UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Integrantes:

- Maydelith Laura Zuñiga Cabrera - 202510437
- Joaquin Mathias Alejandro Llallire Meza - 202510406
- Joseph Paulo Geraldo Soto - 202510412
- Juan Manuel Moreno Moran - 202510417
- Leonardo Julian Huanay Quiroz - 202510516

Nombre del estudiante	Porcentaje de aporte
Maydelith Laura Zuñiga Cabrera	30%
Joaquin Mathias Alejandro Llallire Meza	20%
Joseph Paulo Geraldo Soto	20%
Juan Manuel Moreno Moran	0%
Leonardo Julian Huanay Quiroz	30%

Actividades realizadas por integrantes:

Informe:

Marco teórico y Antecedentes: Joseph Paulo Geraldo Soto

Descripciones del proceso realizado por pregunta: Maydelith Laura Zuñiga Cabrera

Interpretación de los cluster de la pregunta 1 y 2: Joaquin Mathias Alejandro Llallire Meza

Recursos Extras usados en el código: Leonardo Julian Huanay Quiroz

Interpretación de los clusters de la pregunta 3 y comparación con pregunta 1: Maydelith Laura Zuñiga Cabrera

Conclusiones: Todos

Código:

Json: Leonardo Julian Huanay Quiroz

Pregunta 1: Joaquin Mathias Alejandro Llallire Meza y Joseph Paulo Geraldo Soto

Pregunta 2: Leonardo Julian Huanay Quiroz

Pregunta 3: Maydelith Laura Zuñiga Cabrera

Antecedentes:

1. El mundo pokemon:

   Pokemon, marca registrada de Nintendo, es una franquicia de entretenimiento que consiste de seres fantasiosos capaces de cosas inimaginables en la realidad, como crear llamaradas de fuego, lanzar chorros poderosos de agua, truenos, etc. Los entrenadores son personas que se dedican a capturarlos y entrenarlos para combates, mediante pokebolas donde se almacenan estos.

2. Pokemon y sus tipos:

   Existen más de mil diferentes movimientos entre todos los tipos de pokemon juntos, algunos más comunes y otros más específicos para ciertos tipos. De tipos solo hay 18. Con esta información, podría encontrarse una forma de agruparlos mediante tipo o mediante capacidad de aprendizaje de movimientos.

Marco Teórico:

1. ¿Qué es Clustering?

   Es un algoritmo no supervisado de machine learning que se encarga de agrupar diferentes objetos o datos de acuerdo a las semejanzas entre estos. (IBM, 2024)

2. ¿Qué es TF-IDF?

   Frecuencia de términos - Frecuencia inversa de documentos, o por sus siglas en inglés TF-IDF, es un algoritmo de preprocesamiento de datos que se encarga de calcular cuán frecuente es el elemento entre todas las palabras existentes en el documento, y su prevalencia entre todos los documentos mediante la siguiente fórmula:

   Frecuencia de palabra en documento * Frecuencia entre todos los documentos / Número de documentos

   (IBM, 2024)

3. ¿Qué es K-Means?

   Es un algoritmo no supervisado que se encarga de dividir un “dataset” en varias particiones de acuerdo a la distancia de cada elemento a uno

o varios centroides establecidos anteriormente. (IBM, 2024)

4. ¿Qué es PCA?

   Análisis de Componentes Principales, o PCA por sus siglas en inglés, es un algoritmo utilizado con el fin de preprocesar una matriz de datos que cuenta con demasiadas dimensiones, reduciendo a un espacio más reducido para evitar que se generalice la información al ser utilizado en otros algoritmos. (IBM, 2024)

5. ¿Qué es archivo CSV?

   Archivo de texto que almacena datos en formato de tabla, donde cada línea representa un registro y cada campo está separado por una coma. (Microsoft, 2025)

6. ¿Qué es el paquete Pandas?

   Pandas es una biblioteca de Python de código abierto ampliamente utilizada para el análisis de datos y el aprendizaje automático.(University of Virginia, 2024)

7. ¿Qué es el paquete scikit-learn?

   La librería scikit-learn, también llamada sklearn, es un conjunto de rutinas escritas en Python para hacer análisis predictivo, que incluyen clasificadores, algoritmos de clusterización, etc. Está basada en NumPy, SciPy y matplotlib, de forma que es fácil reaprovechar el código que use estas librerías.(Universidad Oberta de Catalunya, 2023)

8. NLTK (Natural Language Toolkit): proporciona una lista predefinida de Stop Words en varios idiomas, la cual se puede utilizar para eliminar estas palabras durante el preprocesamiento de texto.(ProjectPro, 2023)

Agrupamiento mediante TF-IDF

Descripción del proceso realizado:

El código implementado realiza un proceso completo de agrupamiento de Pokémon basado en sus movimientos utilizando la técnica TF-IDF. A continuación, se detallan los pasos realizados y cómo cumplen con los requisitos solicitados:

1. Generación de la matriz TF-IDF con n-gramas

Se utilizó TfidfVectorizer de scikit-learn con un rango de n-gramas de 1 a 3 (unigramas, bigramas y trigramas), lo que permite capturar no solo palabras individuales sino también secuencias de 2 y 3 palabras. Esto se evidencia en la línea:

vec = TfidfVectorizer(stop_words=stopwords.words("english"), ngram_range=(1,3))

2. Conteo e impresión de tokens

   El código calcula y muestra el número total de tokens generados, así como la lista completa de estos:

   tokens = vec.vocabulary_.keys()

   print(f"El número total de tokens es: {len(tokens)}")

   print(f"Tokens: {tokens}")

3. Creación del DataFrame con la matriz TF-IDF

   Se generó un DataFrame donde las columnas corresponden a los tokens del vocabulario y los valores son los pesos TF-IDF para cada Pokémon (Se incluye la columna “Pokemon” para relacionar cada pokémon con su debida data):

   smogon_agrupado=pd.DataFrame(data=agrupamiento.toarray(),columns=sorted(tokens))

   print(smogon_agrupado)

4. Agrupamiento con K-Means

   Se aplicó el algoritmo K-Means con 21 clusters (18 clusters por la cantidad de tipos de pokemones y 3 por los tipos especiales que necesitaban su propio cluster) para agrupar los Pokémon basándose en sus vectores TF-IDF:

   km = KMeans(n_clusters=21, n_init=100)

   lista = km.fit_predict(smogon_agrupado)

5. Generación del archivo CSV

   Se creó un archivo CSV que contiene:

   - Los nombres de los Pokémon
   - La matriz TF-IDF completa
   - Los tipos primario y secundario de cada Pokémon en base a la asignación de clusters.
   - La asignación de cluster para cada Pokémon

   smogon_agrupado.to_csv("smogon_agrupado_1.csv")

6. Interpretación de clusters:

   El código utiliza el archivo JSON (pokedex.ts):

Se extrae información de tipos mediante file_to_json.retornar_tipos(). Esto proporciona el tipo primario (Type1) y secundario (Type2) de cada Pokémon. Los tipos vienen del documento oficial de Pokémon Showdown. A partir de los tipos primarios y secundarios de pokemones se les da significado a cada uno de los 21 clusters de la siguiente manera:

La mayoría de clusters se agruparon en base a la frecuencia en la que se repetían las descripciones de sus movimientos, ello permitió agruparlos en grupos que posteriormente al añadirle los tipos a cada pokemon cobraron sentido, siendo que la mayoría compartían al menos un tipo en específico sea primario o secundaria:

Acotación: Hubo excepciones en algunos clusters, donde al menos un pokémon no coincidía en el cluster que se le fue asignado por K-means, debido a que su tipo primario y/o secundario no concordaban con el tipo asignado al cluster en general.

