

ACÀMICA

---

# ¡Bienvenidos/as a Data Science!



# Agenda

---

¿Cómo anduvieron?

Repaso: Procesamiento de Lenguaje Natural

Explicación: Term Frequency - Inverse Document Frequency

¿Sabías que...?

Hands-On + Break

¿Sabías que...?

Cierre



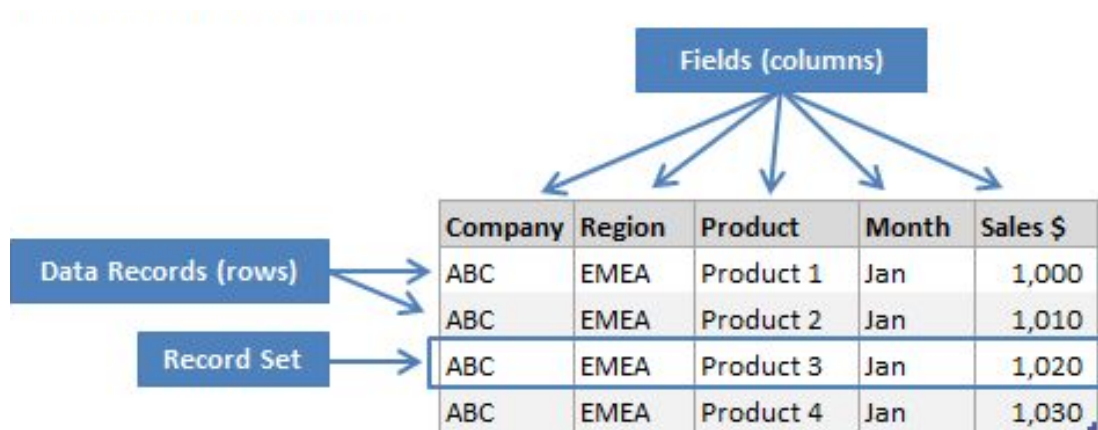
# ¿Cómo anduvieron?



# Repaso: Procesamiento de Lenguaje Natural (NLP)



Sabemos trabajar con datos **estructurados** (tablas y números).



Sabemos trabajar con datos **estructurados** (tablas y números).

**Problema:** Hay muchísimos datos disponibles en forma de lenguaje natural (texto, no estructurado) que contienen información relevante.

Data R

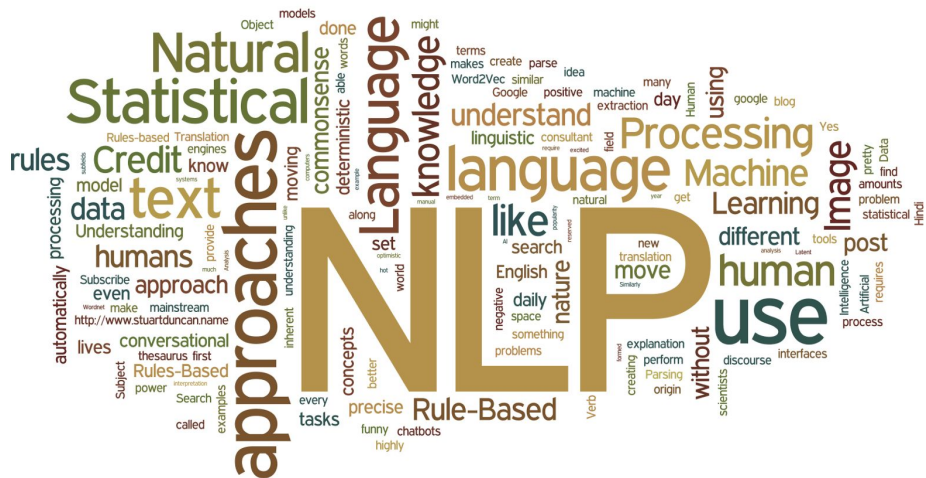
¿Cómo hacemos para darle sentido a estos datos y trabajar con ellos en el marco de Ciencia de Datos?



