**Documentación del Proyecto: Análisis de Comentarios de Google Play con VADER**

**Resumen**

Este proyecto se centra en la obtención de comentarios de Google Play relacionados con una aplicación específica y su posterior análisis de sentimientos utilizando la biblioteca VADER. El objetivo es comprender las opiniones de los usuarios sobre la aplicación y evaluar su sentimiento general.

**Información General**

* **Nombre del Proyecto:** Análisis de Comentarios de Google Play con VADER
* **Fecha de Inicio:** 01/07/2023
* **Fecha de Finalización:** 30/10/2023
* **Responsable del Proyecto:** Grupo 6 IMMUNE TECHNOLOGY INSTITUTE
* **Versión del Código:** 1.0

**Objetivos**

1. Recopilar comentarios de Google Play relacionados con una aplicación específica.
2. Limpiar y preprocesar los datos de comentarios.
3. Realizar un análisis de sentimientos utilizando VADER.
4. Aplicar algoritmos de clasificación como kmeans, Random Forest etc. Para valorar el éxito del modelo.
5. Aplicar modelos supervisados como LDA (Latent Dirichlet Allocation) y SVM (Support Vector Machine)
6. Visualizar los resultados del análisis.
7. Identificar patrones y tendencias en los comentarios de los usuarios.

**Tecnologías Utilizadas**

* Python 3.x
* Librerías Python:
  + google\_play\_scraper
  + googletrans
  + nltk
  + sklearn
  + vaderSentiment
  + pandas
  + matplotlib
  + wordcloud
  + langdetect
  + os
  + random
  + datetime
  + collections
  + BeautifulSoup
  + requests
  + Gemsin
  + PyLDAvis
  + Seaborn
  + Numpy
  + TextBlob
  + ThreadPoolExecutor
  + Translator

**Flujo de Trabajo**

* Se recolectan los comentarios a través de la librería Google\_play\_scraper.
* Se realiza el preprocesamiento de los comentarios, que incluye la tokenización, eliminación de stopwords y limpieza de texto.
* Se detectan y eliminan comentarios en idiomas diferentes al inglés.
* Se aplica VADER a los comentarios y se almacenan en una copia del dataframe original
* Se muestra información acerca de las aplicaciones desarolladas por el mismo developer.
* Se muestra una gráfica en la que se comparan las puntuaciones dadas por los usuarios que comentan, y las puntuaciones dadas en general.
* Se muestra una gráfica para visualizar la evolución del sentimiento con el tiempo y las updates . Las updates se obtienen mediante un scraper de la página web [www.appbrain.com](http://www.appbrain.com) que contiene el changelog de las aplicaciones
* Se muestran graficas de Term Frequency
* Se muestran wordclouds de palabras positivas y negativas que contienen los comentarios con mayor frecuencia, así como de bigramas y trigramas
* Se muestra un gráfico de barras que muestra la cantidad de comentaros por mes.
* Se aplican algoritmos no supervisados como kmeans y se muestra una gráfica que representa la discrepancia que hay entre el score y el sentimiento que VADER le ha asignado
* Se aplican algoritmos supervisados como LDA y SVM

**Análisis de Sentimientos**

* Se utiliza la biblioteca VADER para analizar el sentimiento de los comentarios.
* Se calculan las puntuaciones de sentimiento negativo, neutro, positivo y compuesto para cada comentario.
* Se añade una nueva columna en función del signo de compound\_VADER

**Visualización de Datos**

* Se generan gráficos y visualizaciones para mostrar los resultados del análisis de sentimientos.
* Se crea una nube de palabras para resaltar las palabras más frecuentes en los comentarios positivos y negativos.
* En el código hay unas líneas para escribr el dataframe con las puntuaciones de Vader en un Excel para su posterior uso en programa como PowerBI para obtener graficos de forma más sencilla que con código de python

**Uso del Código**

* Para la ejecución del código será necesario la instalación de todas las bibliotecas utilizadas. Hay que recordar que en caso de ejecutarlo en notebook los pip irán precedidos de un signo de exclamación.

pip install gensim

pip install os

pip install re

pip install csv

pip install collections

pip install concurrent

pip install datetime

pip install random

pip install numpy

pip install matplotlib

pip install nltk

pip install scikit-learn

pip install textblob

pip install vaderSentiment

pip install pandas

pip install google-play-scraper

pip install tqdm

pip install wordcloud

pip install seaborn

pip install statsmodels

pip install langdetect

pip install googletrans

pip install requests

pip install beautifulsoup4

#primero hay que importar nltk

import nltk

nltk.download('wordnet')

nltk.download('stopwords')

nltk.download('punkt')