Clusters Nombrados:

1. Cluster 0: Normal-voladores:

   La columna moves de los pokemones que pertenecen a este cluster cuentan en su mayoría con ataques del tipo volador de categoría física.

2. Cluster 1: Fighting:

   Agrupa mayormente a todos los pokemons que cuentan con ataques del tipo lucha y relacionados con puños.

3. Cluster 2: Pokemon de Nicho

   Son aquellos pokemons que cuentan con la menor cantidad de movimientos aprendidos. Como el TF-IDF basa su procesamiento de datos en base a la frecuencia del texto y estos pokemones cuentan con la menor cantidad de datos, arrojan un resultado similar en TF-IDF. Por ello, cuando son agrupados con K-means este los ubica en un solo clúster. Ejemplo: Tanto Smeargle como Ditto únicamente aprenden un movimiento (Esquema y Transformación respectivamente).

4. Cluster 3: Poison - Estado:

   Son todos los pokemons que aprenden movimientos relacionados con púas y afectar el campo de su enemigo. Ejemplo: La gran mayoría aprende uno o varios de los siguientes ataques: Púas, Púas Tóxicas, Trampa Rocas, Fuego Fatuo, Mofa, etc.

5. Cluster 4: Grass

Son los que mayormente aprenden ataques relacionados con tipo planta.

6. Cluster 5: Water - Ofensivo:

   Pokemons de tipo agua que aprenden ataques ofensivos, como por ejemplo: Hidrobomba, Escaldar, Surf, etc.

7. Cluster 6: Rock - Steel:

   Son todos los pokemons que aprenden ataques de tipo roca y algunos de tipo acero, mayormente relacionados a subir sus estadísticas (Defensa férrea).

8. Cluster 7: Water - Defensivo:

   Pokemons de tipo agua que aprenden ataques defensivos, como por ejemplo: Armadura Ácida, Recuperación, Acua Aro, etc.

9. Cluster 8: Water - Extras:

   Son todos los pokemons que aprenden ataques ofensivos, defensivos y de aumento de estadísticas.

10. Cluster 9: Bug - Normal - Flying:

    Pokemons de primeras etapas evolutivas que aparecen en las primeras rutas de los juegos. Ejemplo: Caterpie, Weedle, todos los bichos regionales.

11. Cluster 10: Electric:

    Todos los pokemons que aprenden en su mayoría movimientos del tipo eléctrico.

12. Cluster 11: Fire

    Son todos los que aprenden movimientos de tipo fuego.

13. Cluster 12: Normal - Fairy - Soporte

    Pokemons que aprenden ataques de tipo hada o de tipo normal con el rol de soporte. Por ejemplo, Chansey (Tipo Normal) - Deseo, Protección, Campana Cura, Sylveon (Tipo Hada) - Brillo Mágico, Fuerza Lunar.

14. Cluster 13: Bug:

    Todos los pokemons que aprenden ataques de tipo bicho y no forman parte del cluster 9 porque son las etapas evolutivas posteriores de los pokemons en ese cluster. Por ejemplo: Caterpie (Del Cluster 9) -> Butterfree (Cluster 13)

15. Cluster 14: Steel:

    Pokemons que mayormente aprenden movimientos del tipo acero tanto ofensivos como defensivos. Por ejemplo: Metang, Aegislash, Ferrothorn.

    Casos especiales:

Hay líneas evolutivas de pokemon tipo bicho que aprenden movimientos del tipo acero por “Tutor de Movimientos”, siendo enviados por ello a este cluster. Ejemplo: Línea evolutiva de Scolipede.

16. Cluster 15: Dragon:

    Pokemons que aprenden movimientos de tipo dragón. También incluye a los pseudolegendarios de cada región.

17. Cluster 16: Psychic:

    Pokemons que aprenden movimientos de tipo psíquico.

18. Cluster 17: Ghost:

    Pokemons que aprenden movimientos de tipo fantasma.

19. Cluster 18: Arceus - Todas las formas

    Cada forma de Arceus está contada como un pokemon distinto en el dataFrame. Al tener el mismo contenido en la columna moves en cada una de estas formas existen 18 datos idénticos en el TF-IDF que k-means agrupa en un mismo clúster.

20. Cluster 19: Normal - Dark - Poison - Ofensivo:

    Pokemons que aprenden una mezcla de ataques de tipo normal y/o dark de categoría ofensivos.

    Casos especiales:

    Existen algunos pokemons de tipo poison que son enviados a este cluster por las razones anteriores.

    Dos de los pokemon pertenecientes al Trio de las Nubes, Tornadus y Thundurus en sus dos formas cada uno, debido a los movimientos que aprenden por MTs los cuales pertenecen al tipo Normal y Dark.

21. Cluster 20: Ice

    Pokemons que aprenden movimientos de tipo Hielo.

    Caso especial:

    Suicune (Water): Es un pokémon de tipo water, pero la mayoría de sus movimientos son de tipo Ice.

Recursos extras usados en el código:

Librería Github (PyGithub): Se usa para acceder al archivo .ts que contiene los tipos de cada Pokemon. Este archivo se encuentra en el repositorio de Pokemon Showdown.

PyGithub: https://github.com/PyGithub/PyGithub

Documentación de PyGithub: https://pygithub.readthedocs.io/en/stable/

Librería re: Se encarga de detectar ciertos caracteres mediante “Regular expressions” con el fin de agregarlos o retirarlos del string en cuestión.

re: https://github.com/blackberry/Python/blob/master/Python-3/Lib/re.py

Documentación de re: https://docs.python.org/3/library/re.html

Token de seguridad Github: Llave necesaria para

```python
import github
import os
import dotenv

#Carga un .env personal por PC
#El .env debe contener un GITHUB_TOKEN = "Inserte token de Github"
dotenv.load_dotenv()

def get_files(repo_name:str,file_path:str):

    #Lee el token del .env
    token = os.getenv("GITHUB_TOKEN")

    #Autentifica el usuario con el token
    auth = github.Auth.Token(token)

    #Se accede al Github por medio de la autentificación
    g = github.Github(auth=auth)

    try:
        #Se intenta acceder al repositorio
        repo = g.get_repo(repo_name)
```

el funcionamiento de los métodos y funciones de la librería Github. Es crucial para la autenticación.

