Universidad Nacional de Rosario

Facultad de Ciencias Económicas y Estadísticas

Procesamiento del Lenguaje Natural

Trabajo práctico N°1: Clasificador de Recomendaciones Recreativas utilizando Procesamiento de Lenguaje Natural

Rodríguez y Barros, Francisco: R-4559/4

Slepoy, David: S-5782/7

[Repositorio de GitHub]

Profesores:

Geary, Alan

Manson, Juan Pablo

Fecha de Entrega: 06/10/24

Introducción.....2
Objetivo:.....2
Resumen:.....2
Metodología:.....3
Desarrollo:.....4
Clasificación del Estado de Ánimo.....4
Ingreso de Preferencias.....5
Web Scraping.....6
Búsqueda de Opciones y Recomendación.....6
Integración y Ejecución.....8
Resultados.....9
Conclusión.....11

Introducción.

Este trabajo se centra en el desarrollo de un sistema de recomendación que sugiere actividades recreativas, como ver una película, jugar un juego de mesa o leer un libro, en función del estado de ánimo del usuario y sus preferencias.
Este proyecto utiliza técnicas de procesamiento de lenguaje natural (NLP) para analizar el estado de ánimo del usuario y sus intereses, permitiendo así la generación de recomendaciones que se alineen con sus preferencias individuales.

Objetivo:

Desarrollar un programa de Procesamiento de Lenguaje Natural que, según el estado de ánimo del usuario, recomiende actividades como ver una película, jugar un juego de mesa o leer un libro.

Objetivos Específicos:

1. Desarrollar un clasificador que categorice el estado de ánimo del usuario en "Alegre", "Melancólico" o "Ni fu ni fa".
2. Permitir al usuario ingresar una frase que describa sus preferencias sobre qué actividad le gustaría realizar.
3. Comparar la frase ingresada con descripciones en tres fuentes de datos: una de películas, otra de juegos de mesa y una última de libros.
4. Generar recomendaciones personalizadas basadas en la clasificación del estado de ánimo y las preferencias ingresadas por el usuario.

Resumen:

Desarrollamos un clasificador de estados de ánimo utilizando un modelo de regresión logística, que fue entrenado generando embeddings a partir de frases ingresadas mediante el modelo pre entrenado SentenceTransformer. Una vez clasificado el estado de ánimo, se solicitó al usuario un prompt sobre su preferencia de actividad.

El sistema combina la entrada del estado de ánimo y la preferencia del usuario para generar embeddings que se comparan con registros en tres bases de datos: una de películas, otra de juegos de mesa y una de libros. Para generar ésta última, se realizó un proceso de web scraping utilizando Python junto con la librería BeautifulSoup para compilar información sobre los 1000 libros más populares del Proyecto Gutenberg.

Los resultados del sistema de recomendación fueron buenos, aunque hay espacio para mejorar la precisión de las sugerencias. Este trabajo muestra cómo usar técnicas de procesamiento de lenguaje natural y aprendizaje automático puede ayudar a personalizar las recomendaciones de actividades.

Metodología:

Fuentes de Datos Utilizadas:

- **bgg_database.csv**: Base de datos de juegos de mesa que contiene información sobre títulos, descripciones y categorías.
- **IMDB-Movie-Data.csv**: Base de datos de películas con detalles sobre títulos, descripciones, géneros y calificaciones.
- **Libros del Proyecto Gutenberg**: Generada mediante web scraping, esta base de datos incluye información sobre los 1000 libros más populares, con detalles como títulos, autores y descripciones.

Métodos y Técnicas Utilizados:

- **Clasificación de Estado de Ánimo**: Se implementó un clasificador utilizando regresión logística, entrenado con embeddings generados a partir de textos mediante SentenceTransformer.
- **Embeddings**: Para convertir las descripciones de películas, juegos y libros en vectores que pueden ser comparados con las entradas del usuario, utilizamos un modelo multilingüe preentrenado de SentenceTransformers. Este modelo permite generar representaciones vectoriales de alta calidad para textos en múltiples idiomas, facilitando así la comparación semántica.
- **Web Scraping**: Se usó BeautifulSoup para extraer datos de la página del Proyecto Gutenberg y crear la base de datos de libros.

