INSTITUTO TECNOLÓGICO DE COSTA RICA **II SEMESTRE 2019**
INGENIERÍA EN COMPUTACIÓN
IC-4700 LENGUAJES DE PROGRAMACIÓN

TAREA PROGRAMADA 1 - HASKELL - GANANCIA DE INFORMACIÓN

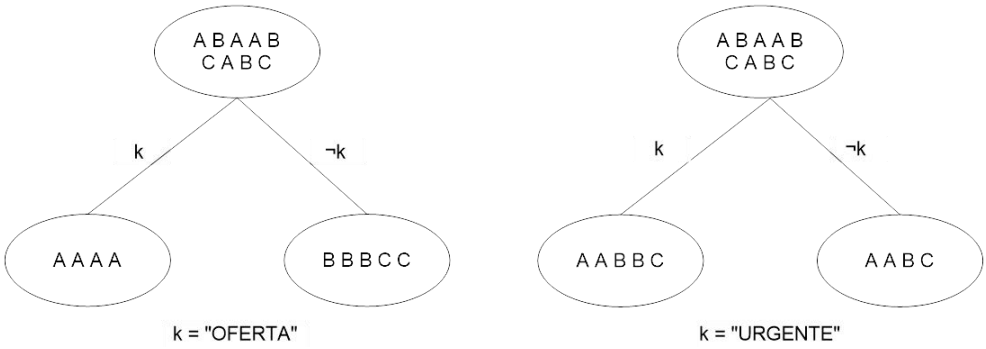
El problema de clasificación en aprendizaje automático (Machine Learning) consiste en asignar (predecir) a qué clase pertenece cada una de las instancias de un conjunto de datos. El conjunto de clases es finito. Por ejemplo, dada una imagen, determinar si lo mostrado es un gato, un perro, un bote, una flor, etc.

La clasificación de texto consiste en asignar a qué clase pertenece cada uno de los documentos de un conjunto de documentos. Es muy útil para filtrar mensajes que son clasificados como spam. Otra aplicación es el análisis de sentimientos, el cual busca determinar la actitud general de un escritor dado un texto que ha escrito. Por ejemplo, se desea tener un programa que tome el texto "**`La película fue un soplo de aire fresco.`**" e indique que se trata de una afirmación positiva; mientras que si recibe "**`Me dieron ganas de sacarme los ojos.`**" indique que es negativa.

Las instancias que son sometidas a un proceso de clasificación son descritas por medio de características (dimensiones) que son usadas por los algoritmos de clasificación. Por ejemplo, para clasificar los clientes de un supermercado, se tienen características como edad, sexo, estado civil, número de hijos, zona de residencia, etc. Un problema que tiene la clasificación de texto, es que cada palabra que aparece en el texto es una dimensión distinta. Esto cual provoca que se tengan miles de dimensiones y que los algoritmos de clasificación no funcionen ni eficientemente ni produzcan buenos resultados.

La Selección de Características es un proceso que selecciona las características más importantes de un conjunto de datos. Hay muchas técnicas y métricas para realizar este proceso. Esta tarea usará la métrica llamada Ganancia de Información (GI) para seleccionar un conjunto reducido de palabras de entre todas las palabras de una colección.

La GI mide el grado en que una palabra reduce el grado de desorden (entropía) de un conjunto de documentos al separarlo en dos: uno con aquellos que tienen dicha palabra y otro con aquellos que no la tienen. Por ejemplo, si se tienen 9 mensajes y se quiere determinar cuál palabra "oferta" o "urgente" es más conveniente para distinguir entre mensajes spam (A), normales (B) e importantes (C), la ganancia de información mide cuán homogéneos son los grupos que se forman al usar una palabra dada para crearlos. Un caso extremo es el que se muestra a la izquierda de la figura siguiente.