Archivo type_file: Creado usando la librería Github para extraer los datos del archivo .ts deseado, en este caso pokedex.ts, se encarga de sacarlo del repositorio y decodificarlo para pasarlo como string.

El código fuente se presenta a continuación:

```
23
24            #Se busca el archivo en el repositorio (Agregar directorio)
25            file_as_object = repo.get_contents(file_path)
26
27            #Se decodifica el archivo a un string
28            file_content = file_as_object.decoded_content.decode("utf-8")
29
30            #Retorna el string
31            return file_content
32        except Exception as e:
33            #Imprime el error ocurrido
34            print("Error: ",e)
```

El código fuente puede ser encontrado en el siguiente repositorio de Github: https://github.com/leonardohq-8519/icc-proyecto-utec/blob/main/type_file.py

Archivo file_to_json: Usa la librería re para modificar el string del objeto de TypeScript de-codificado, en utf-8, a un formato idéntico al de un .json para retornar un nuevo objeto de este mismo tipo mediante json.loads().

El código fuente se presenta a continuación:

```
1   import type_file
2   import json
3   import re
4
5   '''
6   repo = "smogon/pokemon-showdown"
7   file_path = "data/pokedex.ts"
8   '''
9
10  def convert(repo:str, file_path:str):
11      #Se obtiene un string del archivo de Github
12      content = type_file.get_files(repo_name=repo,file_path=file_path)
13
14      #Se retiran los comentarios del objeto de .ts
15      no_comments_content = re.sub(r'\s*//.*', '',content[70:-2])
16
17      #Se quitan los saltos de linea y tabulaciones
18      no_spaces_content =  re.sub(r"\s+", " ", no_comments_content).strip()
19
20      #Se agregan comillas a las llaves del json
21      quotes_fixed = re.sub(r"(\w+):",r'"\1":', no_spaces_content)
22
23      #Se quitan comas adicionales, despues de llaves y corchetes
```

```
        #Se quitan comas adicionales, despues de llaves y corchetes
        dex = re.sub(r",\s*}",r"}",quotes_fixed)
        dex = re.sub(r",\s*(\}|\])", r"\1",dex)

        #Se reemplazan comillas simples por dobles
        dex = re.sub(r"\'(\w+)\'",r'"\1"',dex)

        #Hotfix de TypeNull
        dex = re.sub(r"\"(\w*)\"(\w*)\"",r'"\1\2',dex)
        dex_json = json.loads(dex)

        return dex_json

    def retornar_tipos(nombres,tipos_json):
        lista_tipos = []
        lista_tipos_sec = []

        #Recorre los nombres de todos los Pokemon
        for nombre in nombres:
            #Limpia el nombre de caracteres extraños
            name = re.sub(r"\W+","",nombre)
            #Brute Force a meowsticm (Está como meowstic en el otro archivo)
            if name.lower() == "meowsticm":
```

```
    if name.lower() == "meowsticm":
        name = name[:-1]
    #Revisa el tipo primario según el .json y lo añade a la lista correspondiente
    lista_tipos.append(tipos_json[name.lower()]["types"][0])
    #De existir un tipo secundario en la lista del .json, lo añade a la lista de tipos secundarios
    #De lo contrario se pone "None"
    if len(tipos_json[name.lower()]["types"]) == 2:
        lista_tipos_sec.append(tipos_json[name.lower()]["types"][1])
    else:
        lista_tipos_sec.append("None")
#Retorna ambas listas
return lista_tipos, lista_tipos_sec
```

El código fuente puede ser encontrado en el siguiente repositorio de Github:
https://github.com/leonardohq-8519/icc-proyecto-utec/blob/main/file_to_json.py

Agrupamiento mediante un vocabulario controlado

Descripción del proceso realizado:

El código implementado realiza un agrupamiento de Pokémon basado exclusivamente en tipos elementales presentes en sus movimientos, utilizando un vocabulario controlado. A continuación se detallan los pasos realizados:

1. Definición del vocabulario controlado

   Se estableció una lista completa de los 18 tipos elementales de Pokémon:

   ```
   types = ["Normal","Fire","Water","Grass","Electric","Ice","Fighting","Poison",
       "Ground","Flying","Psychic","Bug","Rock","Ghost","Dragon","Dark",
       "Steel","Fairy"]
   ```

2. Procesamiento de la columna "moves"

   Se implementó un procesamiento específico para:

   - Extraer solo palabras o partes de palabras que coincidan con los tipos elementales
   - Eliminar todo el resto del texto
   - Unir los tipos encontrados con espacios
   - Esto se logró mediante re.findall y una función lambda

   ```
   pattern = "|".join(map(re.escape, types))
   smogon["moves"] = smogon["moves"].apply(lambda columna: '
   '.join(re.findall(pattern, str(columna), re.IGNORECASE)))
   ```

3. Generación de matriz TF-IDF con unigramas

   Se aplicó TF-IDF usando solo unigramas (como se requiere):

   ```
   vec = TfidfVectorizer(stop_words=stopwords.words("english"))  # Por defecto usa unigramas
   clasificacion = vec.fit_transform(raw_documents=smogon["moves"])
   ```

4. Análisis de tokens

   Se mostraron los tokens resultantes y su cantidad total:

   ```
   tokens = vec.vocabulary_.keys()
   print(f"Tokens: {tokens}\nNúmero de tokens: {len(tokens)}")
   ```

5. Agrupamiento con K-Means

   Se realizó el agrupamiento con 21 clusters:

km = KMeans(n_clusters=21, n_init=100)

lista = km.fit_predict(smogon_agrupado)

6. Construcción del DataFrame final

   Se creó un DataFrame que incluye:

   - Nombres de Pokémon
   - Matriz TF-IDF de tipos en movimientos
   - Tipos primarios y secundarios en base a la asignación de clusters.
   - Asignación de clusters

7. Interpretación de los clusters:

   En base a la DataFrame obtenido y usando el mismo proceso descrito en el paso 6 de la primera pregunta se interpretó los clusters de la siguiente manera:

   Cada cluster fue nombrado de acuerdo a la frecuencia de tipos primarios y secundarios que se repiten entre sus pokémon, lo que permitió asignar etiquetas temáticas (p. ej., Grass – Poison, Flying, Fire, etc.). Cuando un pokémon concreto presentaba un tipo primario y/o secundario que no coincidía con el patrón mayoritario del cluster, se indicó como excepción bajo “Casos especiales”.