Pasos Seguidos para Cumplir los Objetivos:

- **Recolección de Datos**: Se recopilaron datos de las tres fuentes mencionadas. Para la base de datos de libros, se realizó un proceso de web scraping utilizando BeautifulSoup para extraer información sobre los 1000 libros más populares del Proyecto Gutenberg.
- **Preprocesamiento**: Se generaron embeddings para las descripciones de las actividades de cada fuente. Esto con el fin de facilitar la tarea de comparación y recomendación.
- **Desarrollo del Clasificador**: Se implementó y entrenó el clasificador de estado de ánimo utilizando un modelo de regresión logística. Para esto, se utilizó un dataset que fue creado específicamente para el entrenamiento, el cual incluía frases representativas clasificadas en las categorías "Alegre", "Melancólico" y "Ni fu ni fa".
- **Interacción con el Usuario**: Solicitar el estado de ánimo y la preferencia de actividad.
- **Generación de Recomendaciones**: Comparación de entradas del usuario con los embeddings de las bases de datos para ofrecer sugerencias personalizadas.

Desarrollo:

El desarrollo del trabajo práctico se llevó a cabo en varias etapas. En primer lugar, se recopiló y preparó la información necesaria, creando listas de frases representativas para cada estado de ánimo y organizando los datos sobre juegos de mesa, películas y libros en DataFrames. En la imagen, se observa cómo se define la clase Recomendador, que es fundamental para la implementación de los métodos que facilitan la recomendación basada en el estado de ánimo del usuario. Esta clase recibe varios parámetros, incluyendo el modelo de regresión logística, el label_encoder, y los DataFrames de cada tipo de recomendación.

```
class Recomendador:
    def __init__(self, modelo, clf, label_encoder, df_juegos_de_mesa, df_peliculas, df_libros):
        self.modelo = modelo
        self.clf = clf
        self.label_encoder = label_encoder
        self.df_juegos_de_mesa = df_juegos_de_mesa
        self.df_peliculas = df_peliculas
        self.df_libros = df_libros
```

Clasificación del Estado de Ánimo.

Entrenamos un modelo de regresión logística utilizando un conjunto de datos específico que contenía frases clasificadas en diferentes estados de ánimo. A continuación se muestra el código para el entrenamiento del modelo:

```
# Cargamos el modelo preentrenado SentenceTransformer para generar embeddings a partir de texto en múltiples idiomas
model = SentenceTransformer('sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2')
```

```
embeddings_train = model.encode(df["frase"].tolist())

# Convertir las etiquetas de texto a numericas
label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(df["estado_animo"])

# Entrenar el modelo con Regresión Logística
clf = LogisticRegression(random_state=42, max_iter=200)
clf.fit(embeddings_train, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=200, random_state=42)
```

Para clasificar el estado de ánimo del usuario, se implementó una función que procesa la frase ingresada, genera embeddings y utiliza un clasificador entrenado. A continuación se

detalla el código utilizado para llevar a cabo esta tarea:

```python
def solicitar_input(self):
    """
    Solicita la frase del usuario para clasificar el estado de ánimo.
    """
    frase_usuario = input("Ingresa una frase para clasificar tu estado de ánimo: ")
    return frase_usuario

def procesar_estado_animo(self, frase_usuario):
    """
    Convierte la frase del usuario en embeddings y predice el estado de ánimo.
    """
    # Generar embeddings para la frase del usuario
    embedding_frase = self.modelo.encode([frase_usuario])

    # Predecir el estado de ánimo
    estado_animo_predicho = self.clf.predict(embedding_frase)

    # Convertir la predicción numérica de vuelta a texto
    estado_animo_predicho_texto = self.label_encoder.inverse_transform(estado_animo_predicho)

    print(f"Su estado de ánimo es: {estado_animo_predicho_texto[0]}")
    return estado_animo_predicho_texto[0]
```

Ingreso de Preferencias.

Una vez determinado el estado de ánimo del usuario, se le solicita que ingrese una frase que describa la temática que le gustaría explorar. Este paso es fundamental para personalizar las recomendaciones de actividades. A continuación se presenta el código utilizado para esta funcionalidad:

```python
def solicitar_tematica(self):
    """
    Solicita la temática de interés del usuario.
    """
    tematica = input("Ingresa una temática que le gustaría explorar: ")
    return tematica
```

Web Scraping.

