

ACÀMICA

¡Bienvenidos/as a Data Science!



Agenda

Resultados de la Competencia. Ganadores

¿Cómo anduvieron?

Explicación: Procesamiento de Lenguaje Natural

Break

Hands-On

Lanzamiento Entrega 05

Cierre



Resultados de la competencia...



1° puesto

Los tres deseos

Lucas de la Fuente, Federico Baldi

Score: 0.715903

2° puesto

Chicago Bulls

Horacio González, Juanjo Campodónico, Juampi Martínez

Score: 0.709754

3° puesto

Los tres chiflados

Joaquín Mouton, Federico Richard, Felipe Vacchiani

Score: 0.708648

4° puesto

Los tres poderes

Gabi Argüello, Barbi Ginés, Joaquín Tavella

Score: 0.672553

Invitamos al equipo ganador a compartir su experiencia en esta Competencia de Kaggle.

Sugerimos tener en cuenta:

- Organización del equipo (división de actividades, tiempos, etc)
- Recursos y herramientas
- Notebook realizado

¿Dónde estamos?



Cronograma

bloque

entrega

tiempo

ADQUISICIÓN Y EXPLORACIÓN

MODELADO

DEPLOY

Exploración de datos

Feature Engineering

Regresión

Optimización de parámetros

Procesam. del lenguaje natural

Sistema de recomendación

Publicación de modelos

SEM 1

SEM 5

SEM 8

SEM 12

SEM 13

SEM 18

SEM 23

SEM 2

SEM 6

SEM 9

SEM 14

SEM 19

SEM 24

SEM 3

SEM 7

SEM 10

SEM 15

SEM 20

SEM 4

SEM 11

SEM 16

SEM 21

SEM 17

SEM 22

Acá estamos

APRENDIZAJE SUPERVISADO

APRENDIZAJE NO SUPERVISADO



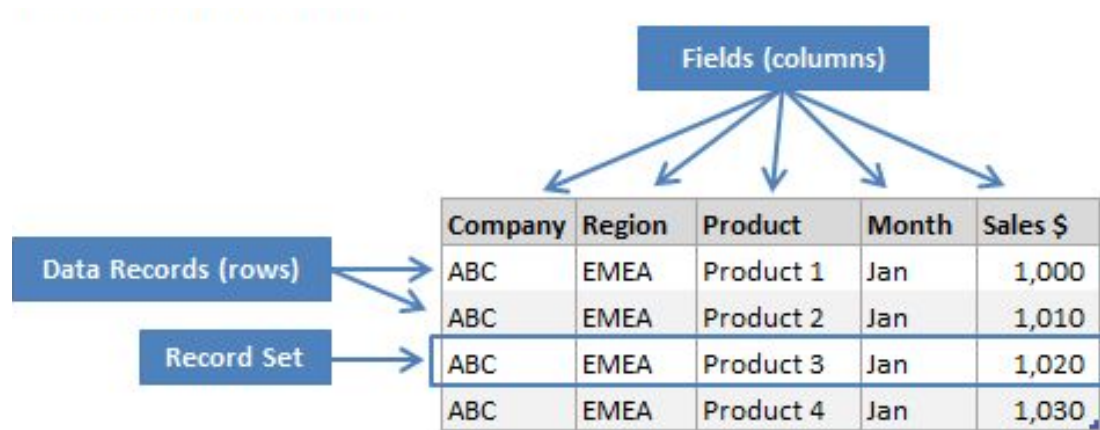
¿Cómo anduvieron?



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.