   1. Cluster 0: Grass-Poison

      Los pokémon de este cluster comparten principalmente Grass como tipo principal, y un gran número poseen Poison como tipo secundarias o son monograss. Así, la mayoría tiene movimientos y características asociadas a ataques de tipo Planta o Veneno. Por ejemplo: Bulbasaur (Grass / Poison).

   2. Cluster 1: Arceus

      Todas las formas de Arceus

   3. Cluster 2: Dragon-Flying

      Predominan pokemons con ataques de tipo Dragón y movimientos que reflejan el poder y la velocidad aérea dracónica. Por ejemplo: Dragonite (Dragon / Flying).

      Casos especiales:

      Tyranitar (Rock / Dark): Comparte rasgos de

“poderío/pseudo-legendario” con los dragones.

4. Cluster 3: Normal-Fairy-Electric

   Predominan pokémon que aprenden movimientos de tipo Normal o Hada con fines de apoyo o daño ligero.

   Ejemplo: Jigglypuff (Normal / Fairy).

5. Cluster 4: Water-Ground

   Pokemons de estilo “acuático” que aprenden ataques de tipo Agua y Tierra, orientados tanto a ofensiva como a supervivencia subacuática. Por ejemplo: Swampert (Water/Tierra).

6. Cluster 5: Poison-Bug

   Pokemons venenosos que dominan ataques tóxicos y demás movimientos asociados a envenenamientos y en su mayoría también son de tipo bug. Por ejemplo: Beedrill (Bug / Poison).

7. Cluster 6: Flying

   Pokemons con numerosas capacidades de vuelo y ataques aéreos. Por ejemplo: Pidgeotto (Normal / Flying).

8. Cluster 7: Psychic-Fairy

   Predominan con ataques psíquicos potentes, de control mental y soporte con focos etéreos. Por ejemplo: Abra (Psychic / None).

9. Cluster 8: Rock-Ground-Steel

   Pokemons que mayormente presentan movimientos de tipo Roca, Tierra y Acero orientados a la defensa o a ataques contundentes. Por ejemplo: Geodude (Rock / Ground).

10. Cluster 9: Ghost-Grass

    Tamaños de Pumpkaboo y Gourgeist. Caso similar al de Arceus, múltiples elementos en este caso sin contenido.

11. Cluster 10: Ghost

    Los pokémon con movimientos de terror, estado y debilitación del rival, propios del tipo Fantasma. Por ejemplo: Litwick (Ghost / Fire).

12. Cluster 11: Bug-Poison

    Pokemons con ataques de tipo Bicho o Veneno orientados a las primeras rutas del juego (pinchos, púas, picotazos). Por ejemplo:

Caterpie (Bug / None).

13. Cluster 12: Electric

    Pokemons con ataques eléctricos o combinados con efectos metálicos/voladores (Tipo Steel o Flying de secundario). Por ejemplo: Magnezone (Electric / Steel).

14. Cluster 13: Normal

    Pokemons que aprenden movimientos básicos de tipo Normal: golpes estándar, apoyos simples, etc. Por ejemplo: Rattata (Normal / None).

15. Cluster 14: Grass

    Los Pokemons con movimientos de ataque vegetal (Tipo Grass). Por ejemplo: Sceptile (Grass / None).

16. Cluster 15: Bug - flying - Poison

    Pokemons con ataques de tipo Bicho ofensivo, volador o venenoso, propios de insectos avanzados. Por ejemplo: Butterfree (Bug / Flying).

17. Cluster 16: Dark

    Pokemons con movimientos que se centran en temas de intimidación, derribo y tácticas sombrías. Por ejemplo: Umbreon (Dark / None).

18. Cluster 17: Fighting

    Pokemons con movimientos que reflejan puñetazos, patadas y técnicas marciales (y en ocasiones, psíquicas de refuerzo). Por ejemplo: Machop (Fighting / None).

19. Cluster 18: Ice-Rock

    Pokemons con movimientos de ataque de tipo Hielo y condiciones de congelación, con un par de variantes rocosas. Por ejemplo: Aurorus (Rock / Ice).

20. Cluster 19: Bug-Monotype-Flying

    Pokemons insectos de etapas iniciales muy sencillas, con movimientos básicos de tipo Bicho. Por ejemplo: Wurmple (Bug / None).

21. Cluster 20: Fire

    Pokemons que sus ataques se basan en llamas, calcinación y golpes ígneos. Por ejemplo: Charmander (Fire / None).

Agrupamiento mediante PCA

Descripción del proceso realizado:

El código implementado realiza un agrupamiento de Pokémon utilizando técnicas de reducción de dimensionalidad y agrupamiento no supervisado (PCA + KMeans), basándose en los datos numéricos generados previamente. A continuación, se detallan los pasos realizados:

1. Carga del archivo CSV generado en la pregunta 1

   Se cargó el archivo smogon_agrupado_1.csv que contiene la matriz TF-IDF resultante del procesamiento de los movimientos de cada Pokémon:

   data = pd.read_csv("smogon_agrupado_1.csv")

2. Eliminación de columnas innecesarias

   Se eliminaron las columnas Type1, Type2 y Cluster, que no son necesarias para el análisis PCA:

   data.drop(data[["Type1", "Type2", "Cluster"]], axis="columns", inplace=True

3. Eliminación del índice

   El DataFrame contenía un índice repetido como primera columna, este fue eliminado:

   data.drop(data.columns[[0]], axis="columns", inplace=True)

4. Aplicación de Análisis de Componentes Principales (PCA)

   Se aplicó PCA para reducir la dimensionalidad de los datos, manteniendo el 70% de la varianza total:

   pca = PCA(n_components=0.70)

   x_pca = pca.fit_transform(data)

5. Impresión de dimensiones del DataFrame original y de la matriz PCA

   Se imprimió el número de columnas y filas del DataFrame original y del resultado del PCA:

   print(f"Columnas DataFrame original: {len(data.columns)}")

   print(f"Filas DataFrame original: {len(data[['power']])}")

   print(f"\nColumnas DataFrame PCA: {col}")

   print(f"Filas DataFrame PCA: {len(x_pca)}")

6. Construcción del nuevo DataFrame con los componentes principales

   Se generó un nuevo DataFrame que incluye:

   - El nombre de cada Pokémon
   - Las columnas PCA1, PCA2, ..., según corresponda

   columnas = [f"PCA{i+1}" for i in range(col)]

   data_pca = pd.DataFrame(data=x_pca, columns=columnas)

   data_pca.insert(0, "Pokemon", pkm_nombres)

