IDF



Document frequency (DF):

Cantidad de documentos que contienen el término

$$DF(\text{term, corpus}) = \frac{\# \text{ docs que contienen el term}}{\# \text{ cantidad total de docs}}$$

# TF-IDF



Inverse document frequency (IDF):

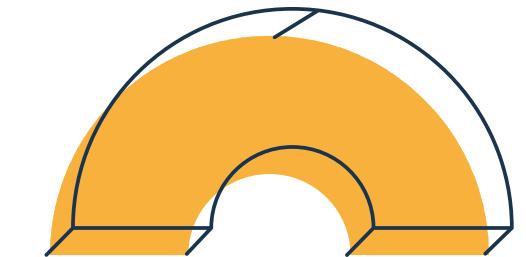
Logaritmo inversa de DF.

$$\text{IDF}(\text{term, corpus}) = \text{Log}\left(\frac{\# \text{ cantidad total de docs}}{\# \text{ docs que contienen el term}}\right)$$

Ejemplo: si está en todos los docs  $\log(N/N) = \log(1) = 0$ .

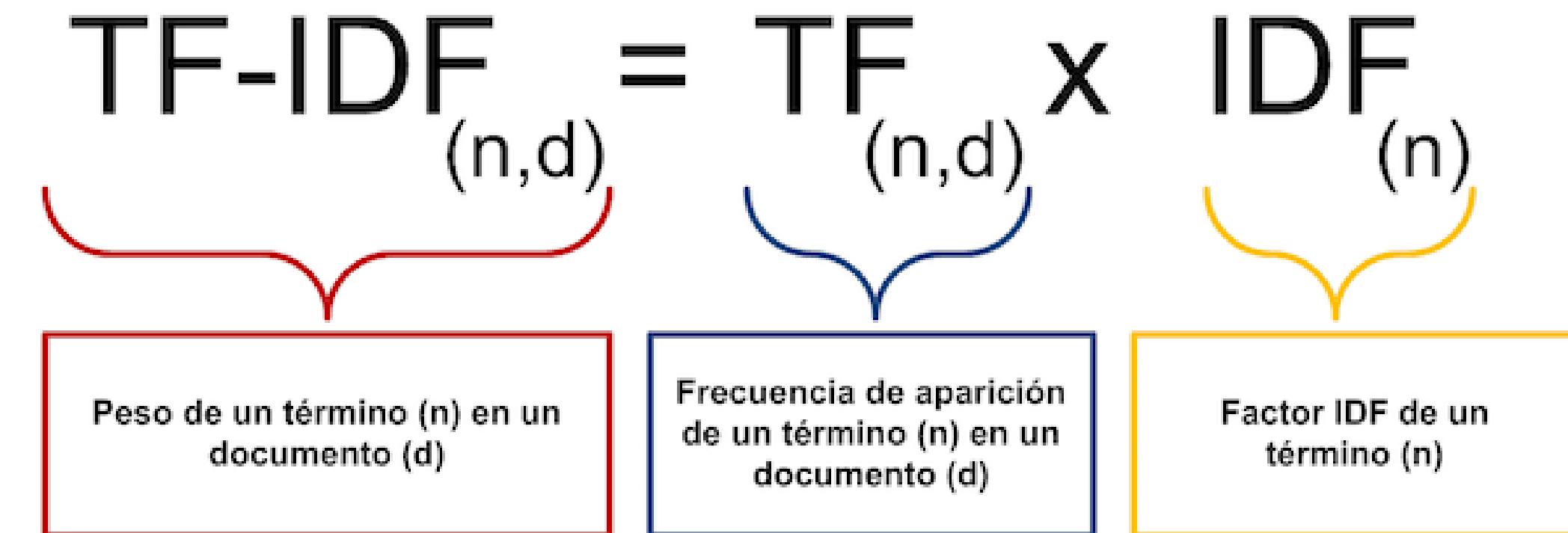
Etonces: Si un térm aparece en muchos documentos, su tf-idf va a ser bajo

# TF-IDF



## TF IDF

$$\text{TF-IDF}(\text{term}, \text{corpus}, \text{doc}) = \text{TF}(\text{term}, \text{doc}) \times \text{IDF}(\text{term}, \text{corpus})$$

