|  |
| --- |
|  |

|  |
| --- |
| Descubrimiento de Información en Textos  Tarea del Tema 7: Weka. Data Mining with Open Source Machine LearningSoftware in Java |
| **Jose Alberto Benítez Andrades** |
| **71454586A** |
| **Descubrimiento de Información en Textos**  **Máster en Lenguajes y Sistemas Informáticos - Tecnologías del Lenguaje en la Web**  **UNED** |
| **24/06/2011** |

# 0.ENUNCIADO

En la siguiente pagina web http://www.cs.waikato.ac.nz/ml/weka/ el alumno podra encontrar el Software Weka, una coleccion de algoritmos de aprendizaje automatico desarrollados por la universidad de Waikato (Nueva Zelanda) e implementados en Java. Weka contiene las herramientas necesarias para realizar transformaciones sobre los datos, tareas de clasificacion, regresion, clustering, asociacion y visualizacion.

En esta tarea, el alumno debera:

1) Descargar una coleccion de paginas web de la siguiente direccion de correo:

http://nlp.uned.es/~vfresno/banksearch/

Nota: la coleccion esta formada por documentos HTML; aunque en apariencia sean archivos TXT, se trata de documentos HTML con una cabecera antes de la etiqueta inicial. En el nombre del archivo esta codificada la categoria a la que pertenece (A,B,C,..., J)

2) Seleccionar aleatoriamente un 40% de los documentos de la coleccion anterior con los que generar un vocabulario a partir del cual representar el conjunto de la coleccion. A esta subcoleccion la denominaremos traning set, mientras que el 60% restante constituira la subcoleccion test set.

3) Representar la coleccion completa empleando una funcion de pesado TFIDF (o con cualquiera de las funciones de pesado utilizadas en la tarea 5) y con el vocabulario generado en el punto 2). Como la dimension del vocabulario sera muy elevada, debera aplicarse un metodo de reduccion; por ejemplo, eliminando aquellos terminos que aparezcan en un numero muy grande y muy pequeno de documentos de la coleccion.

Nota: Dependiendo de los valores que se tomen como umbral, la dimension sera mayor o

menor. Queda a la eleccion del alumno el establecimiento de estos umbrales.

4) Seleccionar y aplicar un algoritmo de clasificacion de entre los que ofrece Weka, entrenando con el training set y clasificando el test set. La evaluacion debera realizarse usando las medidaF explicada en el tema.

**Documentacion a entregar:**

Un informe en el que se describa brevemente el algoritmo seleccionado, se den los valores de clasificacion obtenidos y se realice un ana.isis critico de los resultados.

# 1. Definiciones

**Matriz de confusión**

También se llama tabla de contingencia. Es de tamaño n\*n, siendo n el número de clases. El número de instancias clasificadas correctamente es la suma de los números en la diagonal de la matriz; los demás están clasificados incorrectamente.

**True Positive (TP) Rate**

Es la proporción de elementos que están clasificados dentro de la clase x, de entre todos los elementos que realmente son de la clase x. Es la parte de la clase que ha sido capturada. En la matriz de confusión es el elemento diagonal dividido por la suma de todos los elementos de la fila.

**False Positive (FP) Rate**

La proporción de ejemplos que han sido clasificados dentro de la clase x, pero pertenecen a una clase diferente. En la matriz de confusión es la suma de la columna de la clase x menos el elemento diagonal menos la suma de las filas del resto de las clases.

**Precisión**

Proporción de ejemplos que realmente tienen clase x de entre todos los elementos que se han clasificado dentro de la clase x. En la matriz de confusión es el elemento diagonal dividido por la suma de la columna en la que estamos. Todas estas medidas son útiles para comparar clasificadores, y las utilizaremos nosotros.

**Algoritmo Decisión Stump**

Es un árbol de decisión con una única división (con una rama adicional para valores no definidos). Es muy efectivo para problemas con dos clases, y para más clases no es fácil conseguir tasas de error menores que 0.5. Es muy simple, pero puede servir como base para comparar el modelo que queremos estudiar o implementar.

El tiempo de cálculo del Decisión Stump es proporcional al número de ejemplos de entrenamiento. Necesita memoria proporcional a Número de clases \* número de atributos \* número de valores Esto puede ser mucho en el caso de atributos con rango continuo de valores (en el peor caso habrá un valor para cada instancia del atributo, aunque se puede utilizar el argumento maxThresholds para controlarlo.

# 2. Resultados con Decision Stump

Iremos comentando el archivo de salida que obtenemos:

El Decision Stump no tiene propiedades editables.