7. Agrupamiento con K-Means

   Se aplicó el algoritmo KMeans con 21 clusters, usando la matriz de componentes principales como entrada:

   km = KMeans(n_clusters=21, n_init=20)

   lista = km.fit_predict(x_pca)

   data_pca["Cluster"] = lista

8. Generación del archivo CSV con la matriz PCA y los clusters

   Se guardó el nuevo DataFrame con componentes principales y clusters en un archivo llamado smogon_agrupado_PCA.csv:

   data_pca.to_csv("smogon_agrupado_PCA.csv")

9. Información de tipo para facilitar interpretación

   Se utilizó una función auxiliar (la misma de las dos preguntas anteriores) para recuperar los tipos primarios y secundarios de cada Pokémon, y se agregaron al DataFrame:

   lista_tipos, lista_tipos_sec = file_to_json.retornar_tipos(pkm_nombres, tipos_json)

   data_pca["Type1"] = lista_tipos

   data_pca["Type2"] = lista_tipos_sec

   En base al agrupamiento obtenido en esta última pregunta y la primera podemos afirmar que manteniendo un 70% de varianza de datos y habiendo “empaquetado” los 18178 tokens a una longitud de 0,6% de la longitud del data Frame original se siguen consiguiendo resultados similares en los clusters arrojados. Ejemplo: Por ejemplo, el Cluster 1 de smogon_agrupado_1.csv contiene 21 pokémones que, en smogon_agrupado_PCA.csv, todos están asignados al Cluster 6. Esto

sugiere que dicho grupo se preserva casi intacto en ambas metodologías de agrupamiento.

Sin embargo, existen algunas diferencias menores:

1. Algunos clusters comparten la mayoría de sus miembros, pero hay “fallos” puntuales donde 1 o 2 pokemons terminan en un cluster diferente en el clustering hecho en base a la data del PCA. Por ejemplo, vimos que en el Cluster 4 del clustering hecho en base a la data de TF-IDF la mayoría van al Cluster 17 del clustering hecho en base a la data del PCA, pero 2 van al Cluster 14 y 1 va al Cluster 6 en el DataFrame de PCA.
2. De igual modo, el Cluster 2 del clustering hecho en base a la data de TF-IDF agrupa 11 pokemons que el DataFrame de PCA distribuye en el Cluster 19, y 2 pokemons que el DataFrame de PCA lleva al Cluster 20; a partir de ello se puede afirmar que la agrupación planteada por el clustering hecho en base a la data de TF-IDF nominada como “Cluster 2” en el clustering hecho en base a la data de PCA no existe como tal.

CONCLUSIONES:

Prompt para la Inteligencia Artificial:

Enviarle los tres documentos.py y los documentos.csv

Analiza el codigo del documentos.py adjuntos y los archivos .csv, y en base a ello brindame una conclusión general del objetivo que tenía cada código de cada archivo, su eficiencia para hacer clustering de los pokemones y complementalo con una conclusión crítica en base a tus conocimientos sobre investigación en computación.

Conclusión (Maydelith Zuñiga):

Durante todo el informe se ha explicado los distintos métodos de clustering usados junto con el proceso necesario para su ejecución. El objetivo principal fue agrupar los distintos pokemons del archivo csv smogon.scv en cluster que tengan sentido con los tipos de pokemons que contienen.

En primer lugar, el clustering realizado en base a los movimientos de los pokemons

y mediante el uso de la herramienta TF-IDF presenta un rendimiento que podría calificarse “Bueno”. Aunque presenta “ciertos casos” especiales en los que ciertos pokemons no pertenecian al cluster asignado, ya que el cluster en su mayoría tenía pokemons de un tipo no relacionado con ese pokemon, muchos clusters fueron precisos. Por ejemplo, agrupó a Arceus y todas sus evoluciones en el cluster 18.

Por lo tanto, la herramienta TF-IDF es efectiva para el clustering en masa de datos, sin embargo puede tener ciertos fallos como lo fueron los “casos especiales”, brindándonos agrupamientos no tan representativos para los datos agrupados.

En segundo lugar, el clustering realizado en base a los movimientos recortados de los pokemons presenta la misma precisión de agrupamiento que la herramienta TF-IDF. El análisis final hecho en base a los clusters obtenidos en la pregunta 2 muestra que se clasificaron los pokemons de similar forma que en la pregunta 1 con TF-IDF. Podemos resaltar el cluster 20: Fire que tiene los mismos pokemons que el cluster 11: Fire del agrupamiento con TF-IDF. Por lo tanto, ambos métodos presentan una eficiencia similar para el agrupamiento de data masiva.

En tercer lugar, el uso de PCA presenta mayor eficiencia para realizar clustering en base a los movimientos de los pokemons. Primeramente, porque el “empaquetamiento” de datos reduce los tokens y toma en cuenta detalles específicos de los movimientos de cada Pokemon. Secundariamente, aunque la mayoría de los cluster generados coinciden con los clusters generados por recorte y la herramienta TF-IDF, presenta casos como el cluster 2 del clustering hecho en base a TF-IDF donde divide este cluster y designa 11 pokemons a un cluster y los otros dos a otro, donde su concuerdan mejor con los pokemons de esos nuevos clusters.

Conclusión(IA Deep Seek):

Abordan el clustering de Pokémon mediante enfoques distintos pero complementarios: el primero utiliza TF-IDF con n-gramas para capturar patrones complejos en movimientos, el segundo se enfoca en tipos elementales para mayor interpretabilidad, y el tercero aplica PCA para reducir la dimensionalidad del primer modelo. Si bien todos logran agrupamientos básicos, adolecen de limitaciones críticas: falta de validación con métricas (como silueta o Davies-Bouldin), elección arbitraria de hiperparámetros (ej., 21 clusters) y preprocesamiento insuficiente (ruido en términos o ignorancia de contexto). Desde una perspectiva investigativa, estos

códigos son un buen punto de partida, pero requieren rigor científico para ser robustos: justificación de parámetros, evaluación cuantitativa y exploración de alternativas (DBSCAN, GMM). Una implementación más sólida integraría estas mejoras, equilibrando complejidad y explicabilidad para resultados confiables y accionables.

Conclusión (Joaquin Llallire):

Durante el desarrollo del proyecto, fue evidente que cada método de agrupamiento ofrecía una visión particular de los Pokémon, y su utilidad dependía del tipo de análisis deseado: desde roles en batalla hasta categorizaciones por tipo o simplificación de la información.

