Tecnicatura Universitaria en Inteligencia Artificial
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario

Informe de Trabajo Práctico I: Clasificador de Recomendaciones Recreativas utilizando NLP

Cátedra: Procesamiento de Lenguaje Natural

Autores: Sofía Rondini y Facundo Fontela

Fecha: 6/11/2024

Índice

1. Resumen

En este trabajo práctico desarrollamos un sistema de recomendaciones recreativas para usuarios, basado en su estado de ánimo y preferencias, utilizando técnicas de Procesamiento de Lenguaje Natural. A través de un clasificador, se identifica el estado de ánimo del usuario, y con técnicas de embeddings y similitud semántica, se encuentran recomendaciones en bases de datos de películas, juegos de mesa y libros.

2. Introducción

- **Contexto:** Una persona dentro de un mes, se tomará 15 días de vacaciones en la playa. Sin embargo, se estima que durante al menos cuatro de esos días habrá lluvias, lo que podría limitar las actividades al aire libre. Para esos días de mal clima, se propone una solución que facilite la recreación en función del estado de ánimo del día.
- **Objetivo:** Desarrollar un programa de Procesamiento de Lenguaje Natural que, según el estado de ánimo del usuario, recomiende entre ver una película, jugar un juego de mesa o leer un libro (o varias opciones para cada caso). Para ello, deberá construir un clasificador que categorice el estado de ánimo del usuario. Luego sugerir el conjunto de recomendaciones basada en una frase de preferencia ingresada por el usuario.

3. Metodología

3.1 Clasificación de Estado de Ánimo

- **Objetivo:** El primer paso es clasificar el estado de ánimo del usuario en categorías ("Alegrìa", "Tristeza", "Enojo") a partir de un prompt ingresado.
- **Implementación:** Se usa un clasificador basado en aprendizaje supervisado, empleando técnicas de NLP para analizar el prompt de entrada del usuario.

3.2 Ingreso de Preferencias

- **Objetivo:** Capturar la temática de preferencia del usuario con una frase descriptiva.
- **Implementación:** Después de determinar el estado de ánimo, el usuario ingresa una frase que describe la temática deseada. Esta frase se usa para identificar términos clave mediante técnicas de embeddings y similitud semántica.

3.3 Búsqueda de Opciones

- **Objetivo:** Comparar la frase de preferencia del usuario con diversas fuentes de recomendaciones, como películas, juegos de mesa y libros.
- **Implementación:** A través de embeddings y métodos de similitud de textos para medir la relación entre la frase del usuario y las descripciones de las actividades en las bases de datos.

3.4 Generación de Recomendaciones

- **Objetivo:** Brindar recomendaciones adecuadas para el estado de ánimo y la preferencia ingresada.
- **Implementación:** A partir del estado de ánimo y la similitud temática calculada, el sistema recomienda actividades recreativas entre películas, juegos de mesa o libros, aplicando las herramientas de NLP vistas en las primeras unidades para generar sugerencias coherentes y personalizadas.

4. Desarrollo o Implementación

4.1 Librerías y función Translator

En este apartado vamos a ubicar todas las librerías del colab, para un mayor orden.

Aquí también colocamos una función que vamos a utilizar frecuentemente:

```
translator = Translator()

def translate(text):

  translation = translator.translate(text, src='es', dest='en')

  return translation.text
```

Como todos los datos de los dataset están en inglés, preferimos traducir los prompt en vez de las bases de datos. Porque nos iba a requerir menos tiempo y recursos computacionales.

4.2 Clasificador de estado de ánimo

4.2.1 Carga y Procesamiento de Datos

Para el clasificador optamos por utilizar una base de datos que contiene frases en inglés y la emoción asociada.

Las emociones en las que clasifica son: tristeza, alegría, amor, enojo, miedo y sorpresa. Cada uno etiquetado con un número (de 0 a 5 respectivamente).

