UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

Máster en Software de Sistemas Distribuidos y Empotrados

Sistemas Distribuidos

Trabajo Final: Análisis de Sentimientos en Reddit

Alejandro Casanova Martín

N.º de matrícula: bu0383

Madrid, 24 de octubre 2023

Índice

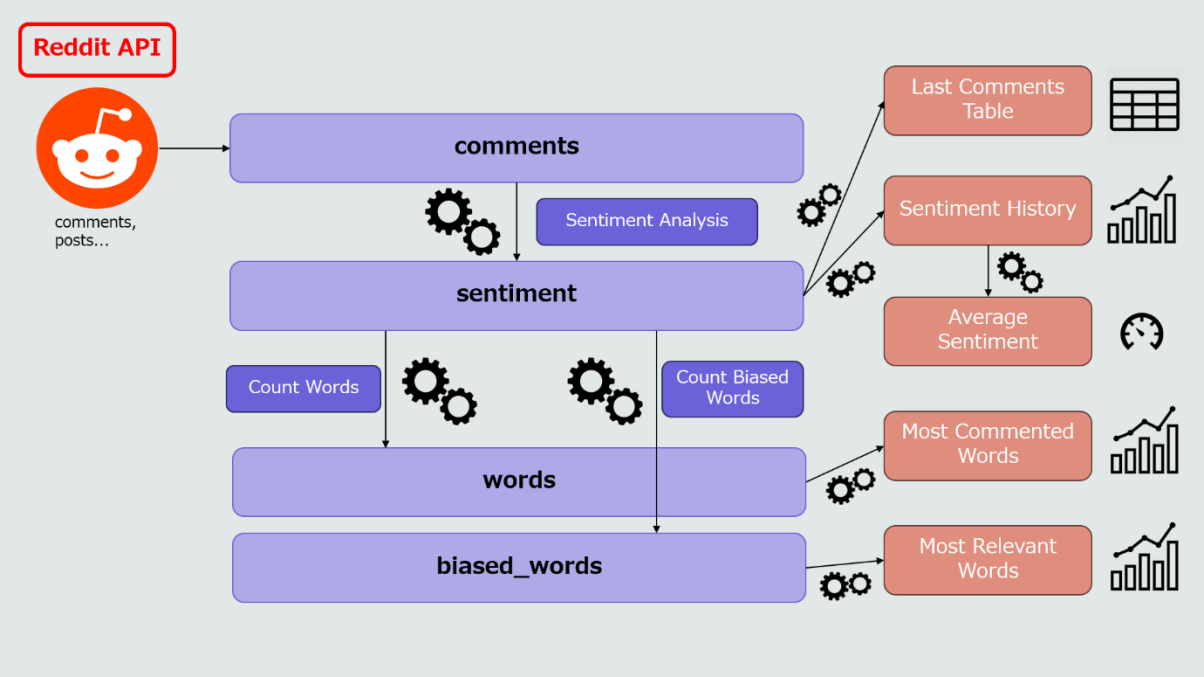
1. Planteamiento

Se implementó una aplicación para el análisis de sentimientos de los comentarios publicados en la plataforma **Reddit**. A partir del flujo en tiempo real recibido a través de la API de Reddit, el objetivo era calcular el sentimiento general en un *subreddit* especifico, y contabilizar las palabras más utilizadas, así como aquellas con una mayor carga subjetiva. La aplicación cuenta con un modo "*streaming*", que procesa los comentarios que van llegando en tiempo real desde un *subreddit* de noticias; y también cuenta con la opción de realizar búsquedas particulares en *subreddits* específicos, para procesar los principales resultados obtenidos.

2. Diseño del Sistema

2.1 Descripción General de la Arquitectura

La programación de la aplicación se ha realizado utilizando **Node-Red**. El procesamiento de los flujos de datos se realizó en el servidor mediante la herramienta **Kafka**. Para la comunicación entre el servidor y los posibles clientes (dispositivos móviles) que se conecten, se utilizó el protocolo **MQTT**, en particular la implementación del bróker **Mosquitto**.



2.2 Flujos de Node-Red

- **Kafka**: en este flujo se configuran los productores y consumidores principales de *Kafka*. Se introduce el contenido recibido de Reddit al *topic* "*comments*", se realiza un análisis de sentimiento a dicho flujo y el resultado se introduce al flujo "*sentiment*", y finalmente se separan las distintas palabras y se introducen a los flujos "*words*" y "biased_*words*" (en este segundo caso, tras realizar un segundo análisis de sentimiento individualmente a cada palabra). También se procesa el contenido obtenido al realizar una búsqueda puntual, siendo introducido también en el *topic* "*comments*".
- **Sentiment Analysis**: en este flujo, se consumen los datos del *topic* "sentiment" y se procesan para ser representados en diferentes componentes de la interfaz: una tabla con los últimos comentarios recibidos y su puntuación de sentimiento, una gráfica de línea que representa el historial de puntuaciones de los últimos 10 minutos, y un indicador de la puntuación media de

sentimiento de los últimos 10 minutos (que también se publica en el *topic* correspondiente del bróker *mosquitto*).