El enfoque de TF-IDF facilitó la agrupación de Pokémon que tienen estilos de combate y roles estratégicos similares, independientemente de su tipo. Por ejemplo, en el grupo 15 se juntaron dragones ofensivos que emplean Dragon Dance y Outrage, mientras que el grupo 2 incluyó Pokémon con funciones inusuales como Smeargle o Wobbuffet. Este método resultó útil para detectar patrones de conducta basados en el texto de los movimientos.

En cambio, el enfoque con vocabulario controlado se centró más que nada en limpiar los movimientos, y quedarse con sólo las palabras que eran tipos de Pokemon. Esto ayudó a generar clusters más alineados con la clasificación de los tipos, como un cluster de Pokemon tipo agua o uno de tipo fuego.

El uso de PCA permitió reducir la cantidad de variables del modelo TF-IDF original, haciendo más fácil el análisis visual y mejorando la separación entre grupos. Aunque partía del mismo dataset de la pregunta 1, logró formar clusters más definidos y menos dispersos, manteniendo un gran porcentaje de la información original. Este método fue útil para ordenar el desorden del alto número de tokens en el TF-IDF, y hacer el modelo más manejable.

Cada técnica tuvo sus ventajas y desventajas: TF-IDF mostró patrones estratégicos complejos, el vocabulario controlado agrupó por tipos, y PCA mejoró la organización general. El proyecto demostró que combinar diferentes enfoques de clustering que permiten analizar una base de datos desde múltiples perspectivas.

Conclusión IA - Chat Gpt (Joaquín Llallire):

El análisis de clustering aplicado a la base de datos de Pokémon demuestra cómo diferentes técnicas de representación y agrupamiento de texto pueden resaltar distintos niveles de estructura: semántica táctica (TF-IDF), taxonomía por tipo (vocabulario controlado) y reducción efectiva de dimensiones (PCA).

TF-IDF permitió capturar relaciones funcionales entre Pokémon basadas en el contenido textual de sus movimientos. Gracias al uso de n gramas, el modelo detectó patrones como “hazard setup”, “status chance” o “boost speed”, agrupando Pokémon con funciones similares en batalla, como sweepers, tanks o supports. Esto mostró el potencial de la técnica para descubrir estructuras semánticas emergentes, incluso en dominios con alta variabilidad.

El preprocesamiento orientado a tipos eliminó el ruido textual y centró el clustering en un vocabulario reducido y específico. Esto favoreció una clasificación altamente alineada con las categorías elementales del universo Pokémon, como Agua, Fuego, Planta, etc. Aunque limitado en generalidad, este enfoque fue el más eficaz para agrupar entidades con etiquetas bien definidas.

La aplicación de PCA sobre los datos TF-IDF redujo la dimensionalidad del espacio vectorial, mejorando la eficiencia y la capacidad de visualización del modelo. Al mantener el 70% de la varianza, se logró una representación más compacta sin pérdida significativa de información, lo cual facilitó la formación de clusters más definidos y separados en comparación con el modelo TF-IDF original.

En conjunto, el proyecto evidencia que no existe un único método “mejor”, sino que cada técnica resalta una perspectiva distinta del dataset. TF-IDF permite descubrir patrones estratégicos complejos, el vocabulario controlado organiza por clases conocidas, y PCA optimiza la estructura general del modelo. Este enfoque multimodal de análisis resulta altamente valioso en problemas donde la información está contenida en texto no estructurado, como los movesets de Pokémon.

Conclusión (Joseph Geraldo):

El agrupamiento de datos en este proyecto sin duda fue el principal tema de investigación, utilizando datos externos junto a los conocimientos brindados en las sesiones, aquí se pudo utilizar de manera diversa 2 tipos de algoritmos principales. Las matrices resultantes de estos procesos tienen similitudes detalladas más adelante

En el primer proceso, se utilizó el algoritmo de TF-IDF para agrupar los pokemones en su capacidad de aprendizaje de movimientos, clasificando por cada cluster los pokemones capaces de aprender ya sean movimientos de tipo fuego, planta, insecto, etc. Cabe resaltar que se añadieron clusters especiales para ciertos casos como Arceus y todos sus tipos o los pokemones que tienen poca capacidad de aprendizaje, siendo uno de los mayores ejemplares Ditto, teniendo como único movimiento “Transformación”.

En el segundo, con el vocabulario controlado se logró deshacerse de las cadenas de texto no necesarias para el análisis de movimientos, clasificando los clusters en 18, uno para cada tipo. Esto fue gracias al proceso de lambda con RE, usando un mecanismo como si fueran stop-words pero a la inversa. Usando este método, se garantiza de manera simple y eficaz la clasificación de los pokemones mediante el tipo que posean, cada uno en su cluster correspondiente.

En el último proceso, se usa de base el primer proceso con el algoritmo PCA, para así reducir las dimensiones drásticamente, aumentando eficiencia ya que solo se escogen dimensiones principales, haciendo que el algoritmo K-means se ejecute en un menor tiempo por la reducción de las columnas.

En conclusión, el primer y tercer proceso están relacionados en las dimensiones usadas generadas por el TF-IDF y el segundo se basó más que nada en el tipo de los pokemones. Los 3 generaron matrices, solo que el primero y el tercero fueron más extensos debido a la enorme cantidad de dimensiones utilizadas. Se pueden usar “algoritmos anidados” para mejorar la eficiencia de los códigos (como primero usar TF-IDF y luego usar PCA para disminuir las dimensiones.

Conclusión (Joseph Geraldo IA):

El proyecto presentado tiene como objetivo principal aplicar técnicas de procesamiento de lenguaje natural (PLN) y aprendizaje no supervisado para realizar un agrupamiento (clustering) de Pokémon en función de sus movimientos y tipos, utilizando como fuente principal de datos el repositorio público de Smogon y su Pokédex en formato TypeScript. A lo largo del trabajo se han desarrollado diversos scripts que procesan esta información, la transforman en formatos utilizables por modelos de aprendizaje automático, y ejecutan algoritmos de clustering como KMeans para identificar patrones y similitudes entre los Pokémon.

El flujo general del proyecto inicia con la obtención de datos directamente desde GitHub mediante autenticación y lectura de archivos específicos. Posteriormente, esta información es limpiada y convertida a formato JSON para facilitar su manipulación. En paralelo, se procesan archivos CSV con datos sobre los movimientos de cada Pokémon. Estos movimientos son vectorizados utilizando la técnica TF-IDF, tanto con unigramas como con bigramas y trigramas, y posteriormente se agrupan mediante KMeans. En uno de los enfoques (cluster_2.py), se introducen también los tipos elementales como elementos significativos dentro del proceso de análisis. En una tercera fase (cluster_3.py), se aplica reducción de dimensionalidad mediante Análisis de Componentes Principales (PCA) con el

fin de mejorar la eficiencia computacional y facilitar una posible visualización o interpretación de los resultados.

