Clasificador de Recomendaciones Recreativas utilizando Procesamiento de Lenguaje Natural

Autores: Iair Borgo Elgart, Albano Nardi

Tecnicatura en Inteligencia Artificial, UNR.

Asignatura: Procesamiento de Lenguaje Natural

Fecha de entrega: 6/11/24

**Índice**

1. Resumen
2. Introducción
3. Metodología
4. Desarrollo / Implementación
5. Resultados
6. Conclusiones
7. Referencias
8. Anexo

**1. Resumen**

Este trabajo presenta un clasificador de recomendaciones recreativas desarrollado utilizando técnicas de Procesamiento de Lenguaje Natural (NLP). El objetivo es clasificar el estado de ánimo de los usuarios y su preferencia para luego sugerir un libro, película o juego de mesa adecuado.

Para esto fue necesaria la recopilación de datos textuales sobre estados de ánimo, el preprocesamiento de estos datos y el entrenamiento de un modelo de clasificación basado en NLP.

**2. Introducción**

El objetivo principal es desarrollar un sistema que clasifique diferentes estados de ánimo y sugiera actividades recreativas. Este informe se organiza como sigue: primero, se presenta la metodología utilizada para la recopilación y preprocesamiento de datos. Posteriormente, se detalla el proceso de desarrollo del modelo de clasificación. Finalmente, se analizan los resultados obtenidos y se extraen conclusiones relevantes.

**3. Metodología**

Para la creación del clasificador de sentimientos, fue fundamental el recopilado y procesamiento de datos, así como la elección y entrenamiento de un modelo de clasificación de texto. A continuación, se detallan los pasos seguidos:

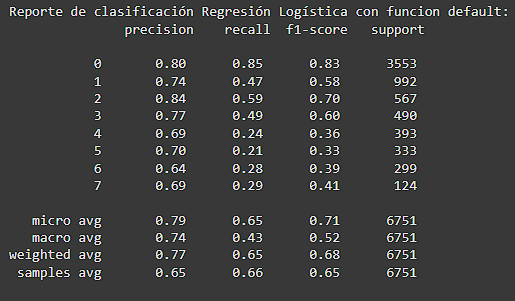
1. Recopilación de Datos  
   Se utiliza el Dataset GOemotions, que tiene etiquetas a mano de comentarios sacados de Reddit. Este fue proveído por Google para tareas de clasificación de emociones.
2. Preprocesamiento de Datos  
   Para asegurar que los datos estuvieran en un formato adecuado para el modelo, se aplicaron las siguientes técnicas de preprocesamiento:
   * Binarización MultiLabel: el dataset provisto podía tener mas de un sentimiento encontrado en cada frase, así que este tipo de binarización es como aplicar OneHotEncoding en base a las etiquetas.
   * Eliminación de sentimientos: debido al tamaño del dataset y el alcance limitado de este trabajo, fue necesario eliminar Procesamiento de Lenguaje Natural más de 20 sentimientos para quedarnos con una lista trabajable de 8. Además, se tuvo que eliminar las observaciones que quedaban sin ningún sentimiento reconocido
   * Vectorización de las frases: un requerimiento básico para toda clasificación con textos. Se utiliza el modelo de embedding MiniLM L6-v2
3. Selección de Modelo  
   Para la clasificación se utilizó regresión logística con un clasificador Multioutput, efectivamente entrenando una regresión por cada uno de los sentimientos presentes
4. Evaluación y Validación  
   El conjunto de datos se dividió en entrenamiento y prueba, asignando un 80% a entrenamiento y un 20% a prueba. Las métricas de evaluación empleadas fueron precisión, exactitud y F1-score.

Para la creación del recomendador, se contaban con 2 datasets provistos por la clase, siendo necesario obtener solo uno más:

1. Recopilación de Datos  
   Accediendo a la página del proyecto Gutenberg se obtienen datos de los libros más descargados mediante web scraping.
2. Preprocesamiento de Datos  
   Para estandarizar los datos se eliminaron columnas no deseadas y se pasó las descripciones por el modelo de embedding para obtener los vectores que representen cada una de estas.
3. Obtención de entidades reconocidas  
   Con las mismas descripciones, se utiliza GLiNER para poder obtener entidades de estas, pasándole una lista definida a mano de lo que deseamos obtener. Estas quedan guardadas en una lista para cada una de las observaciones de cada dataset, llamada “Entidades”.

**4. Desarrollo**

Una vez desarrollado el modelo de clasificación en base a las herramientas previamente comentadas, se obtuvieron las siguientes métricas



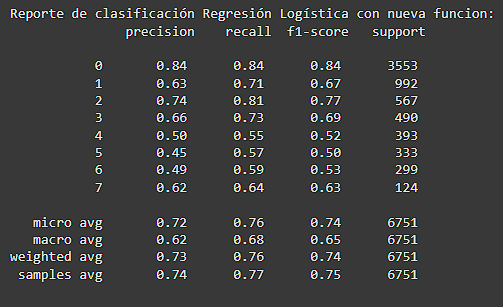
Primero, se variaron los thresholds iniciando en 0,05 teniendo un salto de 0.01 hasta llegar a 0.99. Para cada uno de estos se calcularon diversas métricas y se llegó a la conclusión de usar el F1-Score al poseer un balance entre precisión y exhaustividad.

Con estos thresholds obtenidos, notamos que surgían 2 problemas:

El primero de estos fue que en la práctica muchas frases eran reconocidas como “Neutro” y otro sentimiento al mismo tiempo, que no tenía sentido. Por lo que si se reconocía alguna otra emoción aparte de Neutro, poníamos a esta en 0.

