

PRÁCTICAS de MINERÍA DE DATOS WEB

Curso 2008-2009 2º Cuatrimestre

Pablo.Bermejo@uclm.es

Este curso se realizarán 2 prácticas, una obligatoria y la otra voluntaria:

1. Obligatoria: Clasificación supervisada de noticias
2. Voluntaria: Balanceado y Clasificación supervisada de correo-e.

Material necesario para las prácticas:

1. IDE de Java para la programación.
2. Weka como herramienta de minería de datos.
3. Lucene como manejador de la colección de documentos a indexar y procesar.
4. Clase LuceneToArff.java
5. Clase NewEvaluation.java
6. Corpus MINI_20NEWSGROUPS.
7. Corpus ENRÓN.

Todo el material necesario par las dos prácticas lo podréis encontrar en la página web de la asignatura.

PRÁCTICA 1: Clasificación supervisada del corpus MINI 20NEWSGROUPS.

Introducción:

El corpus MINI_20NEWSGROUPS es un conjunto de ficheros de texto almacenados en el directorio que da nombre a la clase a la que corresponden. Por ejemplo, cada archivo en la carpeta *sci.space* es un archivo de texto plano con un correo electrónico siguiendo un hilo de discusión cuya temática es la *ciencia* (y concretamente el *espacio*).

Este corpus tiene una variable clase de cardinalidad=20 (20 carpetas o temas de discusión) y está totalmente balanceado: todas las clases contienen 100 documentos, así que en este sentido nuestros clasificadores no estarán sesgados.

Preprocesamiento:

Puesto que queremos trabajar con la herramienta Weka, debemos transformar este corpus en un archivo .Arff, donde cada instancia sea un documento y su clase indique la carpeta a la que pertenece. Para realizar esta transformación debéis utilizar la clase LuceneToArff.java que hace posible dicha conversión utilizando Lucene. En el proceso de creación del índice de términos en Lucene ya se han eliminado cualquier referencia dentro de los documentos a la clase que pertenece. Esto se realiza para no alterar las condiciones reales de un clasificador.

Hemos de tener en cuenta que la variable clase que genera LuceneToArff.java es de tipo string, por tanto, debemos de convertirla a nominal mediante el filtro de weka correspondiente.

La base de datos .Arff obtenida representa los documentos utilizando frecuencias. Utilizad esta base de datos para crear otros 2 tipos de representación:

1. Representación vectorial TF-IDF normalizado.
2. Representación binaria (esta última se puede obtener mediante un filtro en weka).

Clasificación Supervisada:

Utilizando el archivo .Arff generado, hemos de validar distintos modelos clasificadores apropiados para las tareas de minería de datos de textos, en las cuales nos basamos en la minería de datos de páginas web o de correos, como es el caso que nos ocupa. Para la validación de los modelos es recomendable utilizar una validación cruzada (5 fold cv). Debemos de tener en cuenta que cada modelo de clasificador debe tener una representación específica de los documentos. Una vez descrito el problema que queremos resolver, nuestro objetivo principal, no podía ser otro, que el de obtener el mejor modelo posible en Accuracy (pctCorrect()), intentando en la medida de lo posible aumentar la tasa de acierto del mismo. Para ello realizaremos los siguientes pasos:

1. Realizar una selección de términos previa, mediante alguna heurística simple, como por ejemplo, eliminar los términos muy frecuentes y los términos raros.
2. A partir de la anterior selección, realiza una selección de términos supervisada mediante alguna técnica adecuada a nuestro problema. Intenta varias selecciones tal que los modelos vean aumentada su tasa de acierto.
3. Realiza un estudio comparativo entre varios modelos de clasificación:
 1. Vecino(s) más cercanos. TF-IDF normalizado. Para este caso, será necesaria modificar o crear una versión de la clase IBk.java de weka, en donde introduzcamos la medida coseno como medida de similitud utilizada.
 2. Bayes Ingenuo. Modelo binario.
 3. Bayes Ingenuo. Modelo Multinomial.
 4. Bayes Ingenuo Complementario. TF-IDF normalizado.
 5. Un clasificador de clase binario todos contra uno, el que veas más apropiado.

Realiza un pequeño informe con los resultados y tareas realizadas en todos los pasos.

PRACTICA 2: Clasificación de Correo-e en carpetas (voluntaria).