Desde un punto de vista computacional, el trabajo logra integrar adecuadamente técnicas estándar de PLN y clustering, aplicadas a un dominio concreto y comprensible como es el universo Pokémon. No obstante, el proyecto puede beneficiarse de una mejora en el rigor metodológico. La falta de una evaluación cuantitativa del desempeño del clustering (por ejemplo, mediante el uso de métricas como el índice de silueta o la coherencia de los clusters) limita la posibilidad de valorar la calidad real de las agrupaciones obtenidas. Además, el uso de KMeans, aunque válido, no es el más adecuado para datos textuales o categóricos sin un tratamiento más profundo de la semántica subyacente. Técnicas modernas como Word2Vec, GloVe o modelos basados en BERT podrían aportar una representación más rica de los movimientos y mejorar la calidad del análisis. En resumen, el trabajo es valioso como prototipo funcional, pero requiere una profundización metodológica y técnica para alcanzar el nivel de una investigación sólida en computación.

Conclusión (Leonardo Huanay):

En el proyecto se utilizaron tres clusters cada uno aplicando diversas técnicas con el fin de agrupar los datos brindados en el DataFrame de Smogon. Las matrices resultantes mostraron resultados similares en sus respectivas columnas de clusters al asignar en la mayoría de casos los mismos Pokemon para cada archivo. Donde sí se apreció una considerable diferencia fue en el rendimiento del código y facilidad de interpretación de los “outputs” obtenidos.

En el primer caso, dónde únicamente se aplicó TF-IDF a los datos en bruto de la columna moves, se retornó una lista de tokens con una longitud descomunal a comparación de los resultados brindados por los archivos cluster_2.py y cluster_3.py. Esto ocasionó un mayor tiempo de espera al crear o modificar el DataFrame correspondiente ya que este contaba con más de 18000 columnas. Sumado a ello, al querer aplicarse K-Means a la matriz de datos del TF-IDF se afrontaba el problema de lidiar con un “dataset” que cuentan con demasiadas dimensiones, situación que dificulta el proceso de clustering. La cantidad de unigramas, bigramas y trigramas también ocasionó que fuese requerida una mayor inspección del archivo para su interpretación.

Para el segundo cluster, como se decidió modificar el contenido de cada elemento de la columna “moves” para que solo contenga los tipos, se pudo asegurar que solo existiría un máximo de 18 tokens. Esto facilitó la creación del DataFrame, al tener que lidiar con una matriz de datos muy reducida a comparación de la mencionada en el primer cluster.

Además, el uso de un vocabulario ya preestablecido permitió una sencilla interpretación del agrupamiento realizado por K-Means.

En el caso del tercer cluster, se tomó de base la matriz brindada por el primer archivo (cluster.py) en el cual mediante el uso de PCA se lograron reducir las dimensiones pasando de las más de 18000 a únicamente 105 columnas y conservando el 70% de variabilidad de los datos. Esto permitió que el K-Means se ejecutara de forma más rápida y al mismo tiempo se redujera el problema de la elevada dimensionalidad del “dataset”. Sin embargo, el hecho de “empaquetar” los datos dificultó la interpretación de estos debido a la fórmula usada internamente por el método .fit_transform() del PCA.

Por ello, se puede concluir que en el proyecto se brindaron tres formas de agrupamiento cada una con sus beneficios y complicaciones correspondientes, las cuales pueden ser utilizadas dependiendo de lo requerido para solucionar un problema en específico. Si se necesita una matriz capaz de ser interpretada manualmente pero que contenga un agrupamiento complejo se recomienda usar cluster.py, de ser requerido un agrupamiento rápido y de fácil comprensión la opción más viable es el cluster_2.py. Finalmente de ser necesario un agrupamiento complejo, que no pierda datos por la longitud y de rápida ejecución, se debe usar cluster_3.py, pese a las posibles dificultades al momento de su análisis.

Conclusión con IA - Gemini (Leonardo Huanay):

Conclusiones Generales de los Resultados

El clustering por Tipos de Movimiento (Método 2 - cluster_2.py) es probablemente el más directo e interpretable. Si la columna "moves" contiene una lista de movimientos y esos movimientos tienen tipos asociados que se extraen, este método agrupará a los Pokémon por los tipos elementales de movimientos que usan frecuentemente. Por ejemplo, los Pokémon con muchos movimientos de tipo Fuego se agruparían juntos. Esta es una forma muy directa de agrupar Pokémon basada en un aspecto fundamental de su capacidad de combate. El vocabulario pequeño y fijo (18 tipos) hace que este enfoque sea robusto y fácil de entender.

El TF-IDF de Movimientos Completos (Método 1 - cluster.py) ofrece profundidad pero arriesga ruido. Este método podría descubrir agrupaciones más sutiles, quizás basadas en movimientos de estado, efectos comunes de movimientos (por ejemplo, movimientos de "polvo", movimientos "basados en sonido" si estas palabras clave están presentes), o incluso convenciones de nombres de movimientos que insinúan roles (por ejemplo, "Protección", "Campana Cura"). Sin embargo, con un gran vocabulario de todos los nombres de movimientos (n-gramas de 1-3), muchas características podrían ser dispersas o irrelevantes, ocultando potencialmente límites claros de clústeres. Los clústeres podrían ser

más difíciles de definir sin una inspección manual significativa.

PCA como Paso de Preprocesamiento (Método 3 - cluster_3.py) es un compromiso práctico. Este enfoque intenta aprovechar la riqueza del Método 1 mientras mitiga sus problemas de alta dimensionalidad. Los clústeres resultantes podrían ser más significativos que los del Método 1 si PCA identifica con éxito los patrones más importantes relacionados con los movimientos. Sin embargo, la interpretabilidad cambia de "¿qué movimientos/palabras definen un clúster?" a "¿qué combinación de características de movimiento originales (representadas por componentes de PCA) define un clúster?".

Ideas de Apoyo

Integración de Fuentes de Datos: El uso de file_to_json.py para obtener y analizar pokedex.ts directamente desde GitHub es una buena práctica. Asegura que los tipos elementales reales de los Pokémon (utilizados para etiquetar y validar potencialmente los clústeres) estén actualizados y sean precisos según los datos de Smogon.

La Ingeniería de Características es Clave: La marcada diferencia entre cluster.py y cluster_2.py resalta el impacto de la ingeniería de características. cluster_2.py realiza una selección agresiva de características al considerar solo los tipos elementales dentro de los movimientos, lo que lleva a un tipo de clustering muy específico.