Para recopilar información sobre los 1000 libros más populares del Proyecto Gutenberg, se implementó un proceso de web scraping utilizando las bibliotecas **requests** y **BeautifulSoup** de Python. Estas herramientas permitieron automatizar la extracción de datos de la web, facilitando la obtención de información relevante sobre cada libro, como título, autor, idioma, fecha de lanzamiento, resumen y temas.
Los datos recopilados se almacenaron en un archivo CSV para su posterior análisis y uso.

Link al repositorio en GitHub para ver el código completo: Web Scrapping en Github.

Búsqueda de Opciones y Recomendación.

El método a continuación llamado recomendar, se encarga de generar las recomendaciones personalizadas de juegos, películas y libros. Recibe como entrada el estado de ánimo del usuario y la temática deseada, combinando esta información en una frase que luego es vectorizada utilizando un modelo de embeddings. A continuación, se calculan las similitudes entre la frase vectorizada y los embeddings de los datasets de juegos, películas y libros. El método devuelve las tres mejores recomendaciones de cada categoría, basándose en las similitudes calculadas, y devuelve un diccionario con los títulos y resúmenes de las recomendaciones.

```python
def recomendar(self, input_estado_animo, estado_de_animo, input_tematica):
    """
    Genera recomendaciones basadas en el estado de ánimo, la temática y el modelo.
    """
    total = input_estado_animo + ' ' + estado_de_animo + ' ' + input_tematica
    print(f"Buscando recomendaciones para: {input_tematica}")

    # Vectorizamos la frase
    vector_comparar = self.modelo.encode(total)

    # Obtener los embeddings de cada dataset
    juegos_ubicacion = self.df_juegos_de_mesa['embeddings'].tolist()
    peliculas_ubicacion = self.df_peliculas['embeddings'].tolist()
    libros_ubicacion = self.df_libros['embeddings'].tolist()

    # Calcular similitudes
    similitudes_juegos = cosine_similarity([vector_comparar], juegos_ubicacion)
    similitudes_peliculas = cosine_similarity([vector_comparar], peliculas_ubicacion)
    similitudes_libros = cosine_similarity([vector_comparar], libros_ubicacion)
```

Con base en el estado de ánimo del usuario y la temática ingresada, el programa genera recomendaciones personalizadas. A continuación se detallan los pasos para obtener las mejores sugerencias.

```python
# Obtener los índices de las top 3 recomendaciones
top_indices_juegos = similitudes_juegos.argsort()[0][-3:][::-1]  # Top 3 juegos
top_indices_peliculas = similitudes_peliculas.argsort()[0][-3:][::-1]  # Top 3 películas
top_indices_libros = similitudes_libros.argsort()[0][-3:][::-1]  # Top 3 libros
```

A partir de los índices seleccionados, se crea una lista de diccionarios que contiene los títulos y descripciones de las recomendaciones. Este proceso se repite para las películas y libros, garantizando que el usuario reciba sugerencias relevantes y personalizadas basadas en su entrada.

```
recomendaciones_peliculas = [
    {
        'titulo': self.df_peliculas.iloc[index]['Title'],
        'summary': self.df_peliculas.iloc[index]['Description']
    } for index in top_indices_peliculas
]
```

El método ejecutar coordina el funcionamiento del sistema de recomendaciones en una única llamada. Solicita al usuario que ingrese una frase para predecir su estado de ánimo y que especifique una temática de interés. Luego, invoca el método recomendar para generar recomendaciones personalizadas, que finalmente se presentan en la consola. Este enfoque simplifica el flujo de la aplicación y mejora la experiencia del usuario al integrar todas las funcionalidades de manera coherente y eficiente.

```
def ejecutar(self):
    """
    Ejecuta el flujo completo de la aplicación.
    """
    frase_usuario = self.solicitar_input()
    estado_animo_predicho = self.procesar_estado_animo(frase_usuario)
    tematica = self.solicitar_tematica()

    # Generar recomendaciones
    recomendaciones = self.recomendar(frase_usuario, estado_animo_predicho, tematica)

    # Mostrar las recomendaciones
    print("\nRecomendaciones de juegos:")
    for rec in recomendaciones['juegos']:
        # print(f"- {rec['titulo']}: {rec['summary']}")
        print(f"- {rec['titulo']}")

    print("\nRecomendaciones de películas:")
    for rec in recomendaciones['peliculas']:
        print(f"- {rec['titulo']}: {rec['summary']}")

    print("\nRecomendaciones de libros:")
    for rec in recomendaciones['libros']:
        print(f"- {rec['titulo']}: {rec['summary']}")
```

Integración y Ejecución.