Si se usa la palabra "oferta" para separar aquellos documentos que la tienen de los que no la tienen, se forma un grupo en que solo hay mensajes spam y se forma otro grupo en que hay mensajes normales e importantes. Por el contrario, el lado derecho de la figura muestra el caso en que se usa la palabra "urgente", y en ese caso hay mensajes spam, normales e importantes tanto en el grupo que tiene urgente como en el que no lo tiene. En resumidas cuentas, la palabra "oferta" es mejor que la palabra "urgente" para localizar mensaje spam. Si hubiera que escoger solo una de las dos palabras, sería mejor escoger "oferta". La GI de "oferta" es 0.9911 mientras que la de "urgente" es 0.0183.

Cálculo de la ganancia de información

Para calcular la ganancia de información, se deben recolectar los siguientes datos:

n_i	número de documentos en los que aparece la palabra k_i
n_p	número de documentos que son de clase c_p
n_{ip}	número de documentos de clase c_p en los que aparece la palabra k_i
N	número total de documentos

Para el ejemplo anterior los datos serían

		CLASES			
		C_1	C_2	C_3	#Clases=3
		SPAM	NORMAL	IMPORTANTE	n_i
PALABRAS	oferta	4	0	0	4
	urgente	2	2	1	5
	n_p :	4	3	2	9
					N

Hay 9 documentos en total; 4 de ellos son SPAM, 3 son NORMALes y 2 son IMPORTANTEs. La palabra "oferta" aparece en 4 documentos y la palabra "urgente" aparece en 5.

Para calcular GI, primero se debe calcular la entropía H que tiene la siguiente fórmula:

$$H = -\sum_{p=1}^{\#Clases} \left(\frac{n_p}{N}\right) \cdot log_2 \left(\frac{n_p}{N}\right)$$

La ganancia de información se calcula con la fórmula:

$$H(k_i) = (\ (\ n_i \cdot log_2 n_i + (N - n_i) \cdot log_2 (N - n_i)$$
$$- \left(\sum_{p=1}^{\#Clases} n_{ip} \cdot log_2 n_{ip}\right) - \left(\sum_{p=1}^{\#Clases} (n_p - n_{ip}) \cdot log_2 (n_p - n_{ip})\right)\Big) \Big/ N$$

$$IG(k_i) = H - H(k_i)$$

Problema a resolver

Se requiere programar una aplicación en Haskell que haga lo siguiente:

- leer un archivo de reseñas cuyo nombre se especifica por medio de un comando
    - convertir todas las letras a minúsculas
    - cambiar todos los caracteres que no sean letras a espacios en blanco
- contar los valores n_i, n_p, n_{ip} y N
    - n_i: número de reseñas en que aparece la palabra ki
    - n_p: número de reseñas que tienen clase p
    - n_{ip}: número de reseñas de clase p en las que aparece la palabra ki
    - N: número total de reseñas
- descartar las palabras con $n_i < 3$; esto reduce enormemente la cantidad de palabras
- calcular el valor GI para cada palabra distinta (k_i) que aparece en las reseñas
- mostrar las m palabras con los mejores valores de GI; en orden descendente; el valor m es un parámetro escogido por el usuario
- guardar en un archivo las palabras junto con sus valores de GI

Archivo de entrada

Se dispone de un archivo que contiene reseñas de películas del sitio web de Rotten Tomatoes. Cada reseña ocupa una línea. Dicha línea empieza con un puntaje numérico que evalúa la película. Luego, en la misma línea, se incluye a continuación el texto de la reseña. El archivo de datos se ve así:

```
1 A series of escapades demonstrating the adage that what is good for the goo
4 This quiet , introspective and entertaining independent is worth seeking .
1 Even fans of Ismail Merchant 's work , I suspect , would have a hard time s
3 A positively thrilling combination of ethnography and all the intrigue , be
1 Aggressive self-glorification and a manipulative whitewash .
4 A comedy-drama of nearly epic proportions rooted in a sincere performance b
1 Narratively , Trouble Every Day is a plodding mess .
3 The Importance of Being Earnest , so thick with wit it plays like a reading
1 But it does n't leave you with much .
1 You could hate it for the same reason .
1 There 's little to recommend Snow Dogs , unless one considers cliched dialo
1 Kung Pow is Oedekerk 's realization of his childhood dream to be in a marti
4 The performances are an absolute joy .
3 Fresnadillo has something serious to say about the ways in which extravagar
3 I still like Moonlight Mile , better judgment be damned .
3 A welcome relief from baseball movies that try too hard to be mythic , this
3 a bilingual charmer , just like the woman who inspired it
2 Like a less dizzily gorgeous companion to Mr. Wong 's In the Mood for Love
1 As inept as big-screen remakes of The Avengers and The Wild Wild West .
2 It 's everything you 'd expect -- but nothing more .
```