* El código se encuentra en un archivo Python y se puede ejecutar para realizar análisis NLP (natural language processing) en diferentes aplicaciones de Google Play. En los puntos posteriores procederé a explicar de forma detallada el uso del código, la mayoría de código “modificable” se encuentra entre la línea 53 y la línea 118 del código.
* La forma de aplicarlo es ir a la línea 53 de código y en la variable app\_id insertar la id de la aplicación que queremos , para ello necesitamos ir a Google play , buscar la aplicación , y con el link , por ejemplo : [https://play.google.com/store/apps/details?id=**com.activision.callofduty.shoother**&hl=es&ql=US](https://play.google.com/store/apps/details?id=com.activision.callofduty.shoother&hl=es&ql=US), la ID seria lo marcado en verde. A continuación, proporcionaré algunas recomendaciones y observaciones:
  + Lo óptimo es analizar aplicaciones con una antigüedad entre 2 años y 4 meses
  + Es recomendable analizar apps con no demasiadas reseñas ya que debido a limitaciones de hardware la recolección de comentarios podría no tener la velocidad deseada. en mi caso el ritmo era de 200 comentarios/segundo, por lo que, lo más adecuado es analizar aplicaciones que posean entre 500 y 10000 comentarios
* En la línea 58 se encuentra la variable csv\_filename , en dicha variable se introducirá el nombre del archivo en el que queramos guardar seguido de .csv . Ejemplo: comentarios\_app.csv. La información incial que incluye es la siguiente: User; Date; Score; Comment y Helpful Count. **Hay que tener en cuenta que si el archivo comentarios\_app.csv (nombre que le hemos dado al archivo en el ejemplo) tiene contenido, el código saltará directamente a la parte de comprobación del idioma de los comentarios, por lo que no recolectará comentarios**, este mecanismo es útil para no tener que recolectar los comentarios cada vez que queramos ejecutar el código. Para recolectar comentarios hay que asegurarse de que el archivo está vacío, o de que creamos un nuevo archivo, cambiando el nombre en la variable csv\_filename.
* En la línea 61 se encuentra la variable csv\_filenameVADER y a diferencia del punto anterior este dataframe contiene las columnas: Negative\_VADER , Neutral\_Vader , Positive\_VADER , Compound\_VADER y Sentiment , además de las columnas comentadas en el punto anterior. El formato de nombre de archivo será igual que el punto anterior, ejemplo: comentarios\_app\_con\_sentimiento.csv, **en este caso no hace falta que el archivo esté vacío, se sobrescribirá el contenido con cada ejecución. Si se necesita almacenar información de varias apps, lo lógico es cambiarle el nombre al archivo de salida en la variable csv\_filenameVADER.**
* En la línea 64 se incluye la opción de escribir el dataframe con el preprocesado de texto y VADER aplicado en un Excel, ejemplo: comentarios\_app\_con\_sentimiento.xlsx.
* En la línea 70 se incluye un filtro opcional con keyword en el caso de que busquemos una palabra en concreto en un rango de fechas en concreto, la fecha de las variables start\_date y end\_date, tienen que ser con el **formato YYYY-MM-DD**.
* En la línea 77 del código se encuentra la variable custom\_stopwords. De forma predefinida, la biblioteca nltk incluye un diccionario de palabras en varios idiomas que nos permite eliminar un gran conjunto de palabras con poco valor para nuestro análisis, como son los artículos.En la línea 78 del código se encuentra el fragmento “custom\_stopwords.update” que nos permitirá “vetar” palabras en nuestro análisis posterior , ya que hay palabras que pueden no aportar demasiado al análisis
* En la línea de código 84, se incluye el “custom\_lexicon”. A pesar de que VADER posee un extenso lexicón (un diccionario que relaciona palabras con puntuaciones), puede no incluir las palabras y la puntuación que se espera al aparecer esa palabra, por lo que podemos personalizar la puntuación que le daremos a una palabra en concreto para que posteriormente VADER le asigne la puntuación. Ejemplo en concreto del codigo : La palabra “ads” (advertisement) , VADER la incluía como neutra y muchos comentarios estaban clasificados como neutros a pesar de que la gente se quejaba de una gran cantidad de anuncios, el custom\_lexicon sirve para abordar este problema mediante la asignación de una puntuación negativa a la palabra “ads”.
* En la línea 109 del código se incluye la variable “top\_n” , el valor de esta variable indica cuantas TermFrequency(frecuencia de palabras) queremos incluir en la gráfica de palabras más comentadas. En la línea 110, se encuentra la variable top\_n\_bigrams , que al igual que “top\_n” indica cuantas frecuencias de palabras queremos incluir en la gráfica , pero esta vez solo tendrá en cuenta los bigramas (composición de 2 palabras). Es el mismo caso con top\_n\_trigrams , pero esta vez se refiere a los trigramas (composición de 3 palabras).
* En la línea 113 a la 119 se incluye un filtro opcional por si se requiere encontrar aplicaciones con un género en específico y muestra la información básica de las aplicaciones con dicho género.
* En la línea 125 se ha incluido una variable para filtrar las fechas del grafico de evolución del sentimiento, ya que en periodos de más de 365 días funciona mal, de predefinido estará el valor days = 100, esto significa que solo mostrara la evolución de sentimiento de los últimos 100 días.

**Mejoras Futuras**

* Posibles mejoras en la recopilación de datos, como la inclusión de más comentarios o la implementación de una interfaz de usuario, como solicitar la app\_id antes de ejecutarlo en vez de tener que ir a la línea de código, (obviamente al tratarse de un capstone y no ir dirigido a un público general, esta mejora no se ha implementado, aunque sería sencillo).
* Implementación de traducción paralelizada para optimizar el tiempo de ejecución con EggThreadExecutor o pyspark, y corrección ortográfica de los comentarios con Textblob. Ambos están preinstalados en el código con las funciones creadas.
* Ampliación de las visualizaciones y análisis de datos.