La resolución del trabajo la llevamos a cabo en dos archivos, el primero llamado ‘Recomendador’ donde resolvemos todo lo relacionado con la parte de la resolución de las consignas para la parte del recomendador, y el segundo llamado ‘Web_Scrapping’ donde tenemos toda la parte del Web Scrapping y donde creamos los Embeddings para pasarlos nuevamente a un csv para trabajarlos en el primer archivo.

Todo el proceso de clasificación de estado de ánimo y generación de recomendaciones está encapsulado en la clase **Recomendador**. Esta clase incluye todos los métodos necesarios para llevar a cabo las diferentes etapas del sistema de recomendación.

Link al código en GitHub: GitHub

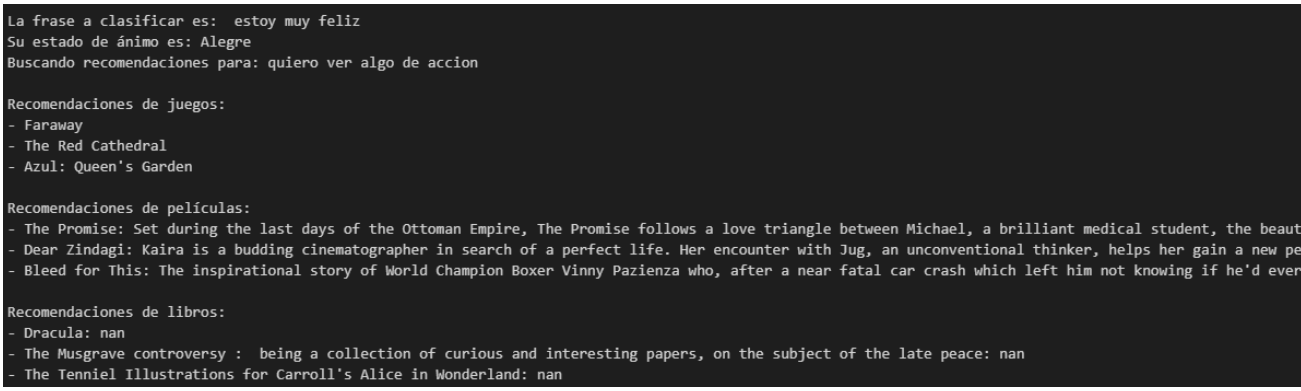
Componentes de la Clase:

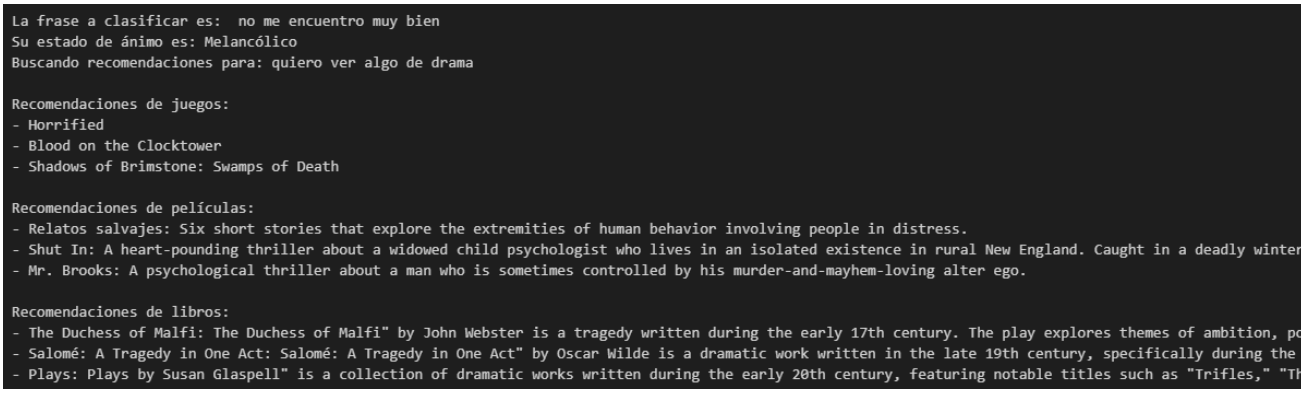
- **__init__**: Este método inicializa la clase, estableciendo los modelos y los DataFrames que contienen los datos de juegos de mesa, películas y libros. Se asignan las instancias de modelo, clf, y label_encoder para su uso posterior.
- **solicitar_input**: Método que solicita al usuario que ingrese una frase para clasificar su estado de ánimo.
- **procesar_estado_animo:** Convierte la frase del usuario en embeddings y utiliza el clasificador para predecir el estado de ánimo, devolviendo la clasificación en texto.
- **solicitar_tematica:** Pide al usuario que ingrese una temática de interés, que se utilizará para personalizar las recomendaciones.
- **recomendar:** Genera recomendaciones basadas en el estado de ánimo del usuario y la temática ingresada. Este método calcula las similitudes entre el vector del usuario y los embeddings de los datasets, seleccionando las tres mejores recomendaciones para juegos, películas y libros.
- **ejecutar:** Este es el método principal que orquesta el flujo completo de la aplicación. Solicita la entrada del usuario, procesa el estado de ánimo, pide la temática y finalmente genera y muestra las recomendaciones correspondientes.

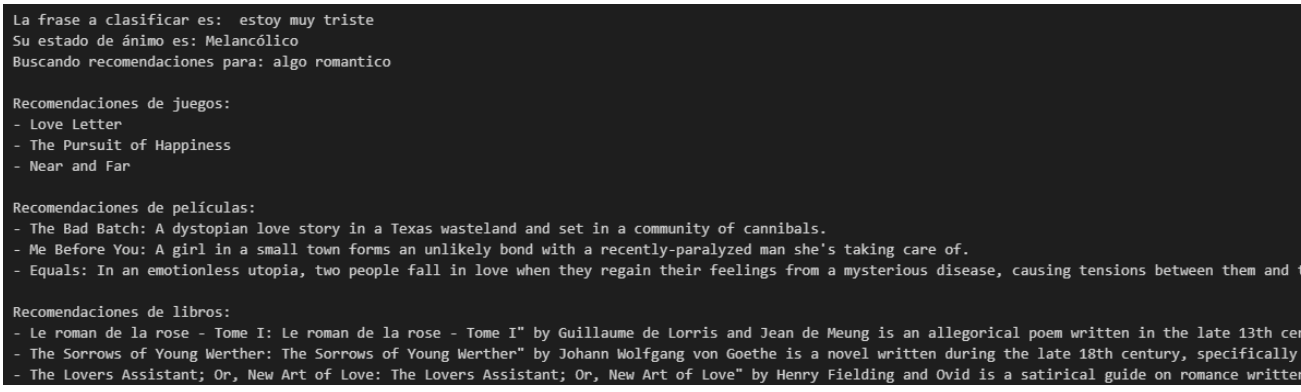
Ejecución del Programa:

Para ejecutar el programa, simplemente se invoca el método **ejecutar** de la clase **Recomendador.** Este método gestiona el flujo de la aplicación, desde la entrada del usuario hasta la presentación de las recomendaciones. Así, el usuario puede interactuar fácilmente con el sistema y obtener sugerencias personalizadas en función de su estado de ánimo y preferencias.

Resultados.

1. 

2. 

3. 

Podemos notar que en las tres imágenes resultantes se puede apreciar cómo detecta correctamente los estados de ánimos.

1. En la imagen uno podemos ver que:

Las recomendaciones de Juegos son: **Love Letter**: Un juego de cartas donde los jugadores intentan entregar cartas de amor a la princesa, **The Pursuit of Happiness**: Un juego que simula la vida, donde los jugadores toman decisiones sobre su carrera, relaciones y vida personal y **Near and Far**: Un juego de aventuras donde los jugadores exploran un mapa y completan misiones.