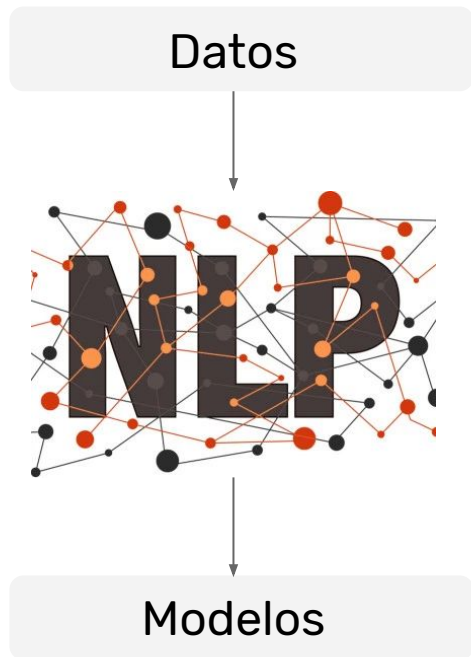


# NLP es la solución

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



# NLP • Flujo de trabajo

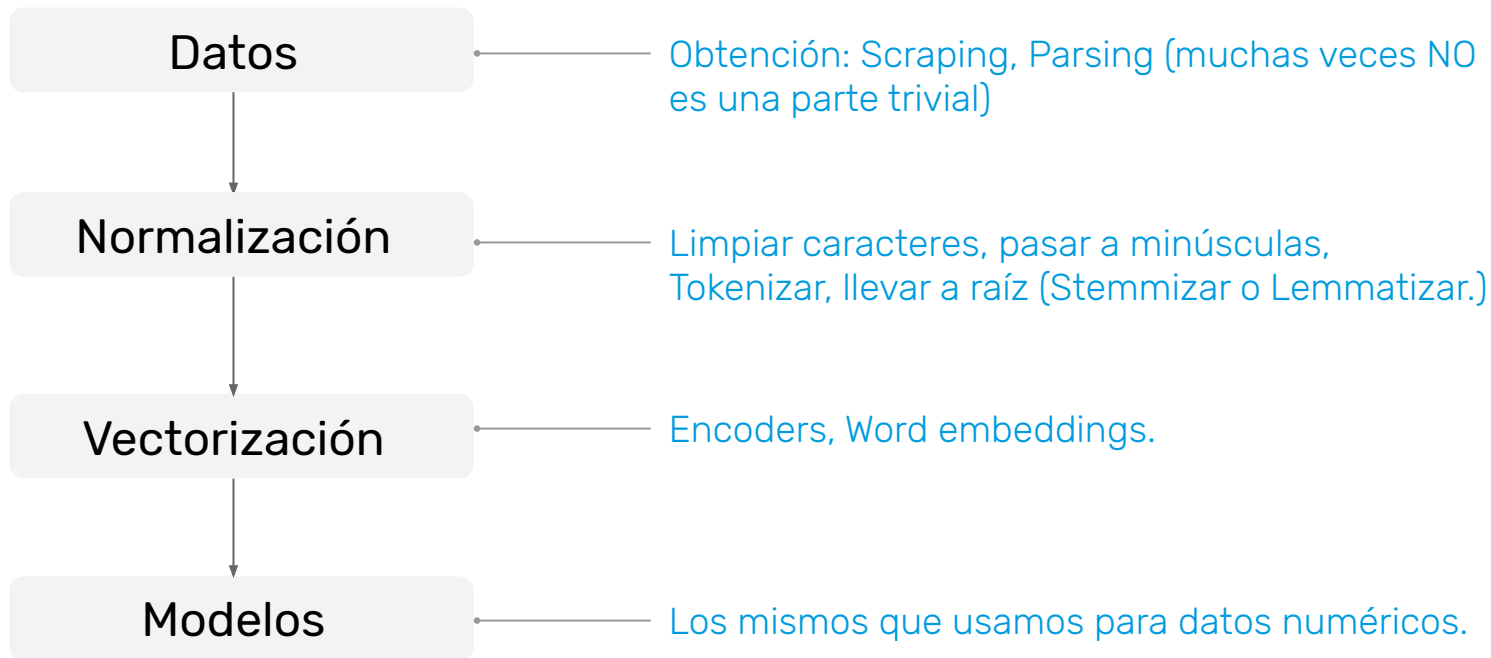


# NLP • Flujo de trabajo



Para esta vamos a usar la librería NLTK de Python

# NLP • Flujo de trabajo



# Normalizar

Quiero Pasear a mi perro por #Palermo



Quisiera pasear a mis perros por Palermo

quiero pasear mi perro Palermo

*Buscamos  
llevarlo a una  
forma común*

## Normalizar • Formas de hacerlo

- **Pasar a minúsculas:** pasar todos los caracteres de un texto a su forma minúscula para homogeneizar.

