UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

GRADO EN INGENIERÍA INFORMÁTICA (ONLINE)

PROYECTO DE COMPUTACIÓN I

ACTIVIDAD 1

CRAWLING, SCRAPING Y PROCESAMIENTO DE DATOS

ALUMNOS:
Javier Mancebo Sánchez
Rubén Lago López

PROFESOR: Marc Franco Salvador
CURSO 2023-2024

Contenido

RESUMEN

Esta memoria corresponde a la entrega de la Actividad 1 crawling, scraping y procesamiento de datos de la asignatura Proyecto de computación I, Grado en Ingeniería Informática online de la Universidad Europea de Madrid. Dicha actividad tiene como finalidad la creación de un detaset con el cual conseguir un clasificador que distinga entre textos escritos por humanos y por ChatGPT (inteligencia artificial).

INTRODUCCIÓN

El objetivo del proyecto es la generación de un dataset con el cual entrenar un modelo que sea capaz de detectar si un texto ha sido escrito por un humano o generado por ChatGPT. Para ello el proyecto se ha divido en 4 puntos:

1. Web crawling: Recopilar webs con conversaciones entre usuarios y ChatGPT.
2. Web scryping: Extraer el texto humano y generado de las webs recopiladas.
3. Limpieza y generación dataset. Eliminar textos duplicados, con menos de 20 caracteres o que no sean inglés. Una vez limpio los datos generamos y guardamos el dataset.
4. Entrenamiento y evaluación del modelo.

ANTECEDENTES

Para dar solución a las necesidades que han surgido a lo largo del proyecto hemos utilizado diversas librerías de Python como:

- Request: realiza solicitudes HTTP y obtiene su contenido.
- BeautifulSoup: parsea la respuesta obtenida del Request y obtiene el contenido de las webs. Además, permite realizar búsquedas en el contenido HTML de forma muy cómoda.
- Langdetect: detecta el idioma de un texto.
- Pandas: trabaja con DataFrame, tablas bidimensionales en nuestro caso etiquetadas como humano y generado (para identificar el origen de los textos).
- Scikit-learn (sklearn): herramientas sencillas y eficaces para el análisis predictivo de datos.

Para el entrenamiento del modelo hemos utilizado como código base el facilitado en la asignatura Proyecto de Computación I, en el siguiente enlace su puede ver una copia de dicho código https://drive.google.com/file/d/1MK-8Wf81jwzmg5_YhZY3ybnZ5vVgT_pO/view?usp=sharing

A la hora de evaluar el modelo obtenido hemos utilizado las siguientes métricas.

- Precision: fracción de instancias identificadas correctamente como positivas entre el total de instancias identificadas como positivas.
- Recall: fracción de instancias identificadas correctamente como positivas entre el total de instancias que deberían de haber sido positivas
- F1- score: media armónica entre precisión y recall: $F1 - score = 2 * \frac{precision * recall}{precision + recall}$
- Accuracy: fracción de instancias identificadas correctamente como positivas entre el total de instancias.
- Macro avg: media simple de las métricas de evaluación utilizadas para cada clase en un problema de clasificación multiclase
- Weighed avg: promedio ponderado de las métricas utilizadas para cada case.

SISTEMA

Para crear el dataset vamos a utilizar las conversaciones entre usuarios y ChatGPT que almacena la web ShareGPT. Como Google tiene indexadas todas las subpáginas de ShareGPT vamos a utilizar Serper, un API para búsquedas en Google, para obtener las url de todas las páginas indicando en la query “site:sharegpt.com”. Con está configuración, Serper nos facilita un código que al ejecutarlo nos devuelve la información de las páginas en formato JSON, dentro del cual buscaremos los títulos que contengan “ShareGPT conversation” para almacenar en una lista los links asociados a esos títulos. Incluyendo este código dentro de un bucle podremos obtener todas las url que necesitamos.

Una vez conseguidas las url toca realizar la parte de Scraping. Obtenemos la página web utilizando Request y utilizando BeautifulSoup paseamos el HTML y buscamos el texto escrito por un usuario y el generado por ChatGPT, teniendo en cuenta que el humano tiene el campo identificador <p class="pb-2 whitespace-prewrap"> y el generado .