- **Word Counter:** se realiza un recuento de las palabras recibidas, y se representan en una gráfica de tipo "radar" las seis palabras más utilizadas.
- **Biased Word Counter:** se realiza un recuento de las palabras que el análisis de sentimientos ha identificado como "subjetivas", y se representa en una gráfica de barras aquellas que han tenido mayor peso en la puntuación del sentimiento general de los comentarios.
- **Controls:** se configuran los inputs de la interfaz del servidor que permiten el uso de la aplicación: un interruptor para activar y desactivar el modo "*streaming*" de comentarios, un selector de *subreddits* para las búsquedas realizadas, un campo de texto para introducir la búsqueda que se desea hacer, y un botón "*reset*" para reiniciar la aplicación. En este flujo también se realizan las subscripciones a los *topics* de "*control*" del bróker *mosquitto*, y se publica en los *topics* de "*status*".
- **MQTT-Mobile:** en este flujo se emula un posible dispositivo móvil que se conecte a la aplicación. Se configuran los elementos necesarios de la interfaz: un interruptor para activar y desactivar el modo "*streaming*", un indicador del estado de dicho modo, un selector de *subreddits*, y un campo de texto para realizar búsquedas. En este flujo se realizan las subscripciones a los *topics* de "*status*" del bróker *mosquitto*, así como también se publica en los *topics* de "*control*".

2.3 Topics de Kafka

- **comments**: se publica el contenido en bruto recibido a través de la API de Reddit (únicamente el texto). Originalmente sólo se publicaban los comentarios, pero finalmente también los títulos y texto principal de los posts recibidos al realizar una búsqueda particular.
- **sentiment**: se publica el contenido del anterior *topic*, una vez ha sido procesado por el análisis de sentimientos. Dichos eventos incluyen el texto original, así como toda la información producida por dicho análisis.
- **words**: se publica individualmente cada palabra del contenido recibido. El propio análisis de sentimiento separa el texto en tokens, por lo que procesando los eventos del *topic* "*sentiment"*, se obtienen las palabras que se publican en éste.
- **biased_words**: de forma similar al *topic* anterior, en este se publican los eventos del *topic* "*sentiment*" tras realizar un procesamiento de estos. Se separan las palabras con carga subjetiva, y se realiza un segundo análisis de sentimiento para obtener una puntuación de cada una. Por lo tanto, cada evento de este *topic* cuenta con una palabra subjetiva y su correspondiente puntuación de sentimiento.

2.4 Topics de MQTT

- **control/stream:** en este *topic* los clientes publican el estado deseado del modo "*streaming*", que deberá ser un *string* de valor "true" o "false". El servidor se subscribe a este *topic* para recibir los comandos.
- **control/query:** en este *topic* los clientes publican un *string* con las palabras de la búsqueda que desean realizar. El servidor se subscribe para recibir dicha petición y realizarla a través de la API de Reddit.
- **control/subreddit:** aquí los clientes publican un *string* con el nombre del *subreddit* en el que desean realizar la búsqueda correspondiente. El servidor se subscribe para recibir dicha petición.
- **status/stream:** aquí el servidor publica el estado actual del modo "*streaming"*, para que los clientes puedan recibirlo y utilizarlo como *feedback*.

- **status/query:** aquí el servidor publica la última búsqueda realizada, para que los clientes puedan recibirla y utilizarla como *feedback*.
- **status/subreddit:** aquí el servidor publica el *subreddit* actualmente seleccionado, para que los clientes puedan conocer su estado actual.
- **metrics/sentiment:** aquí el servidor va publicando el valor medio actual de sentimiento del contenido recibido, de modo que los clientes puedan subscribirse y recibirlo.

3. Semántica de Productores Consumidores

3.1 Kafka

Los *topics* se configuraron por defecto, por lo que la semántica en todos los casos es "*at-least-once*". Una duplicación de algún evento introducirá un poco de ruido en los valores calculados, pero no será crítico.

3.2 MQTT

- **control/stream:** QoS: 1, retain: false.
- **control/query:** QoS: 2, retain: false.
- **control/subreddit:** QoS: 1, retain: false.