El valor numérico de evaluación tiene el siguiente significado:

- 0: negativo
- 1: algo negativo
- 2: neutral
- 3: algo positivo
- 4: positivo

Se tendrán solo tres clases: Negativo (valores 0 y 1), Neutral (valor 2) y Positivo (valores 3 y 4).

Comandos

El programa se debe implementar como un ciclo interactivo en el que el usuario emite comandos para realizar las diferentes acciones. Se les proveerá de un programa base que muestra como hacer esto en Haskell basándose en el monad IO. A continuación se describen los comandos y sus formatos.

`procesar` *archEnt*	Leer el archivo **`archEnt`** de reseñas; contar los valores n_i,n_p, n_{ip} y N; calcular GI. Si este comando se ejecuta de nuevo, elimina los valores anteriormente almacenados antes de procesar el archivo. Debe imprimir los siguientes datos: • número de reseñas (N) • número de clases • número de palabras

`mejores m`	Imprime las m palabras con los mejores valores de GI, en orden descendiente. Debe imprimir cada palabra y su valor de GI.
`guardar archSal`	Guarda en **`archSal`** todas las palabras junto con sus valores de GI, en orden descendiente.
`fin`	Termina la ejecución.

Sugerencias

- Obtener la solución paso por paso, definiendo una serie de pequeñas y bien definidas funciones conforme se avanza. Esto es considerado como el estilo correcto de programación funcional.
- Definir estructuras de datos adecuadas para almacenar la información que se requiere:
  - Contadores requeridos para los cálculos
  - Valores calculados para las palabras.
- Darle prioridad a la parte "pura" del programa. Esto es, asegúrese de tener primero funciones sencillas que hacen algo así como:

  calcularNlogN(n)

  y generalizar las usando formas funcionales como map para obtener funciones un poco más complejas como:

  calcularH(lista_de_valores_n).
- Luego puede incorporar la parte pura con el ciclo iterativo que hace I/O.
- Ejemplos de funciones que pueden considerar:
  - Función o funciones que dada una reseña va acumulando y actualizando valores en una estructura que permita calcular los valores de las palabras.
  - Función o funciones que para una palabra calcule su GI dados los contadores ya acumulados.
- Aprovechar tipos ya provistos por Haskell como HashMap usado en el programa base.
- Para evitar contar incorrectamente, puede usar un HashMap auxiliar para guadar las palabras que vaya encontrando en una reseña, y al terminar la reseña actualizar el HashMap principal (estado) usando dicho HashMap auxiliar. Debe reiniciar el HashMap con cada reseña.

Consideraciones finales

- Se proveen los siguientes archivos:
  - **`TP1.docx`**, contiene este enunciado.
  - **`TP1-base.hs`**, programa Haskell con ciclo base de ejecución.
  - **`TP1-Ejemplo - Cálculo H.hs`**, programa para calcular entropía (H).
  - **`TP1-Ejemplo - Cálculo GI.xlsx`**, ejemplo cálculo GI.
  - **`TP1-res1.txt`**, archivo de texto que contiene 100 reseñas.
  - **`TP1-res2.txt`**, archivo de texto que contiene 1000 reseñas.
  - **`TP1-res3.txt`**, archivo de texto que contiene 8529 reseñas.

- Usar la plataforma de Haskell provista para alguna de las siguientes arquitecturas:
    - Linux
    - OS X
    - Windows
- La tarea es individual.
- La fecha de entrega es el lunes 30 de setiembre a las 8:00 am.