# Instrucciones Miniproyecto 3

## Introducción

En los últimos años las redes sociales han adquirido gran importancia al permitir la comunicación directa entre un gran número de usuarios, rompiendo algunas barreras como la ubicación geográfica y posición social entre personas, y brindando además, la posibilidad de difundir opiniones de forma masiva. Si bien este tipo de plataformas presentan grandes ventajas en el desarrollo de las comunicaciones y en la generación de información, también presentan algunos aspectos negativos, de entre los cuales se destaca, el servir de medio para la manifestación de mensajes ofensivos que afectan a personas o grupos, los cuales pueden terminar en acciones violentas.

Las grandes empresas de redes sociales como Facebook y Twitter, tienen un alto interés en la detección automática de mensajes ofensivos con el objetivo de frenar su difusión, y así evitar posibles hechos de violencia. Sin embargo, esta tarea presenta dos dificultades inherentes a las características de los datos, que son:

1. el manejo de grandes volúmnes de datos generados por millones de personas alrededor del mundo,
2. la clasificación de un mensaje como ofensivo a partir del análisis del texto.

Para resolver estas dificultades, se deben utilizar de forma combinada las dos principales herramientas que hemos visto a lo largo de este curso, que son: manejo de Big Data y técnicas de Machine Learning.

En el presente miniproyecto trabajaremos con datos provenientes de la red social Twitter y abordaremos el problema de clasificación de mensajes a partir del texto. Para ello, utilizaremos gran parte de las herramientas aprendidas durante el curso, con el fin de entrenar un modelo que permita clasificar mensajes de Twitter en uno de los tres siguientes tipos:

1. mensaje de odio (hate speech),
2. mensaje ofensivo (offensive language),
3. ninguno de los anteriores (neither)

## Análisis de Datos

El dataset que utilizaremos en esta oportunidad, corresponde a la base de datos *Hate Speech and Offensive Language Dataset*, publicada en la comunidad de ciencia de datos *Kaggle* [1]. Estos datos corresponden a twitters publicados por usuarios reales dentro de la red social, y que han sido clasificados por la plataforma de *crowdsourcing CrowdFlower*, en una de las 3 clases que se muestran a continuación:

1. Clase 0: mensaje de odio (*hate speech*)
2. Clase 1: mensaje ofensivo (*offensive language*)
3. Clase 2: ni de odio, ni ofensivo (*neither*)

Debido a que cada twitter dentro del dataset ha sido etiquetado por más de una persona, es posible que hayan existido ciertas discrepancias en los criterios de su clasificación. Para poder resolverlas y definir la clase de cada dato, se adopta el criterio mayoría de votos, es decir, si por ejemplo 2 etiquetadores asignaron la clase 0 a un determinado twitter, y 3 etiquetadores la clase 1, el twitter en cuestión será definido como un twitter de clase 1. Dado esto, el dataset además entrega información acerca del número de personas que optaron por una determinada clase, por cada twitter.

La tabla que se muestra a continuación contiene los campos que componen el dataset, y dentro de ella, se han dispuesto tres datos a modo de ejemplo.

| tweet_id | count | hate_speech | offensive_language | neither | class | tweet |
|---|---|---|---|---|---|---|
| 0 | 3 | 2 | 1 | 0 | 0 | *hate speech* |
| 1 | 3 | 0 | 3 | 0 | 1 | *offensive language* |
| 2 | 3 | 0 | 1 | 2 | 2 | *neither* |

A continuación, se entrega una breve descripción de cada campo:

- tweet_id: número único para identificación de cada dato o tweet.
- count: número total de etiquetadores que clasificaron el tweet en cuestión
- hate_speech: número de etiquetadores que clasificaron el tweet como *mensaje de odio*
- offensive_language: número de etiquetadores que clasificaron el tweet como *mensaje ofensivo*
- neither: número de etiquetadores que clasificaron el tweet como *ni de odio, ni ofensivo*
- class: número que identifica la clase asignada al tweet en cuestión, según:
    1. *hate speech*,
    2. *offensive language*,

3. *neither*

- tweet: mensaje de texto original escrito en Twitter

Para facilitar la carga de datos en Colab y su posterior lectura, usted contará con una base de datos MySQL poblada con todos los datos que componen el dataset. Dicha base de datos tendrá por nombre *testdb*, y será creada durante la instalación del software necesario para el miniproyecto, por medio del script *miniproyecto3_installer.py*. Dentro de esta base de datos, se creará la tabla *hate_speech* que contendrá todos los datos a utilizar en esta experiencia.