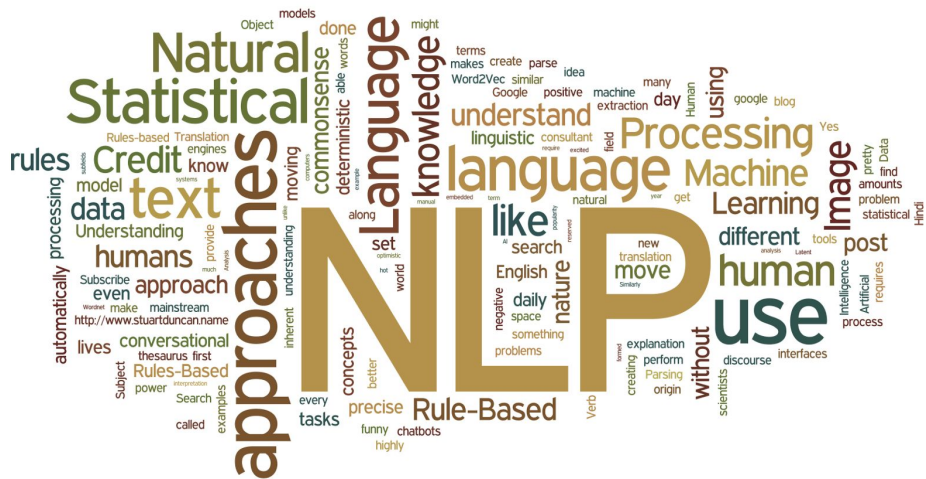
Data R

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

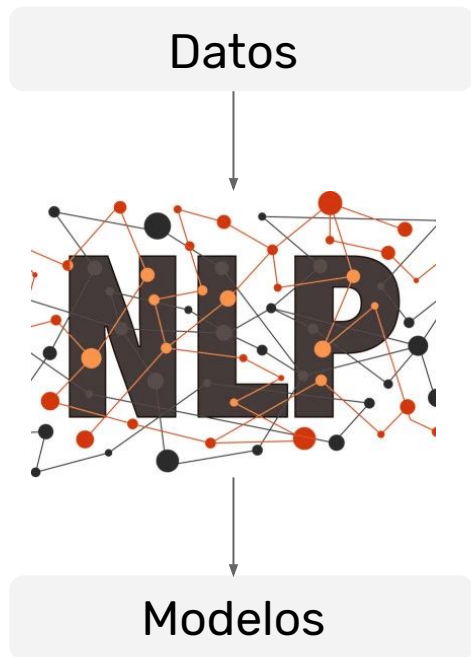


NLP es la solución

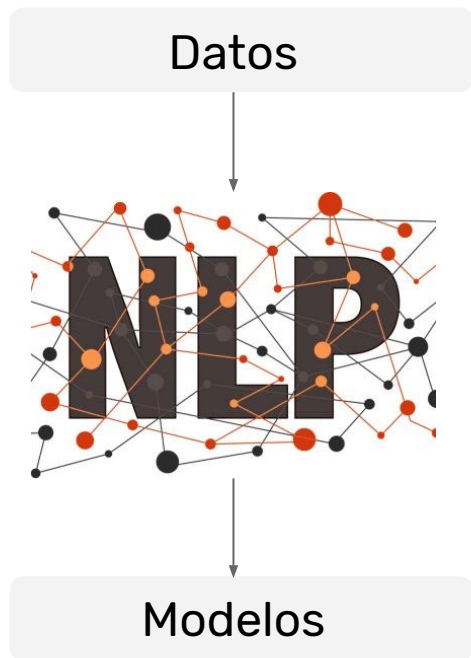
El procesamiento de lenguaje natural es una rama de la inteligencia artificial que se enfoca en permitirle a las computadoras entender y procesar lenguaje natural.