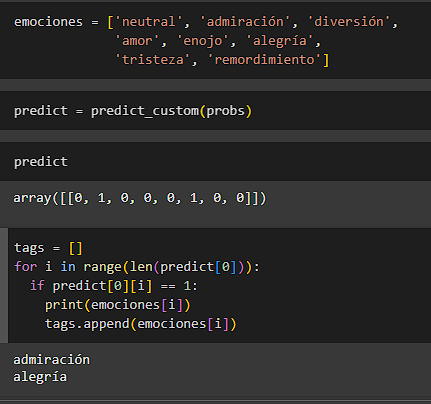
El segundo problema es que había casos en los que no se detectaban ningún sentimiento. Para estos casos se decidió dividir la probabilidad de cada sentimiento por su respectivo threshold, y agarrar el que tuviera el mayor número (es análogo a bajar el umbral hasta que se acepte un sentimiento).

Con estos cambios, se obtuvieron las siguientes métricas:



Mejorando así por mucho el recall, obteniendo una precisión levemente menor y aumentando el f1-score en todos los sentimientos. Veamos un ejemplo en la práctica:





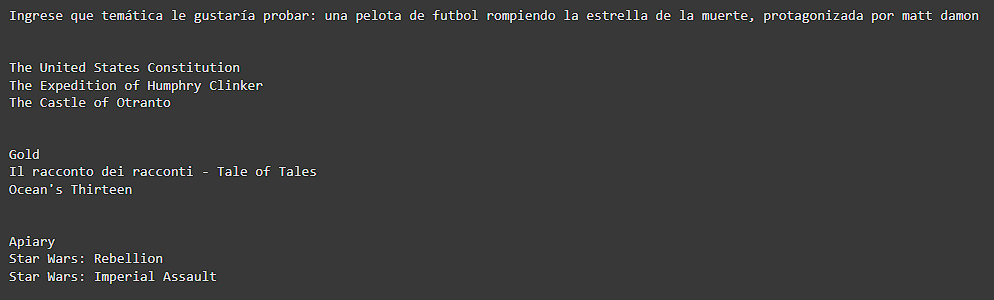
Notamos en este ejemplo que a pesar de escribir “Parque de diversiones” el modelo identifica correctamente alegría.

Con respecto al modelo de recomendación se crearon funciones para mejorar la capacidad del embedding de reconocer el tema que se trata:

Primero is\_match() que toma como entrada una a una cada entidad de la frase escrita por el usuario y una lista de entidades, que se obtienen viendo cada una de las observaciones de algún dataset. Al estar guardada en formato texto, uso un paquete llamado *ast* para que vuelva a ser una lista de Python, y se itera buscando la entidad nombra dentro de esta. Para suavizar errores de como está escrita por el usuario o en la base de datos se utiliza jaro-wrinkler con un threshold de 0.8, aunque este puede ser ajustado mediante un parámetro opcional. La salida es True si existe en la lista, caso contrario False.

Esta anterior función es utilizada en check\_entidad que la utiliza para todas las filas de los datasets y nos devuelve un vector del largo del dataset con la suma de la cantidad de entidades encontradas en la base de datos.

Usando la similtud del coseno, suma\_calcular\_similitud\_logodds() nos la devuelve escalada a todos los números reales, para después aplicarle suma\_entidad() que como parámetro toma alfa, que es el número que le vamos a sumar al obtenido anteriormente por cada una de las entidades reconocidas, pudiendo así obtener recomendaciones según lo nombrado por el usuario.

Tomando como ejemplo:

El modelo logra reconocer “Estrella de la muerte” y “Matt Damon” como entidades, lo que le permite realizar sugerencias decentes tanto en películas como juegos de mesa.

**5.Resultados**

Los resultados obtenidos del clasificador de emociones fueron evaluados en base a su capacidad para identificar distintos estados de ánimo y sugerir actividades recreativas.

1. Rendimiento General del Modelo  
   El modelo de clasificación de sentimientos logró una performance mas que aceptable, gran parte de este nivel es debido a la cantidad de datos que provee Google. En cambio, el modelo de recomendación muchas veces tenía resultados confusos que no eran los esperados. Puede ser que utilizar otro tipo de embedding ayude a mejorar estos casos. Otra posible mejora es extender el modelo NER para lograr reconocer casos que se pasaron por alto y así aprovechar las funciones
2. Análisis de Errores  
   Se identificaron ciertos errores en la clasificación de emociones que presentaban ambigüedad o ambivalencia. Por ejemplo, expresiones que contenían emociones mixtas o palabras que podían tener significados opuestos según el contexto presentaron un mayor índice de error. Además, al haber eliminado sentimientos, muchas veces nos encontraremos con la clasificación ´neutro´ en lugar de estos.

**6. Conclusiones**  
El clasificador desarrollado logró cumplir con el objetivo principal de reconocer sentimientos y sugerir actividades recreativas basadas en lo ingresado por el usuario. Los resultados muestran un desempeño efectivo del sistema de recomendación en la clasificación de emociones, aunque se identificaron áreas de mejora en el manejo de textos ambiguos.

Como recomendación para una mejora futura, se puede experimentar con modelos de clasificación más complejos, como redes neuronales, y ampliar el conjunto de datos con descripciones de emociones más variadas. También sería valioso incorporar características adicionales como el contexto del usuario para personalizar aún más las recomendaciones.

**7. Referencias**  
Henderson, M., Saeedi, A., & Google Research (2020). *GoEmotions: A dataset of fine-grained emotions* [Data set]. Hugging Face. <https://huggingface.co/datasets/google-research-datasets/go_emotions>

**8. Anexo**  
Código en Colab: <https://colab.research.google.com/drive/1QVDHm7ZHtj5pbfbLesJBWS7zDWDNBT3i?usp=sharing>