Lo primero que aparece es el algoritmo que se ha utilizado y un resumen del archivo con los datos que vamos a utilizar:

=== Run information ===

Scheme: weka.classifiers.trees.DecisionStump

Relation: C\_\_Users\_Pepe\_Desktop\_DIT-TEMA7\_traning set-weka.filters.unsupervised.attribute.StringToWordVector-R1-W1000-prune-rate-1.0-N0-stemmerweka.core.stemmers.NullStemmer-M1-tokenizerweka.core.tokenizers.WordTokenizer -delimiters " \r\n\t.,;:\'\"()?!"

Instances: 400

Attributes: 1001

[list of attributes omitted]

Test mode: user supplied test set: size unknown (reading incrementally)

Aquí aparece el modelo de clasificación. Recordemos que este modelo es un árbol de un único nivel, así que en el nodo raíz sólo coge un atributo. El mejor que ha encontrado es el ***Java.***

=== Classifier model (full training set) ===

Decision Stump

Classifications

Java <= 0.5 : I

Java > 0.5 : D

Java is missing : H

Class distributions

Java <= 0.5

A B C D E F G H I J

0.11320754716981132 0.10377358490566038 0.10062893081761007 0.0220125786163522 0.0660377358490566 0.050314465408805034 0.11635220125786164 0.14150943396226415 0.14779874213836477 0.13836477987421383

Java > 0.5

A B C D E F G H I J

0.036585365853658534 0.0 0.012195121951219513 0.5 0.2073170731707317 0.18292682926829268 0.0 0.04878048780487805 0.012195121951219513 0.0

Java is missing

A B C D E F G H I J

0.0975 0.0825 0.0825 0.12 0.095 0.0775 0.0925 0.1225 0.12 0.11

Time taken to build model: 0.76 seconds

A continuación aparece el error que tiene nuestro clasificador, según el conjunto de datos de test que hemos introducido. Como preveíamos, no es muy bueno, ya que sólo clasifica bien un 15.667 % de las instancias. Recordemos que en la parte teórica de esta trabajo dijimos que era difícil encontrar una tasa de error menor que 0.5. Aquí tenemos un error absoluto relativo de un 84.3%, lo que no dice mucho a favor de este modelo. El estadístico kappa mide la coincidencia de la predicción con la clase real (1.0 significa que ha habido coincidencia absoluta). Como vemos, este estadístico tiene un valor de 0.07.

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure ROC Area Class

0 0 0 0 0 0.55 A

0 0 0 0 0 0.569 B

0 0 0 0 0 0.569 C

0.865 0.053 0.608 0.865 0.714 0.906 D

0 0 0 0 0 0.566 E

0 0 0 0 0 0.496 F

0 0 0 0 0 0.569 G

0 0 0 0 0 0.567 H

0.942 0.87 0.093 0.942 0.17 0.536 I

0 0 0 0 0 0.558 J

Weighted Avg. 0.157 0.08 0.061 0.157 0.077 0.584

=== Evaluation on test set ===

=== Summary ===

Correctly Classified Instances 94 15.6667 %

Incorrectly Classified Instances 506 84.3333 %

Kappa statistic 0.0766

K&B Relative Info Score 6991.1897 %

K&B Information Score 230.9893 bits 0.385 bits/instance

Class complexity | order 0 2018.7058 bits 3.3645 bits/instance

Class complexity | scheme 2945.0369 bits 4.9084 bits/instance

Complexity improvement (Sf) -926.3311 bits -1.5439 bits/instance

Mean absolute error 0.1722

Root mean squared error 0.294

Relative absolute error 95.5079 %

Root relative squared error 97.6903 %

Total Number of Instances 600

Time taken to build model: 0.76 seconds

Aquí vemos la precisión de nuestro modelo de forma más detallada, especificando por clases:

La matriz de confusión, que nos indica cómo se han clasificado las distintas instancias:

=== Confusion Matrix ===

a b c d e f g h i j <-- classified as

0 0 0 2 0 0 0 0 59 0 | a = A

0 0 0 0 0 0 0 0 67 0 | b = B

0 0 0 0 0 0 0 0 67 0 | c = C

0 0 0 45 0 0 0 0 7 0 | d = D

0 0 0 15 0 0 0 0 47 0 | e = E

0 0 0 8 0 0 0 0 61 0 | f = F

0 0 0 0 0 0 0 0 63 0 | g = G

0 0 0 0 0 0 0 0 51 0 | h = H

0 0 0 3 0 0 0 0 49 0 | i = I

0 0 0 1 0 0 0 0 55 0 | j = J