NLP • Flujo de trabajo

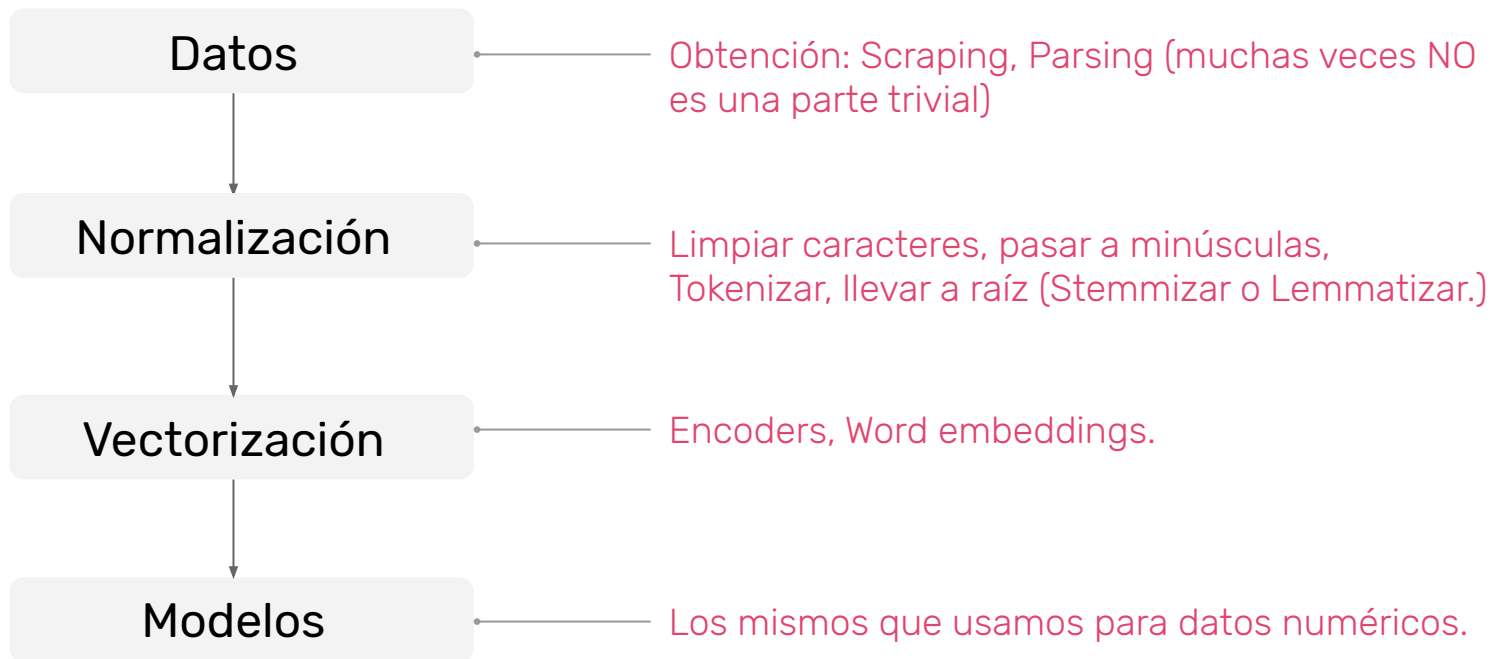


NLP • Flujo de trabajo



→ Para esta vamos a usar la librería NLTK de Python

NLP • Flujo de trabajo



Normalización



Normalizar

Idea: llevar todo el texto a un formato común donde palabras escrita de manera distinta o con significados similares se representen de la misma manera.



Normalizar

Quiero Pasear a mi perro por #Palermo



Quisiera pasear a mis perros por Palermo

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
- Tokenizar
- Limpiar caracteres
- Llevar a raíz

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

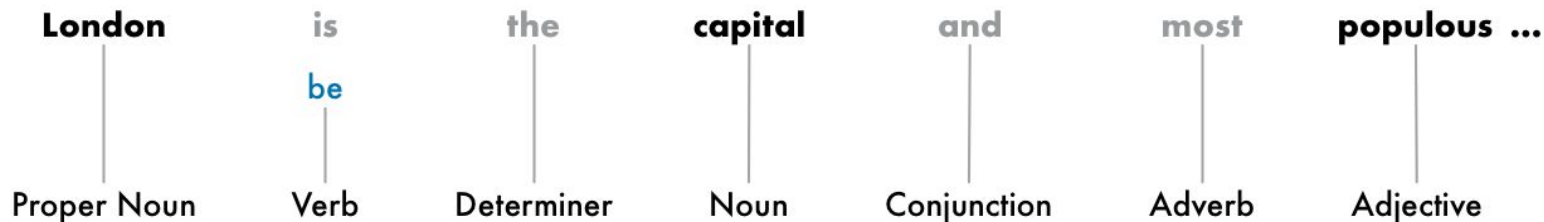
- **Llevar a raíz:** buscamos llevar palabras distintas con significados similares a una forma común.