---

“Esto es un texto. Tiene varias oraciones. Todas son distintas, ninguna es igual.”

`texto.lower()`



“esto es un texto. tiene varias oraciones. todas son distintas, ninguna es igual.”

# Normalizar • Formas de hacerlo

- **Tokenizar:** pasar de un único string de texto a una lista de strings de oraciones.
- 

“esto es un texto. tiene varias oraciones. todas son distintas, ninguna es igual.”



[“esto es un texto.”,  
“tiene varias oraciones.”,  
“todas son distintas,  
ninguna es igual.”]

`nltk.tokenize.sent_tokenize(texto)`

# Normalizar • Formas de hacerlo

- **Tokenizar palabras:** pasar de un único string de una oración a una lista de strings de Tokens (palabras, puntuaciones, símbolos).
- 

"esto es un #hashtag."



["esto", "es", "un", "#",  
"hashtag", "."]

```
nltk.tokenize.word_tokenize(texto)
```



## Normalizar • Formas de hacerlo

- **Limpiar caracteres:** nos quedamos sólo con los caracteres de interés. Esto dependerá de nuestro problema en particular. En nuestro caso vamos a utilizar la librería 're', que nos permite modificar texto.

---

["esto es un #hashtag."]



["esto es un hashtag"]

```
import re
re.sub("[^a-zA-Z]", "", str(texto))
```

# Normalizar • Formas de hacerlo

- **Llevar a raíz:** buscamos llevar palabras distintas con significados similares a una forma común.

- 
- **Opción 1: Stemmizer:** Logra esto recortando las palabras mediante un proceso heurístico. Es rápido y fácil de usar, pero a veces no es certero.

["starting", "wants",  
"repartitions",  
"america's"]



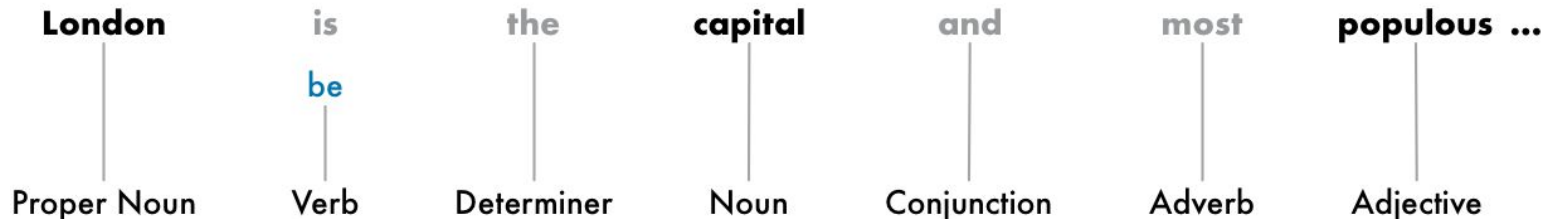
["start", "want", "repar",  
"america"]

```
from nltk.stem import PorterStemmer  
stemmer = PorterStemmer()  
stemmer.stem(palabra)
```

# Normalizar • Formas de hacerlo

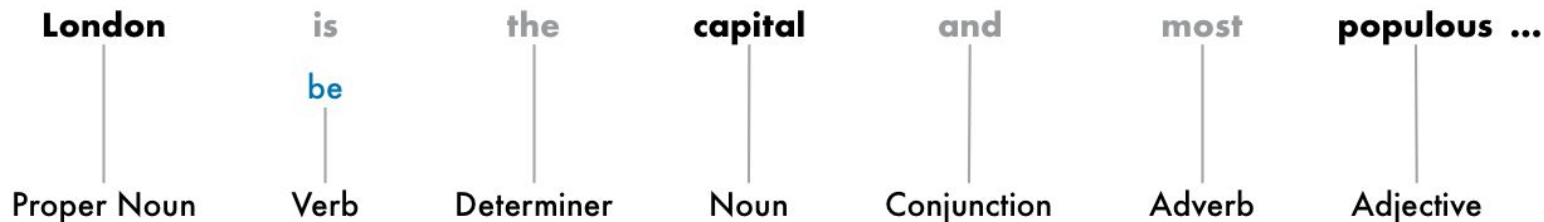
- **Llevar a raíz:** buscamos llevar palabras distintas con significados similares a una forma común.