Lo primero que hacemos es preguntarnos sí este dataset está balanceado, para que el modelo no nos quede sesgado.

```
Índice triste: 5797 veces
Índice alegre: 6761 veces
Índice amor: 1641 veces
Índice enojo: 2709 veces
Índice miedo: 2373 veces
Índice sorpresa: 719 veces
```

Vemos que el resultado no está balanceado. Por lo que vamos a tomar las siguientes decisiones:

- No vamos a tener en cuenta amor y sorpresa.
- De las emociones que dejamos para trabajar (enojo, tristeza y alegría) vamos a tomar la cantidad de la menor (enojo, 2709) igualar las restantes.

Preferimos contar con menos registros pero que esté balanceado.

4.2.2 Modelo de clasificación

La clasificación la realizamos utilizando embeddings generados por el modelo pre entrenado SentenceTransformer, que convierte las frases del usuario en representaciones vectoriales para capturar las relaciones semánticas entre palabras. Este enfoque nos permite reconocer la emoción centrándose en el contenido completo de la frase, mejorando la precisión.

Pasos principales:

- Convertir a minúsculas el dataset, para procesarlo mejor.
- Modelo de Clasificación: Con los embeddings obtenidos, entrenamos un modelo de Regresión Logística Multinomial para clasificar los estados de ánimo.

4.2.3 Métricas

```
Precisión Regresión Logística: 0.7742927429274292
Reporte de clasificación Regresión Logística:
              precision    recall  f1-score   support

           0       0.75      0.73      0.74       561
           1       0.81      0.80      0.81       545
           3       0.76      0.80      0.78       520

    accuracy                           0.77      1626
   macro avg       0.77      0.78      0.77      1626
weighted avg       0.77      0.77      0.77      1626
```

4.2.4 Función para clasificar

En la función vamos a procesar brevemente la frase de entrada: traducimos, convertimos a minúscula, luego la vamos a vectorizar con el modelo y devuelvo la emoción encontrada más la frase ingresada.

4.2.5 Pruebas/ejemplos

A Continuación mostramos ejemplos del clasificador en funcionamiento:

```
Frase: 'me siento con felicidad y entusiasmo'
La emoción más presente hoy es: alegre

Frase: 'Siento enojo, no tengo ganas de hacer nada'
La emoción más presente hoy es: enojo

Frase: 'Me siento inseguro'
La emoción más presente hoy es: triste

Frase: 'siento feliz y con amor'
La emoción más presente hoy es: alegre

Frase: 'me siento triste, las cosas no me motivan'
La emoción más presente hoy es: triste

Frase: 'hoy es un dia que me tiene irritado, no quiero hablar con nadie'
La emoción más presente hoy es: enojo

Frase: 'hoy me encuentro motivado para hacer nuevas cosas'
La emoción más presente hoy es: alegre
```

4.3 Recomendador

4.3.2 Juegos: analisis y limpieza de datos

Tenemos que focalizar primero en asignar una emoción a cada fila.

Analizamos las columnas del dataframe de juegos y observamos que tenemos las siguientes categorías:

Categorías únicas: ['Abstract Strategy', 'Action / Dexterity', 'Adventure', 'Age of Reason', 'American Civil War', 'American Indian Wars', 'American Revolutionary War', 'American West', 'Ancient', 'Animals', 'Arabian', 'Aviation / Flight', 'Bluffing', 'Card Game', "Children's Game", 'City Building', 'Civil War', 'Civilization', 'Collectible Components', 'Comic Book / Strip', 'Deduction', 'Dice', 'Economic', 'Educational', 'Electronic', 'Environmental', 'Expansion for Base-game', 'Exploration', 'Fantasy', 'Farming', 'Fighting', 'Game System', 'Horror', 'Humor', 'Industry / Manufacturing', 'Mafia', 'Math', 'Mature / Adult', 'Maze', 'Medical', 'Medieval', 'Memory', 'Miniatures', 'Modern Warfare', 'Movies / TV / Radio theme', 'Murder/Mystery', 'Music', 'Mythology', 'Napoleonic', 'Nautical', 'Negotiation', 'Novel-based', 'Number', 'Party Game', 'Pike and Shot', 'Pirates', 'Political', 'Post-Napoleonic', 'Prehistoric', 'Print & Play', 'Puzzle', 'Racing', 'Real-time', 'Religious', 'Renaissance', 'Science Fiction', 'Space Exploration', 'Spies/Secret Agents', 'Sports', 'Territory Building', 'Trains', 'Transportation', 'Travel', 'Trivia', 'Video Game Theme', 'Vietnam War', 'Wargame', 'Word Game', 'World War I', 'World War II', 'Zombies']

Luego vamos a asignarle a cada categoría una emoción, subjetivamente, según nuestra percepción.