-
- **Opción 2: 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 2: 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))
```

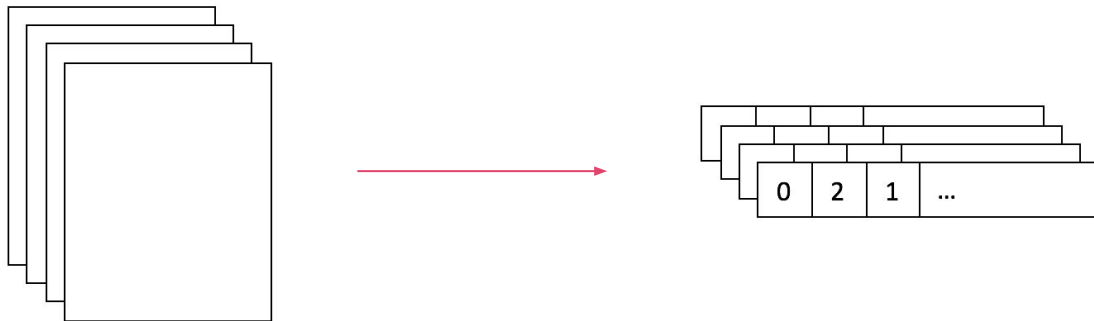
Es más preciso que el Stemmer, pero lleva más tiempo y su performance depende de la precisión con la que le pasemos los POS.

Vectorización



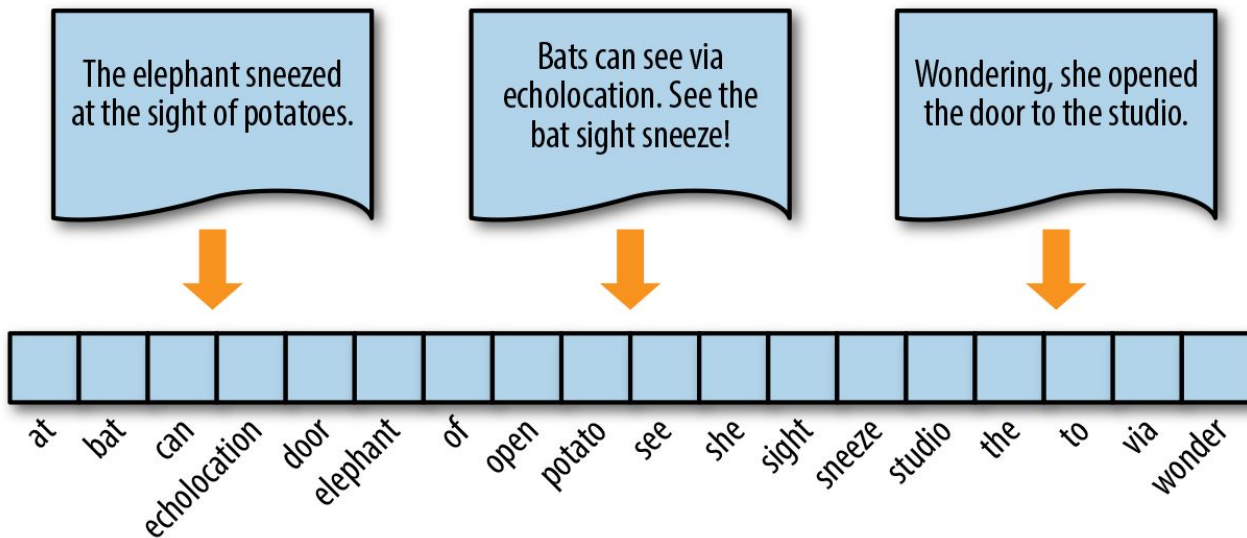
Vectorizar

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



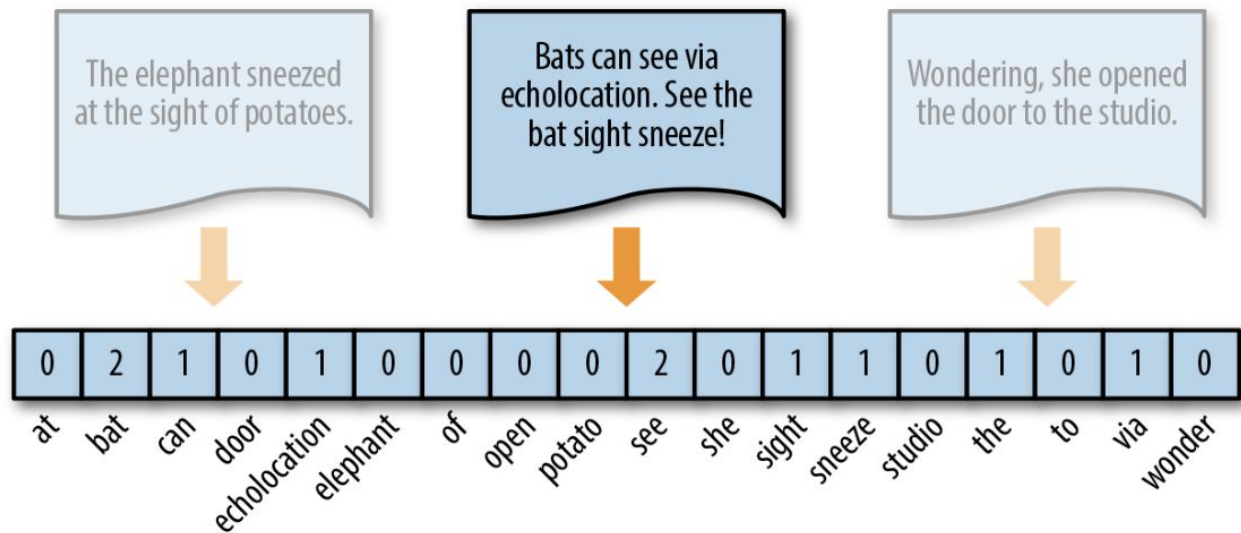
Vectorizar • Bag of words

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.

A close-up photograph of a white ceramic cup filled with a latte. The surface of the milk is decorated with intricate latte art, featuring a central heart shape surrounded by concentric, wavy lines. The cup is placed on a matching white saucer. In the background, a white napkin and a silver fork are visible, though they are out of focus. The overall lighting is soft and warm, creating a cozy atmosphere.

¡BREAK!



Hands-on training



DS_Encuentro_33_NLTK.ipynb

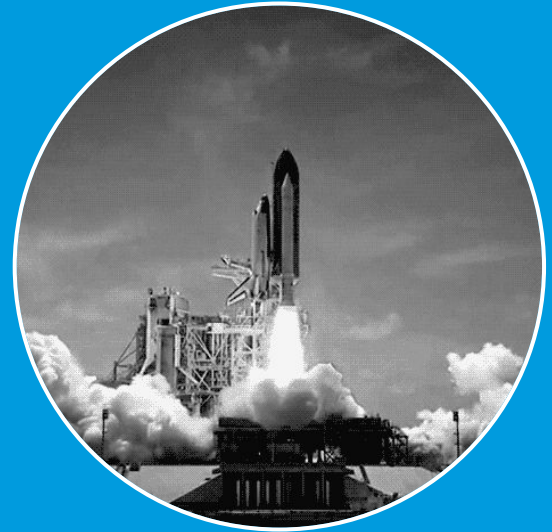
Parte 1 a 3



Proyecto 2:

Lanzamiento

Entrega 05



Proyecto 2: Modelado (Entrega 05)



Entrega 5: Procesamiento...



Conocé como procesar el lenguaje natural para usarse en clasificación

● Beginner