- 
- **Opción 1: Lemmatizer:** Logra esto utilizando un vocabulario y realizando un análisis morfológico de las palabras. Precisa que además de la palabra se le informe cual es la función de la palabra en el texto



# Normalizar • Formas de hacerlo

- **Opción 1: Lemmatizer:** Logra esto utilizando un vocabulario y realizando un análisis morfológico de las palabras. Precisa que además de la palabra se le informe cual es la función de la palabra en el texto



Para determinar la función de la palabra automáticamente nos ayudamos con la **función 'nltk.pos\_tag'**. A esta función se le llama POS (Part of Speech)

# Normalizar • Formas de hacerlo

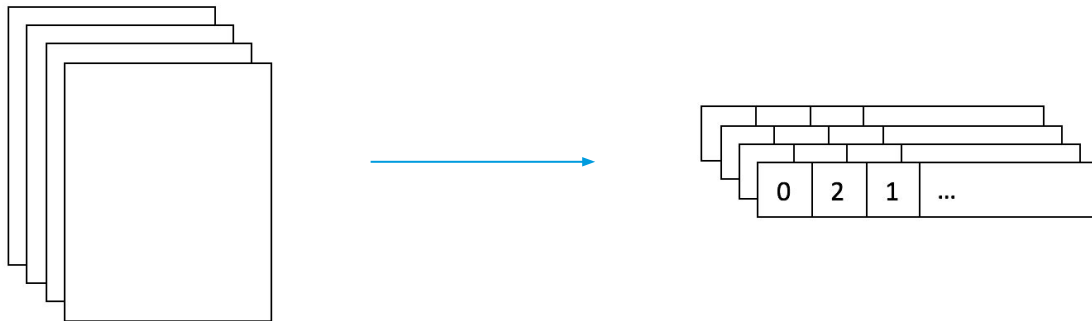
["was", "running", "hours"]  ["be", "run", "hour"]

```
from nltk.stem import WordNetLemmatizer  
wordnet_lemmatizer = WordNetLemmatizer()  
wordnet_lemmatizer.lemmatize(palabra,  
                               get_wordnet_pos(palabra))
```

Es más preciso que el Stemmer, pero lleva más tiempo y su performance depende de la precisión con la que le pasemos los POS.

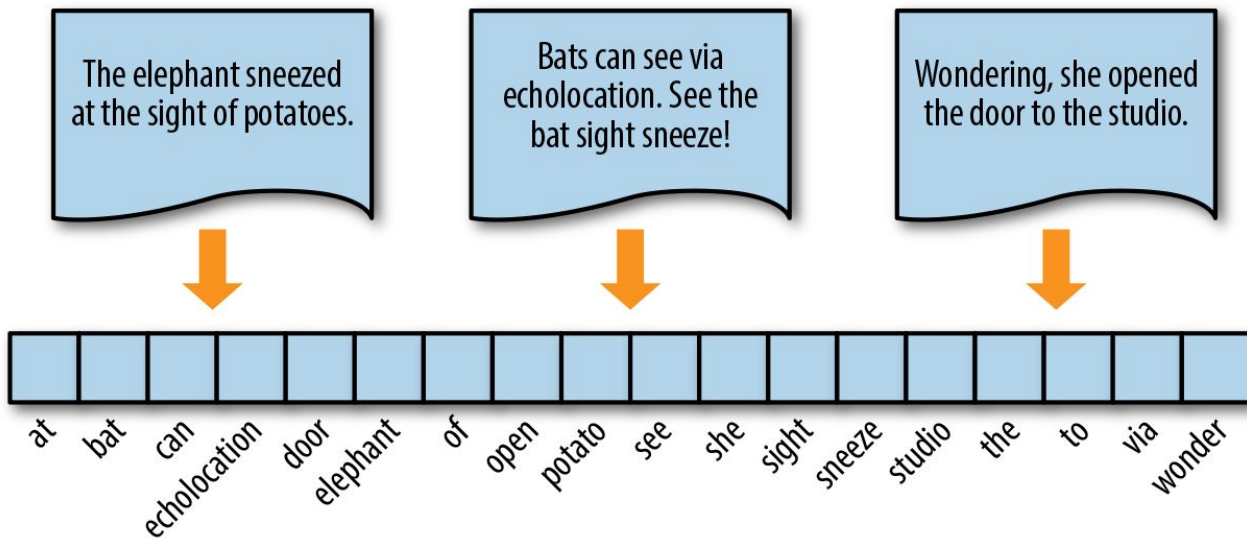
# Vectorizar

**Objetivo:** Representar cada texto (instancia de la base de datos) como un vector que podamos usar como vector de features para entrenar una de los modelos