## Actividades

Para realizar el proyecto, deberá estar familiarizado con las herramientas de *Sqoop*, *Hive*, *Spark SQL* y *Sparl MLlib*. Las actividades permitirán condensar lo aprendido durante el curso en un ejercicio completo, desde la incorporación de los datos en una plataforma de *Big Data* como Apache Hive al procesamiento de datos y entrenamiento de un modelo predictivo con Apache Spark. Dispone de una plantilla llamada *plantilla_mini_proyecto_3.ipynb* como un notebook de Google Colab base, el que deberá completar donde corresponda según las instrucciones de cada actividad.

A continuación, se especifican las actividades que deberá realizar:

1. Suponiendo que tiene el entorno ya generado de Hadoop y Spark, revise la tabla *hate_speech* de la base de datos *testdb* de MySQL, que se crea usando el mismo script de instalación. Para esto, use el comando

   mysql -u root –password=password testdb

   Ejecute una consulta que muestre los primeros 5 registros de la tabla mencionada.

2. Usando el comando sqoop, inserte la base de datos *testdb* a Hive desde MySQL. Use como fuente la conexión jdbc:mysql://localhost/testdb. Muestre el output de sqoop donde se muestran las tareas MapReduce de la carga de datos a Hive.

3. En Hive, muestre los primeros 10 registros de la tabla *hate_speech*.

4. Usando la instancia spark de tipo SparkSession, genere un Dataframe df_hate a partir de la tabla *hate_speech* de Hive. Filtre aquellos *tweets* nulos y aquellos cuyo largo sea menor a un caracter. (*hint*: puede usar la función LENGTH de SQL). Muestre los primeros 10 registros del Dataframe generado y compárelo con el generado en Hive. A continuación, con uno de los comandos vistos durante el curso, separe el Dataframe en Dataframes de training y testing, donde contengan el 95% y 5% de los datos respectivamente.

5. Usará funciones de Spark para el procesamiento de texto, en particular la clase RegexTokenizer, que separará un texto en sus palabras constituyentes sin considerar signos de puntuación, y la clase StopWordsRemover, que permitirá eliminar palabras muy comunes. Investigue los argumentos de entrada de ambas clases en [2].

Ambas clases son de tipo transformadoras, por lo que implementan una función transform. En particular, deberá definir los argumentos de entrada faltantes a ambas instancias para definir las columnas de entrada y salida.
En particular, para el Dataframe training aplique el *tokenizer* para generar el Dataframe training_words donde words debe ser la columna con las palabras separadas. Muestre sus columnas *words* y *tweet*. Luego, aplique al Dataframe training_words la transformación necesaria para obtener una lista de palabras limpias (sin *stopwords*) en la columna clean_words de un nuevo Dataframe training_clean. Finalmente, muestre el contenido del Dataframe train_words, donde se filtra aquellos *tweets* con menos de 3 palabras válidas.

6. Transforme los datos de texto contenidos en la columna clean_words en *features* numéricos usando el transformador Word2Vec, usando 32 dimensiones, y donde la columna de *features* numéricos se llame features. Muestre el contenido del Dataframe final con la columna features.
7. Basándose en [3], la documentación del algoritmo *Random Forest*, entrene un modelo predictor de la columna label usando como vectores de entrenamiento los vectores numéricos de la columna features del Dataframe train_features, y donde la predicción (clase) quede almacenada en la columna prediction. Use 200 árboles de desición para el modelo.

A continuación, repita las transformaciones necesarias para lograr obtener un Dataframe test_features a partir del Dataframe testing, y efectúe la predicción del modelo entrenado en el Dataframe test_features. Para mostrar predicciones, muestre las columnas class, prediction y tweet.

Finalmente, muestre el *accuracy* de su modelo entrenado en el set de test.

## Requisitos de entrega

Para la entrega de este miniproyecto, deberá enviar el *ipython notebook* con todos sus códigos ejecutados, de manera tal que las salidas y/o resultados de cada comando estén visibles en el archivo. Respete el orden de ejecución y los nombres de las variables.

Para esta entrega se proporcionará la plantilla *plantilla_mini_proyecto_3.ipynb*. En ella vendrán los códigos necesarios para la instalación del software base y código por completar según las instrucciones de las actividades.

## Referencias

[1]https://www.kaggle.com/mrmorj/hate-speech-and-offensive-language-dataset

[2]http://spark.apache.org/docs/2.4.6/ml-features.html

[3]http://spark.apache.org/docs/2.4.6/ml-classification-regression.html#random-forest-classifier