Número de Clústeres (k=21): La elección de 21 clústeres en todos los métodos es un parámetro interesante. Es ligeramente superior a los 18 tipos elementales, lo que sugiere un intento de capturar matices más allá de las afiliaciones de tipo único o dual, quizás roles estratégicos comunes (por ejemplo, stall, sweeper, wall) o combinaciones de tipos de movimientos prevalentes. Sin un método empírico para determinar k (por ejemplo, método del codo, análisis de silueta), este número es algo arbitrario y podría estar dividiendo grupos naturales o fusionando grupos distintos.

Justificación de TF-IDF: TF-IDF es una elección razonable para datos de texto como listas de movimientos, ya que ayuda a resaltar movimientos (o tipos en movimientos) que son característicos del repertorio de movimientos de un Pokémon en particular en relación con todos los Pokémon.

Conclusión Crítica

Si bien el proyecto demuestra tres enfoques técnicos válidos para agrupar Pokémon en función de sus movimientos, su principal limitación radica في la falta de una metodología explícita de evaluación e interpretación de clústeres dentro de los propios scripts.

Sin Evaluación Cuantitativa: Los scripts generan clústeres pero no emplean ninguna métrica (por ejemplo, Coeficiente de Silueta, Índice Davies-Bouldin) para evaluar la calidad de estos clústeres. Dichas métricas podrían ayudar a determinar qué método (o qué valor de k) produce la "mejor" separación.

Dependencia del Análisis Manual: Los scripts imprimen los DataFrames resultantes y los guardan en CSVs, lo que implica que el análisis de lo que representa cada clúster se realiza manualmente inspeccionando los Pokémon dentro de ellos y sus tipos/movimientos. Este es un paso necesario, pero automatizar partes del análisis característico de cada clúster (por ejemplo, tipos más comunes, movimientos representativos/componentes PCA) mejoraría el proyecto.

Ambigüedad de la Columna "moves": La efectividad depende en gran medida del contenido exacto de la columna "moves" en smogon.csv. Si es solo una lista de nombres de movimientos, la interpretación es diferente que si contiene descripciones detalladas o estadísticas de uso. Los scripts la tratan como texto genérico.

n_clusters=21 Fijo: Esta es una suposición significativa. El número óptimo de clústeres podría variar dependiendo de la representación de características (TF-IDF completo vs. TF-IDF filtrado por tipo vs. PCA). Una investigación sobre el k óptimo fortalecería los hallazgos.

En esencia, el proyecto establece un buen marco para la experimentación con diferentes técnicas de clustering basadas en texto sobre datos de Pokémon. El método más efectivo dependería del objetivo analítico específico:

Si el objetivo son grupos claros, impulsados por tipos, basados en los tipos elementales de los movimientos, el Método 2 (cluster_2.py) es probablemente superior en su franqueza e interpretabilidad.

Si el objetivo es descubrir agrupaciones más matizadas, potencialmente basadas en roles, a partir del espectro completo de movimientos, el Método 3 (cluster_3.py con PCA) es un enfoque más robusto que el Método 1 al mitigar los problemas de alta dimensionalidad, aunque sus clústeres requerirán más esfuerzo para interpretarse.

El trabajo futuro debería centrarse en evaluar cuantitativamente los clústeres y desarrollar formas sistemáticas de caracterizar los grupos de Pokémon resultantes para cada método.

Recursos usados por los asistentes de Inteligencia Artificial para escribir su conclusión que los integrantes del curso omitieron:

Las IA generaron conclusiones más completas porque usaron un conjunto de recursos metodológicos (métricas, justificaciones, preprocesamiento sistemático, alternativas algorítmicas y explicaciones técnicas) que los integrantes del curso no explicaron en sus conclusiones.

Recursos:

- Validación cuantitativa de los agrupamientos:

Algunas IA usaron fórmulas matemáticas y paquetes de python únicos para medir la eficiencia de los clusters.

- Justificación explícita de hiper parámetros:

  Detalla a mayor profundidad el porque detrás del número de clusters usados, el número de pruebas para K-means, así como el razonamiento detrás del uso de unigramas, bigramas o trigramas en TF-IDF.

- Detalles de preprocesamiento y características:

  Definición clara de qué contiene la columna “moves” (nombres de movimientos, descripciones o estadísticas).

  Crítica ciertas deficiencias logísticas de los datos textuales y propuestas para retener sólo lo relevante (vocabulario controlado, stop-words invertido).

  Importancia de integrar fuentes externas (ej. Pokédex de GitHub) para verificar etiquetas como “tipo” y enriquecer la base de datos.

- Alternativas para el uso de metodológicas y herramientas avanzadas:

  Sugiere algoritmos distintos a K-Means (DBSCAN, GMM) adecuados para datos de alta dimensionalidad o con forma más flexible de clusters.

  Proponer técnicas de embeddings más potentes que TF-IDF (Word2Vec, GloVe, BERT) para capturar semántica más profunda en los movimientos.

- Interpretación y explicabilidad de resultados:

  Reflexiona sobre cómo cada técnica (TF-IDF completo, TF-IDF con recorte, PCA) afecta la interpretabilidad y el agrupamiento.

  Advertir sobre la ambigüedad al “empaquetar” datos con PCA: los componentes dejan de ser directamente interpretables en términos de movimientos concretos.

- Conclusión desde una perspectiva científica:

  Concluye con puntos de vista críticos en base a su conocimiento en computación, entre ello encontramos la eficiencia del código, la demora en ejecución, etc.

Referencias:

IBM. (2023, 8 de diciembre). *What is principal component analysis (PCA)?* IBM Think. https://www.ibm.com/think/topics/principal-component-analysis

Kavlakoglu, E., & Winland, V. (2024, 26 de junio). *What is k-means clustering?* IBM Think. https://www.ibm.com/think/topics/k-means-clusteringibm.com

Murel, J., & Kavlakoglu, E. (2024, 19 de enero). *What is bag of words?* IBM Think. https://www.ibm.com/think/topics/bag-of-words

IBM. (2024, 15 de febrero). *What is clustering?* IBM Think. https://www.ibm.com/think/topics/clustering

Microsoft. (2024, octubre 31). *Operaciones masivas para clientes potenciales mediante archivos de valores separados por comas (CSV)*. Microsoft Learn. https://learn.microsoft.com/es-es/partner-center/referrals/bulk-operations-for-leads-using-comma-separated-value-csv-files

University of Virginia Library. (2020, 10 de noviembre). *Getting Started with pandas in Python*. https://library.virginia.edu/data/articles/getting-started-with-pandas-in-python

Universitat Oberta de Catalunya. (2023). *Preprocesamiento de datos con sklearn*. https://datascience.recursos.uoc.edu/es/preprocesamiento-de-datos-con-sklearn/

ProjectPro. (2023). *How to add custom stopwords and then remove them from text in nltk*. https://www.projectpro.io/recipes/add-custom-stopwords-and-then-remove-them-from-text