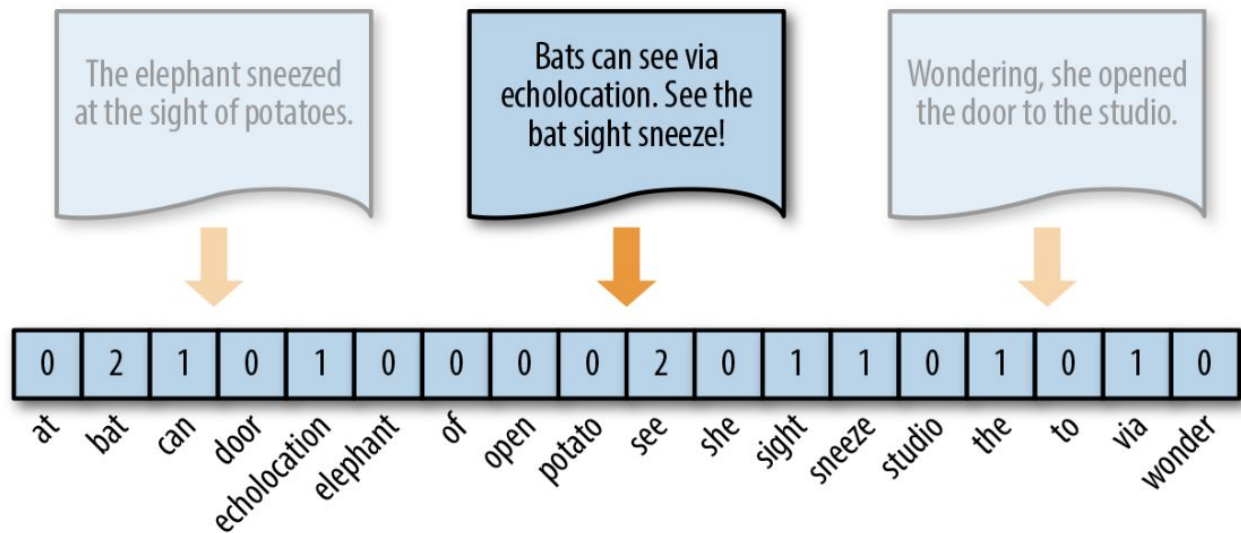
# Vectorizar • Bag of words

**Idea:** Generar un vector que represente todas las palabras del corpus. Representar cada instancia como un vector con la cantidad de veces que aparecen las palabras.



# Vectorizar • Bag of words

**Idea:** Generar un vector que represente todas las palabras del corpus. Representar cada instancia como un vector con la cantidad de veces que aparecen las palabras.





## Vectorizar • Bag of words

Para implementar esto utilizamos una función de sklearn llamada CountVectorizer:

```
from sklearn.feature_extraction.text import CountVectorizer
```

## Vectorizar • Bag of words

Para implementar esto utilizamos una función de sklearn llamada CountVectorizer:

**Problema:** la cantidad de palabras en la base de datos suele ser muy grande. No conviene tener tantos features.

## **Solución (por ahora)**

Utilizamos sólo las palabras que aparecen una mayor cantidad de veces en el texto, o que aparecen en un mayor número de instancias.

# Term Frequency - Inverse Document Frequency



## Vectorizar • Bag of words con N-gramas

**Problema:** hay palabras que cobran sentido cuando se las agrupa con otras, ejemplos: “Plaza Italia” y “Control Remoto” .

# Vectorizar • Bag of words con N-gramas

## Solución

Además de cada palabra por separado, agregamos los grupos de 2 (ó N) palabras contiguas a nuestro vector de Features.