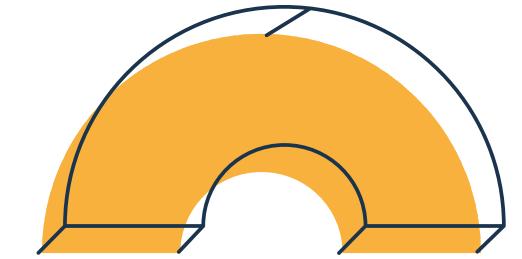




Twitter

---

# Twitter

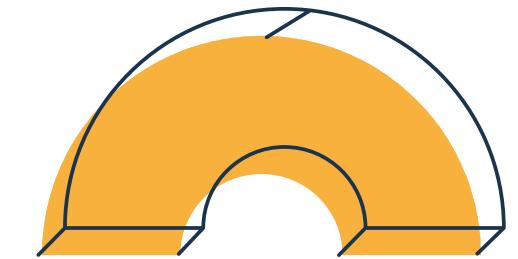


La red social twitter, nos permite tener una "developer account". Con esta cuenta, podemos a partir de una API extraer tweets (y muchas cosas más).

En nuestro caso, la usaremos para extraer tweets de algún tema en específico (ya que podemos filtrar por keywords) y luego usaremos esos datos para análisis.

Nadie está obligado a crearse una cuenta, yo puedo armar un dataset y pasárselos, pero de todas formas veremos como se hace para introducir algunos conceptos que necesitaremos conocer siendo data scientists como ¿Qué es una API?

# Twitter

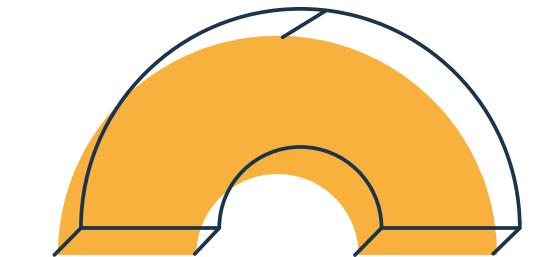


Primero que nada, (los que quieran, no es obligatorio) deberán registrarse en twitter. Si ya tienen una cuenta, pueden usar su cuenta propia. No se preocupen que no tiene ningún costo, no va a publicar nada en su cuenta, ni va a pasar nada raro.

Una vez que estén registrados, seguiremos el siguiente tutorial:

<https://developer.twitter.com/en/docs/twitter-api/getting-started/getting-access-to-the-twitter-api>

En nuestro caso, iremos por la opción "Standard"



# How to get access to the Twitter API

## Step one: Apply and receive approval for a developer account

To make any request to the Twitter API, you must first apply for a developer account and have your use case approved.

You can choose to apply for the [Standard or Academic Research product tracks](#), which offer tailored support, access levels, and pricing.

- **Standard** - The default product track for most developers, including those building something for fun, for a good cause, to learn or teach. All approved developers will be able to create and use a Standard [Project](#).
- **Academic Research** - This product track provides qualified academic researchers access to elevated access and enhanced functionality, including access to the [full-archive search endpoint](#), a higher monthly [Tweet cap](#), and enhanced filtering capabilities with the filtered stream and recent search endpoints.

Once approved, you can create a Standard or Academic Research [Project](#) and an associated [developer App](#) which will provide you a set of credentials that you will use to [authenticate](#) all requests to the API.

We require an approved use case to use the Twitter API to protect the people that use Twitter. Before you apply, we strongly encourage you to understand our [developer policy](#), and to review our list of [restricted use cases](#). If your use case does not adhere to our policy, we will reject your application.

[Apply for the Standard product track >](#)

[Apply for the Academic Research product track >](#)

# Twitter



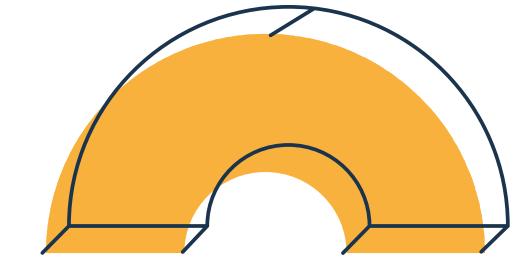
Get started with Twitter APIs and tools

## Apply for access

Apply for a developer account

Restricted used cases →

# Twitter



Deberán contestar algunas preguntas que les hará twitter para activar su cuenta de desarrolladores.

Simplemente contesten con la verdad, están aprendiendo y quieren usar los datos para practicar.

Probablemente demore unos días en activarse su cuenta, por eso lo hacemos esta semana (ya que lo usaremos en la próxima clase)

La próxima clase aprenderemos a consumir tweets desde python y luego los analizaremos.

Utilizando machine learning aprenderemos a:

- Extraer sentimientos
- Extraer palabras clave
- Extraer tópicos
- Extraer entidades
- y más ...



BREAK!