Existe el caso de que hay varias categorías para cada juego, por eso usamos una función llamada emocion_mayoritaria, en la que elegimos la emoción que más se repite.

Ubicamos esta información en una columna llamada emoción.

Vemos que tan balanceado nos queda y alegría=800, tristeza=146, enojo=54. No está balanceado. Esto implica que, según el criterio que tomamos, se dispondrá de menos opciones para recomendar en el caso de enojo o tristeza.

4.3.3 Juegos: modelo y recomendador

Para juntar toda la información que necesitamos para recomendar vamos a concatenar las siguientes columnas: nombre del juego (game_name), categoría (categories), descripción (description) y elementos del juego (mechanics) en una nueva columna llamada “Combined”, que servirá para realizar embeddings con mayor información.

4.3.4 Peliculas: análisis y limpieza de datos

Cargamos los datos, y recolectamos los generos: 'Action', 'Adventure', 'Sci-Fi', 'Mystery', 'Horror', 'Thriller', 'Animation', 'Comedy', 'Family', 'Fantasy', 'Drama', 'Music', 'Biography', 'Romance', 'History', 'Crime', 'Western', 'War', 'Musical', 'Sport'.

Subjetivamente le asignamos una emoción a cada género y luego elegimos la mayoritaria para asignar a la fila.

Contabilizamos las cantidades: alegría=521, tristeza=320 y enojo=159. No está equilibrado, pero nuevamente no nos preocupa ya que no estamos entrenando.

4.3.5 Películas: modelo y recomendador

En este caso, en la columna Combined juntaremos al título (Title), género (Genre), descripción (Description) y director.

4.3.6 Libros: Web Scraping

Para recolectar la información que necesitamos para crear la base de datos primero vamos a solicitar la página, analizar el contenido HTML a través de BeautifulSoup y extraer autor, subject (descripción) y summary (resumen).

Luego recorremos las primeras 1000 páginas extrayendo esa información.

4.3.7 Libros: análisis y limpieza de datos

Para procesar vamos a extraer el título, que se encuentra entre comillas en la columna summary y poder diferenciarlo.

A su vez comenzamos a trabajar para categorizar en emociones las categorías.

Comenzamos planteando palabras “clave” de diferentes géneros para a través de ellos clasificar, como venimos realizando con los anteriores dataset. Generamos una función

clasificar_genero para que sí encuentra una palabra de estas clave en las descripciones y sino coloca “Desconocido”.

Para trabajar con los desconocidos vamos a extraer las categorías de entidades en el texto, para a través de ellos definir un género.

Una vez hecho esto contamos los “Desconocidos” y siguen siendo bastantes. Los vamos a relacionar según las categorías que devuelve NER una categoría.

Luego los contamos: ['ciencia ficción': 299, 'interés general': 185, 'historia': 165, 'romance': 117, 'poesía': 60, 'biografía': 59, 'diccionario': 30, 'terror': 18, 'filosofía': 16, 'educación': 11, 'religión': 10, 'arte': 5, 'fantasía': 4, 'psicología': 3]

Ahora relacionamos estas categorías a emociones. Quedándonos: enojo=482, alegría=412 y tristeza=88.

4.3.8 Libros: modelo y recomendador

Creamos el modelo con la columna combinada author, summary y subject (incluye el nombre en summary).

4.4 Recomendador de opciones

4.4.1 Procesamiento de frases de entrada

En primer lugar vamos a importar librerías y modelos.

Luego describimos una función extract_entities que a través de NER encuentra las entidades: género, autor, título, director o actor.