La elección del *QoS* de los *topics* de control depende de la experiencia de usuario que queremos conseguir. En los tres casos, se escogió un nivel de al menos 1, dado que la frecuencia de petición realizadas por un usuario humano a través de un móvil será baja, y queremos evitar que el usuario realice una petición y esta no llegue, dado que supondría una mala experiencia de usuario. Para la selección del modo "*streaming*" y del *subreddit* actual, se eligió el nivel 1, dado que no supone ningún problema que dicha petición llegue más de una vez, el valor seleccionado seguirá siendo el mismo. Sin embargo, en el caso del campo de búsqueda, si dicha petición se duplica, la búsqueda se realizará varias veces, teniendo como resultado una mala experiencia de usuario.

- **status/stream:** QoS: 1, retain: true.
- **status/query:** QoS: 1, retain: true.
- **status/subreddit:** QoS: 1, retain: true.

Los *topics* de "*status*" serán leídos por los dispositivos móviles que se conecten a la aplicación. Para la mejor experiencia de usuario, se escogió el nivel *QoS* 1, dado que la frecuencia de publicación en estos *topics* será baja, y queremos que el usuario no pierda información del estado de la aplicación. En este caso, no importa que un evento de estado llegue duplicado. Se ha activado la opción "*retain*" para que al conectarse un nuevo dispositivo, este reciba el estado actual de la aplicación.

- **metrics/sentiment:** QoS: 0, retain: false.

En este *topic* se va publicando en tiempo real el valor medio de la puntuación de sentimiento. Dada la frecuencia de publicación, no resulta un problema que se pierda algún mensaje.

4. Conclusiones

En el transcurso de este trabajo se han empleado las tecnologías **Node-RED**, **Kafka** y **MQTT**, para llevar a cabo un análisis de sentimientos de los comentarios obtenidos a través de la API de Reddit. Mediante la realización de este trabajo, se ha confirmado que cada una de estas tecnologías aporta ventajas significativas, y su combinación ha permitido la construcción de un sistema eficiente y escalable para analizar y comprender la opinión de los usuarios en esta plataforma de redes sociales.

Node-RED se ha demostrado como una herramienta versátil para la construcción de flujos de trabajo de manera visual e intuitiva. Su interfaz gráfica facilita el desarrollo sin requerir una codificación intensiva, facilitando la realización pruebas y ajustes en tiempo real. Además, cuenta con una amplia biblioteca de nodos predefinidos, facilitando la integración con servicios como *Kafka* y *MQTT*, entre otros. Finalmente, este entorno se basa en *Node.js*, por lo que aprovecha al máximo su modelo no bloqueante basado en eventos. Esto lo hace ideal para aplicaciones en el *edge*, en hardware de bajo coste como la *Raspberry Pi*, así como en la nube.

Kafka ha sido esencial para la construcción de una arquitectura de mensajería robusta, escalable, y persistente. Esto ha permitido manejar grandes volúmenes de comentarios de Reddit y asegurar que los datos estén disponibles en todo momento. Sus capacidades de procesamiento en tiempo real han garantizado que los datos se entregaran de manera oportuna a los distintos componentes de la aplicación

MQTT es un protocolo de comunicación ligero con un uso de recursos mínimos, óptimo para comunicaciones con una baja frecuencia de mensajes, altas latencias, o conexiones inestables. Ha sido utilizado para implementar la comunicación con los diferentes dispositivos móviles que se conecten a la aplicación. Es un protocolo altamente escalable y que garantiza la entrega de mensajes.

5. Futuros Trabajos

En el futuro podrían implementarse nuevas métricas para evaluar el contenido generado en Reddit, hacer un análisis de sentimientos más sofisticado, o realizar un mejor filtrado de las palabras relevantes en función del contexto o del tipo de dichas palabras (sustantivos, preposiciones, verbos…).

Implementando módulos personalizados en *Node-RED*, podría añadirse la posibilidad de activar el modo "*streaming*" para diferentes *subreddits*, y no únicamente para el de noticias. Haciendo uso de la funcionalidad "*last-will*" de *MQTT*, podría llevarse un registro en el servidor de los dispositivos móviles conectados actualmente a la aplicación, y representar dicha lista en la interfaz de usuario del servidor.

También podría desarrollarse una aplicación móvil que permita la conexión con el bróker *mosquitto*, para probar el funcionamiento de la aplicación con un dispositivo móvil real, en lugar de uno emulado. Además, a la interfaz móvil podrían añadirse funcionalidades adicionales para aprovechar la funcionalidad del servidor al completo, y representar todas las métricas medidas en este.