by



Francisco Dorr

1. Bajar los materiales.
2. Leer la Checklist
3. ¡Empezar a trabajar en la entrega!

Recursos



NLP

Introducción corta a NLP, accesible y gráfica (con muy poca matemática):
<https://towardsdatascience.com/introduction-to-natural-language-processing-for-text-df845750fb63>

Otra parecida, pero más enfocada en la parte lingüística:
<https://medium.com/@gon.esbuyo/get-started-with-nlp-part-i-d67ca26cc828>

Bibliografía extendida sobre NLP con NLTK (libro open access):
Natural Language Processing with Python - <https://www.nltk.org/book/>



Recordatorio sobre condiciones de aprobación



Proyectos • Condiciones de aprobación

Los/as evaluadores/as considerarán una entrega como **Aprobada** cuando el/la estudiante haya cumplido satisfactoriamente el 100% de los puntos que pide el checklist (aunque no los haya hecho a todos perfectos).

Caso contrario, el/la evaluador/a considerará la entrega como **Para rehacer.**

Nota: no hay un límite de iteraciones (el/la estudiante puede tener que rehacer su trabajo todas las veces que sea necesario hasta aprobar).

Para la próxima

1. Terminar de ver los videos de la plataforma “Procesamiento del Lenguaje Natural”
2. Trabajar en la entrega 05.
3. Completar Notebook de hoy y atrasados.

ACÀMICA