Introducción:

El Enron Corpus es un conjunto de correos electrónicos pertenecientes a antiguos trabajadores de la empresa Enron. Tras un escándalo financiero, los correos electrónicos de todos los empleados de la empresa fueron hechos públicos para su estudio por parte de las autoridades.

Para cada empleado, disponemos de todos sus correos almacenados en carpetas tal y como el mismo empleado las definió en su día. Así, la carpeta a la que un correo pertenece puede verse como un posible valor de la variable clase a predecir a partir del contenido de los correos electrónicos.

En el enlace http://www.cs.umass.edu/~ronb/datasets/enron_flat.tar.gz puede descargarse la colección preprocesada de correos de 7 de estos usuarios. El preprocesamiento que encontramos en estas bases de datos es:

- No existen jerarquías de carpetas. Los correos pertenecientes a subcarpetas han sido copiados al primer nivel.
- Las carpetas con menos de 3 e-mails han sido eliminadas.

- El campo *X-folder* de las cabeceras ha sido eliminado ya que contenía el nombre de la carpeta.
- Se han eliminado las carpetas *all_documents*, *calendar*, *contacts*, *deleted_items*, *discussion_threads*, *inbox*, *notes_inbox*, *sent*, *sent_items* and *_sent_mail*.

Para las prácticas se os facilitarán las bases de datos de 7 usuarios convertidas en formato .Arff, donde cada instancia representará un correo electrónico y se encuentran ordenadas en orden temporal de llegada a la bandeja de entrada del usuario correspondiente. La variable clase indica el nombre de la carpeta donde el usuario almacenó el correo-e.

El caso de la clasificación de correo electrónico en carpetas es especial debido a su carácter temporal. Así, modelos típicos de evaluación como Cross-Validation no son válidos pues selecciona los conjuntos de entrenamiento y validación aleatoriamente. En este problema utilizaremos el modelo de evaluación: Time-based Splits Evaluation. Como Weka no ofrece este modelo, hay que implementarlo. Esta evaluación consiste simplemente en entrenar con los primeros N documentos y validar con los siguientes 100, entonces entrenar con los primeros 2N documentos y validar con los siguientes 100; luego entrenar con los primeros 3N y validar con los siguientes 100,... así hasta validar con los últimos 100 o menos documentos.

Como las instancias en las bases de datos que se os entregan ya están ordenadas temporalmente, la evaluación consistirá en crear los conjuntos de entrenamiento desde la instancia 0 hasta la N-1, 2N-1,...y, correspondientemente, los conjuntos de validación con las instancias N hasta N+99, 2N hasta 2N + 99,... No tenéis que implementar esta evaluación, se os dará una clase `newEvaluation` que extiende la clase `weka.classifiers.Evaluation` conteniendo el método `public void timeBasedSplitEvaluation(Classifier c, Instances data, int N)`.

En esta base de datos nos encontraremos con el problema de que el número de documentos por clase (carpeta) no está uniformemente distribuido, y por lo tanto nuestro clasificador aprenderá modelos sesgados a partir de los conjuntos de entrenamiento. Así que compararemos los resultados antes y después de aplicar un filtro de balanceado previo a la clasificación.

Clasificación y Balanceado:

Las tareas a realizar en esta práctica son:

1. Extraer las características de cada usuario:
 - a. Número de carpetas (cardinalidad de la clase).
 - b. Número de instancias
 - c. Número de atributos
 - d. (base,Peak): número menor y mayor de documentos encontrados a lo largo de todas las carpetas.
 - e. (μ : σ): media y desviación típica del número de documentos por clase.
 - f. Coeficiente de Curtosis de Fisher del número de documentos por clase.
2. Realizar clasificación supervisada utilizando: Naive Bayes Multinomial, Support Vector Machine e ibK (k=1). Calcular las siguientes métricas: Accuracy general,

Precisión para cada clase, Recall para cada clase, ROC, AUC. Fijar el parámetro del Time-based Splits Evaluation N=100.

3. Igual que el anterior pero preprocesando los conjuntos de entrenamiento con el filtro `weka.filters.supervised.instance.SMOTE`.
4. Conclusiones:
 - a. Comenta y compara los resultados antes y después de balancear.
 - b. Compara el beneficio de balancear en los distintos clasificadores.
 - c. Comenta qué métrica/s te ha parecido más útil.
 - d. Comenta qué estadística/s de la Tarea 1 te parece más útil para predecir la necesidad de aplicar un preprocesamiento de balanceado.