Las recomendaciones de Películas son:**The Bad Batch**: Una historia de amor en un contexto distópico. **Me Before You**: Un drama romántico que trata sobre la conexión entre

dos personajes en situaciones difíciles. **Equals**: Una película que explora el amor en una sociedad donde las emociones están prohibidas.

Las recomendaciones de Libros son: **Le roman de la rose**: Un poema alegórico del siglo XIII que explora el amor. **The Sorrows of Young Werther**: Una novela que trata sobre la pasión y el sufrimiento amoroso. **The Lovers Assistant**: Una guía satírica sobre el romance escrita por Henry Fielding y Ovidio.

2. En la imagen dos podemos ver que:

Las recomendaciones de Juegos son: **Horrified**: Un juego cooperativo de mesa donde los jugadores se unen para derrotar a monstruos clásicos del cine de terror, como Frankenstein y la Momia. Se centra en la estrategia y el trabajo en equipo para sobrevivir y completar objetivos. **Blood on the Clocktower**: Un juego de deducción social donde los jugadores asumen roles y deben descubrir quién es el villano en un contexto de misterio. Fomenta la interacción y la argumentación entre los participantes. **Shadows of Brimstone**: Swamps of Death: Un juego de aventuras en el que los jugadores exploran un mundo lleno de peligros y misterios, enfrentándose a criaturas sobrenaturales y recolectando tesoros.

Las recomendaciones de Películas son: **Relatos salvajes**: Una película argentina que presenta seis historias cortas que exploran los extremos del comportamiento humano y cómo las situaciones de estrés pueden llevar a decisiones extremas. **Shut In**: Un thriller psicológico que sigue a un psicólogo infantil viudo que vive en aislamiento y enfrenta una situación mortal durante una tormenta invernal. **Mr. Brooks**: Un thriller psicológico que narra la vida de un hombre que lidia con su lado oscuro, donde su alter ego le impulsa a cometer actos de violencia y caos.

Las recomendaciones de Libros son: **The Duchess of Malfi**: Una tragedia escrita por John Webster en el siglo XVII que explora temas de ambición, poder y corrupción en una historia de amor y traición. **Salomé**: Una obra dramática escrita por Oscar Wilde en el siglo XIX, que se centra en el tema de la seducción y el poder de la mujer en un contexto bíblico. **Plays**: **Plays by Susan Glaspell**: Una colección de obras dramáticas del siglo XX que incluye títulos notables que abordan temas de la vida femenina y social.

3. En la imagen tres podemos ver que:

Las recomendaciones de Juegos son: **Faraway**: Un juego de exploración en el que los jugadores deben descubrir nuevos mundos y superar desafíos. **The Red Cathedral**: Un juego de estrategia en el que los jugadores compiten para construir una catedral en Moscú. **Azul: Queen's Garden**: Una variante del popular juego Azul donde los jugadores diseñan jardines para la reina.

Las recomendaciones de Películas son: **The Promise**: Ambientada en los últimos días del Imperio Otomano, sigue un triángulo amoroso entre un estudiante de medicina, una mujer hermosa y un periodista estadounidense. **Dear Zindagi**: La historia de Kaira, una joven cinematógrafa, que busca la felicidad y encuentra nuevas perspectivas a través de una

relación con un pensador poco convencional. **Bleed for This**: Basada en la historia real del boxeador Vinny Pazienza, quien regresa al ring después de un accidente casi mortal.

Las recomendaciones de Libros son: **Drácula**: La clásica novela de Bram Stoker sobre el famoso vampiro y su intento de mudarse a Inglaterra. **Buccaneers and Pirates of Our Coasts**: Un libro que explora la historia de los piratas en las costas estadounidenses. **The Tenniel Illustrations for Carroll's Alice in Wonderland**: Un conjunto de ilustraciones de John Tenniel para la famosa obra de Lewis Carroll.

Conclusión.

En resumen, el sistema de recomendación ha demostrado ser bastante efectivo. El clasificador de estado de ánimo acierta en la mayoría de los casos, aunque todavía se podría mejorar su precisión. En cuanto a las recomendaciones, funcionan bien en general, pero a veces se confunden debido al contexto que se les proporciona. A pesar de esto, el sistema cumple con su objetivo de ofrecer sugerencias personalizadas y relevantes, haciendo que la experiencia del usuario sea positiva en la mayoría de las interacciones.