---

**Para implementar esto usando CountVectorizer:**

```
CountVectorizer(max_features=max_features, stop_words="english" , ngram_range=(1, 2))
```

# Vectorizar • Bag of words con N-gramas

## Solución

Además de cada palabra por separado, agregamos los grupos de 2 (ó N) palabras contiguas a nuestro vector de Features.

---

**Para implementar esto usando CountVectorizer:**

```
CountVectorizer(max_features=max_features, stop_words="english", ngram_range=(1, 2))
```



Ojo con la cantidad de Features

# Vectorizar • TF - IDF

**Observación:** si buscamos diferenciar cada documento por las palabras que lo componen, las palabras que están en todos ellos no aportan información.



# Vectorizar • TF - IDF

**Observación:** si buscamos diferenciar cada documento por las palabras que lo componen, las palabras que están en todos ellos no aportan información.

**Idea:** hay que medir no sólo cuanto aparece una palabra en una instancia (documento), sino también qué tan frecuente es esa palabra en todo el corpus.

**Term Frequency - Inverse Document  
Frequency**

TF - IDF

# Vectorizar • TF - IDF

## Term Frequency

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

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

# Vectorizar • TF - IDF

## Term Frequency

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

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

Ejemplo en un documento:

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

# Vectorizar • TF - IDF

## Document Frequency

Fracción de todos los documentos en nuestro corpus que contienen el término.

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

Ejemplo en un documento:

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

# Vectorizar • TF - IDF

## Inverse Document Frequency

Logaritmo inversa de DF.

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

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

# Vectorizar • TF - IDF

## Inverse Document Frequency

Producto del valor de TF por el de IDF.

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

# Vectorizar • TF - IDF

## Inverse Document Frequency

Producto del valor de TF por el de IDF.

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

Cada palabra tiene un valor asociado en cada documento, con esto formamos nuestro vector (no necesariamente serán valores enteros):

0	0.2	0.5	0	0.3	0	0	0	0	2	0	0.1	1	0	1	0	1	0
at	bat	can	door	echolocation	elephant	of	open	potato	see	she	sight	sneeze	studio	the	to	via	wonder

# GPT-2

## Parte 1





SYSTEM PROMPT  
(HUMAN-WRITTEN)

*A train carriage containing controlled nuclear materials was stolen today. Its whereabouts are unknown.*

MODEL  
COMPLETION  
(MACHINE-  
WRITTEN, FIRST  
TRY)

The incident occurred on the downtown train line, which runs between Ashland stations.

In an email to Ohio news outlets, the U.S. Department of Energy said it was working with the Federal Railroad Administration to find the stolen material.

"The theft of this nuclear material will have significant implications for the safety of our nation on public and environmental health, our workforce and the economy," said Tom Hicks, the U.S. Energy Secretary. "Our top priority is to secure the theft and ensure it doesn't happen again."

The stolen material was taken from the University of Cincinnati's Triangle Park nuclear research site, according to a news release from the Department officials.

The Nuclear Regulatory Commission did not immediately release any details about the theft.

According to the release, the U.S. Department of Energy's Office of Nuclear Energy is



TWO MINUTE  
PAPERS

WITH KÁROLY ZSOLNAI-FEHÉR

# Hands-on training



DS\_Encuentro\_33\_NLP.ipynb

Parte 4 en adelante



# Para jugar



## Talk to Transformer

See how a modern neural network completes your text. Type a custom snippet or try one of the examples. [Learn more](#) below.

[Follow @AdamDanielKing](#) for more neat neural networks.

**[Sponsor the site.](#) Show your product to 100,000s of monthly visitors.**

Custom prompt



Type something and a neural network will guess what comes next.

COMPLETE TEXT

# GPT-2

## Parte 2





# TWO MINUTE PAPERS

WITH KÁROLY ZSOLNAI-FEHÉR

# Para la próxima: Data Science en mi vida



# Data Science en mi vida

En 10/15 minutos, tendrán que contar a sus compañeros y equipo docente lo siguiente:

- a) En qué problemas estoy aplicando lo aprendido en DS y cómo lo estoy encarando.
- b) Contar algún tema que me interese o que proyecto aplicar relacionado con lo que vimos.



# Para la próxima

---

1. Trabajar en la Entrega 05
2. Completar notebooks de hoy y atrasados.
3. Ver videos de la plataforma “Aprendizaje No Supervisado”.
4. Preparar el relato “Data Science en mi vida”.

ACÀMICA