WhatsApp

En este ejercicio vamos a analizar los mensajes de un grupo de Whatsapp. En concreto, podremos obtener toda esta información:

- Número de mensajes a lo largo del tiempo
- Número de mensajes según el día de la semana, o la hora del día
- Número de mensajes escritos por cada participante en el grupo
- Palabras más utilizadas en el grupo
- Palabras más características de cada participante en el grupo

Para llevar a cabo el ejercicio, necesitamos un archivo de log de un grupo de Whatsapp. Junto a este notebook encontrarás un fichero de log ficticio que puedes utilizar para desarrollar los ejercicios (generado a partir de los diálogos de la primera temporada de The Big Bang Theory).

El el formato del fichero que genera Whatsapp varía según se trate de la versión Android o iOS. Veamos un ejemplo de cada caso:

- *Android:*

  26/02/16, 09:16 - Leonard: De acuerdo, ¿cuál es tu punto?

  26/02/16, 16:16 - Sheldon: No tiene sentido, solo creo que es una buena idea para una camiseta.

- *iOS:*

  [26/2/16 09:16:25] Leonard: De acuerdo, ¿cuál es tu punto?

  [26/2/16 16:16:54] Sheldon: No tiene sentido, solo creo que es una buena idea para una camiseta.

Como se puede observar, cada mensaje del chat viene precedido por la fecha, la hora, y el nombre del usuario. El formato no es CSV, como en otros ejercicios, sino que es, digamos, más libre. Así, la fecha ocupa el principio de una línea, que puede venir seguida de una coma. Tras un espacio en blanco, viene la hora (que puede contener o no los segundos), seguida de un espacio, un guión y otro espacio, o bien de dos puntos y un espacio, según la versión de Whatsapp. Por último, tenemos el nombre del usuario, acabando en dos puntos, y tras otro espacio, aparece el texto del mensaje.

Tipos

PalabraUsuario:

Propiedades

- Palabra:String
- Usuario: String

Mensaje:

Propiedades:

- Fecha:LocalDate
- Hora: LocalTime
- Usuario: String
- Texto: String

Representación: en el formato Android

Factoría:

- Parse dado un texto con todas las partes del mensaje

Conversacion:

Propiedades:

- Mensajes: *List<Mensaje>*
- MensajesPorPropiedad(Function<Mensaje,P> f): Map<P,List<Mensaje>>, derivada
- NumeroMensajesPorPropiedad(Function<Mensaje,P> f): Map<P,Integer >, derivada
- PalabrasHuecas:Set<String>, compartida, leida de un fichero
- NumeroPalabrasUsuario: Map<String,Integer>, derivada, NPU
- FrecuenciaPalabrasUsuario: Map<PalabraUsuario,Integer>, derivada, FPU
- NumeroPalabrasResto: Map<String,Integer>, derivada, NPR
- FrecuenciaPalabrasResto: Map<PalabraUsuario,Integer>, derivada, FPR
- ImportanciaPalabra(String usuario, Integer umbral): Double. IP
- PalabrasCaracteristicasUsuario(String usuario, Integer umbral): Map<String,Double>, PCU

Notas

$$IP(u,p) = \frac{FPU(u,p) * NPR(u)}{NPU(u) * FPR(u,p)}$$

Asumimos que $FPR(u,p)$ tiene un valor mínimo, caso de que no u no use p, de un valor dado por ejemplo 0.00001

PalabrasCaracteristicasUsuario(usuario,umbral): Un diccionario con la importancia de las palabras para el usuario que tengan frecuencia superior a umbral

Expresiones regulares

De cada línea del fichero habrá que extraer los distintos campos de información (fecha, hora, usuario y texto de cada mensaje). La mejor manera de extraer esta información es mediante el uso de **expresiones regulares**. Estas expresiones nos permiten definir patrones en los que pueden encajar distintos trozos de texto, y extraer información a partir de distintos trozos de la cadena que ha sido reconocida. Cada lenguaje de programación tiene pequeñas diferencias para tratar las expresiones regulares. En el caso de Java el patrón que describe la estructura de los mensajes es:

```
String RE =
"(?<fecha>\\d\\d?/\\d\\d?/\\d\\d?) ,? (?<hora>\\d\\d?:\\d\\d) -
(?<usuario>[^:]+): (?<texto>.+)";
```

- Cada uno de los tramos encerrados entre paréntesis, y con ? seguido de un nombre, se corresponde con un dato que vamos a extraer. Hay cuatro tramos.
- El primer tramo de la cadena, *(?<fecha>\\d\\d?/\\d\\d?/\\d\\d?),* es un patrón que encaja con las fechas que aparecen en los mensajes. Los \\d indican dígitos, y las interrogaciones indican partes que pueden aparecer o no.
- El segundo tramo, *(?<hora>\\d\\d?:\\d\\d),* encaja con las horas.
- El tercer tramo, *(?<usuario>[^:]+)* , reconoce los nombres de usuario. La expresión *[^:]+* significa "cualquier secuencia de caracteres hasta dos puntos excluidos" (es decir, que reconoce todo el trozo de texto que viene antes de los dos puntos).
- Y el último tramo, *(?<texto>.+),* captura el texto de los mensajes.

La forma de trabajar tras declarar el patrón es:

```
String RE =  "(?<fecha>\\d\\d?/\\d\\d?/\\d\\d?) ,?
                        (?<hora>\\d\\d?:\\d\\d) - (?<usuario>[^:]+): (?<texto>.+)";
Pattern pattern = Pattern.compile(RE);
Matcher m = Mensaje.pattern.matcher(mensaje);
Boolean find = m.find();  // si ha encontrado matching o no
Integer n = m. groupCount(); // número de grupos. Deberían ser 4
String fecha = m.group(“fecha”);
String hora = m.group(“hora”);
String usuario = m.group(“usuario”);
String texto = m.group(“texto”);
…
```