En este punto tenemos dos listas, una con los textos generados y otra con los humanos. Para que el dataset sea bueno primero hacemos una limpieza de los textos eliminando duplicados, aquellos de menos de 20 caracteres y los que no estén en inglés (utilizando Langdetect). Usando Pandas generamos un Dataframe en el cual guardamos las dos listas utilizando las etiquetas “humano” y “generado” para clasificar los textos. El Dataframe lo guardamos en formato tsv para poder utilizarlo en el entrenamiento del modelo sin necesidad de tener que ejecutar todo el código.

Antes de utilizar el dataset para entrenar el modelo balanceamos el número de instancias para tener la misma cantidad de generadas que de usuarios, lo realizamos mediante un subsampleo (utilizando pandas) para que la clase con más instancias pase a tener las misma que la clase con menos. Después asignamos un 80% de los datos al de entrenamiento y un 20% de los datos al test, para ello utilizamos train_test_split de Sklearn. Los dos conjuntos de datos obtenidos los vectorizamos utilizando TfidVectorizer también de Sklearn y ya podemos entrenar el modelo elegido, LogisticRegression.

RESULTADOS Y EVALUACIÓN

Muestra el rendimiento de la solución. Puede contener una evaluación comparativa con diferentes sistemas, un caso de uso, estadísticas de datos, análisis de salidas del sistema, datos numéricos en tablas, y figuras para ilustrar funcionamiento o calidad.

Después de realizar las tareas de web crawling, web scraping y limpieza hemos conseguido un dataset con las siguientes estadísticas.

```
Número de instancias en el dataset: 4157
Número de instancias humanas: 1916
Número de instancias generadas: 2241
Longitud media en carácteres de las instancias humanas: 484.84342379958247
Longitud media en carácteres de las instancias generadas: 1025.26416778224
```

Cómo hemos mencionado anteriormente, previo a entrenar el modelo hemos balanceado las instancias para que haya el mismo número de instancias de textos generados y humanos, además hemos dividido los datos en dos subconjuntos para utilizarlos en entrenamiento y en el test el modelo, siendo estás sus estadísticas:

```
Número instancias en el training: 3065
Número instancias en el test: 767
Número instancias humanas en el training 1532
Número instancias generadas en el training 1533
Número instancias humanas en el test 384
Número instancias generadas en el test 383
```

Con estos datos hemos entrenado un modelo LogisticRegression y evaluado contra los datos de test:

	precision	recall	f1-score	support
human	0.89	0.84	0.87	383
machine	0.85	0.90	0.87	384
accuracy			0.87	767
macro avg	0.87	0.87	0.87	767
weighted avg	0.87	0.87	0.87	767

CONCLUSIONES

Cierre del trabajo donde se resume lo presentado y se señalan los puntos fuertes.

En este ejercicio hemos aplicado lo aprendido en las clases para generar un dataset utilizando las conversaciones almacenadas en ShareGPT, filtrando entre instancias humanas y generadas y aplicando técnicas de limpieza. Dicho dataset lo hemos tenido que balancear y dividir en dos subconjuntos, training y test, con los que hemos entrenado nuestro modelo.

Aunque después de valuar el modelo contra el conjunto de datos de test obtenemos una accuracy del 87%, podemos apreciar un posible sesgo en el dataset al analizar sus estadísticas ya que la longitud media de caracteres de los textos generados es más del doble que la longitud media de caracteres de los textos escritos por usuario.

REFERENCIAS

Github: RxbxnUEM/proyectocomputacionI (github.com)

Serper: https://serper.dev/

ShareGPT: https://sharegpt.com/

Miniconda: https://docs.conda.io/projects/miniconda/en/latest/

Introducción a Conda: https://conda.io/projects/conda/en/latest/user-guide/getting-started.html

Documentación BeautifulSoup: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Langdetct documentación: https://github.com/Mimino666/langdetect

Sklearn TFF-IDF https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Sklearn train test split: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Precision, recall y f1-score: https://en.wikipedia.org/wiki/Precision_and_recall

Requests: https://requests.readthedocs.io/en/latest/

Set: https://ellibrodepython.com/sets-python