Ejemplo:

```
extract_entities("I want to read Science Fiction books from Stephen King, as The Green Mill. Or maybe watch movies like Titanic, from James Cameron with Leonardo Di Caprio")

Asking to truncate to max_length but no maximum length is provided and the model has no predefined maximum length. Default to no truncation.
{'genre': ['science fiction'],
 'author': ['stephen king'],
 'title': ['the green mill', 'titanic'],
 'director': ['james cameron'],
 'actor': ['leonardo di caprio']}
```

4.4.2 Recomendador de opciones de juegos, películas y libros

Desarrollamos una función para cada caso, que toma como parámetro el prompt del usuario y el dataframe.

En los 3 casos la función realiza las mismas tareas:

1. Traduce la frase y realiza el embedding

2. Con NER extrae entidades de la frase y compara con las del dataframe filtrando
    a. En el caso de películas se buscan las entidades: “actor”, “director”, “title”, “genre”
    b. Juegos: “title”
    c. Libros: “title”, “author”, “genre”
3. Si el resultado del filtrado da como resultado un dataframe menor o igual a 5, devuelve dicho subset como recomendación
4. Si el filtrado es mayor a 5, se compara con similitud para dar las recomendaciones.
5. Si el filtrado NER filtra por completo el dataframe (longitud 0), se restablece el dataframe inicial para comparar por similitud y devolver las 5 mejores.

4.5 Recomendaciones finales

Elaboramos la función final, que reúne todo lo trabajado, toma ambos prompts del usuario. Primero utiliza el clasificador para obtener la emoción y filtrar inicialmente los dataframe por esta. Luego llama a los 3 recomendadores detallados previamente.

Finalmente, esta devuelve las recomendaciones pertinentes de acuerdo a los criterios ya explicados.

5. Resultados

Aquí se mostrarán algunos resultados obtenidos en la aplicación del recomendador:

```
frase_emocion = "hoy me siento con ganas de hacer algo divertido"
frase_usuario = "quisiera ver o leer algo de accion o aventura"
recomendar(frase_emocion, frase_usuario)
```

```
La emocion encontrada es: enojo
Se recomiendan los siguientes juegos:
* Undaunted: Normandy
* Tajemnicze Domostwo
* Hammer of the Scots
* Detective: A Modern Crime Board Game
* 1775: Rebellion
Se recomiendan las siguientes peliculas:
* Live by Night
* King Cobra
* The Ghost Writer
* Crawlspace
* Contratiempo
Se recomiendan los siguientes libros
* The Burial of the Guns
* The Haunted Man and the Ghost's Bargain
* At the Earth's Core
* Summer
* The Cricket on the Hearth: A Fairy Tale of Home
```

En este ejemplo se observa como el uso de NER permite recomendar una película del director solicitado en el prompt y un libro relacionado con Titanic.

```
frase_emocion = "hoy me siento motivado para hacer cosas"
frase_usuario = "Estoy interesado en algo como Titanic de James Cameron"
recomendar(frase_emocion, frase_usuario)
```

```
La emocion encontrada es: enojo
Se recomiendan los siguientes juegos:
* Unfathomable
* Forgotten Waters
* Nemo's War (Second Edition)
* Oceans
* Deep Sea Adventure
Se recomiendan las siguientes peliculas:
* Avatar
Se recomiendan los siguientes libros
* Sinking of the Titanic and Great Sea Disasters
```

6. Conclusiones

Se logró implementar un sistema de recomendaciones personalizadas que responde a los estados de ánimo y preferencias temáticas del usuario. Esto ayuda a adaptar las recomendaciones en función del contexto emocional, proporcionando opciones relevantes y diversas (películas, juegos o libros) según la situación.

Se puede destacar que el principal inconveniente que surge en el clasificador es relacionado al filtrado por emoción en el dataframe. Esto se debe a que se ha tomado un criterio muy subjetivo para decir que película/libro/juego corresponde a cada emoción. Por ejemplo, es válido que para alguien “Avatar” sea una película para ver cuando se está alegre, mientras que para otra persona esta puede ser también una opción para días de tristeza.

En conclusión, nuestro programa funciona y cumple con los requerimientos propuestas por la cátedra.