Trabajo práctico: Filtrado de spam.
Inteligencia Artificial, Grado en Ingeniería Informática

Javier Civera, Universidad de Zaragoza
jcivera@unizar.es

1 Objetivos

1. Entender cómo funciona un sistema de filtrado de spam en correos electrónicos.
2. Aplicar un clasificador de Bayes ingenuo a un problema real.
3. Diseñar e implementar un sistema de detección de spam en Python.
4. Evaluar el sistema implementado en bases de datos públicas.

2 Trabajo práctico

2.1 Bases de datos públicas

Para la resolución del trabajo se usará la base de datos pública Enron-Spam [1], diseñada para el entrenamiento y evaluación de sistemas de filtrado de spam. En concreto se usarán los correos electrónicos preprocesados (denominados como `Enron1`, `Enron2`, `Enron3`, `Enron4`, `Enron5` y `Enron6`). Descarga dicha base de datos y descomprime los correos electrónicos en una carpeta de tu ordenador.

2.2 Python

El entrenamiento del clasificador se realizará mediante Python y se usará el paquete para aprendizaje automático `scikit-learn`[2]. Dicho paquete tiene dependencias con las librerías de cálculo científico `NumPy` y `SciPy`.

Se recomienda la instalación de la distribución Python Anaconda[3], que contiene versiones actualizadas de los paquetes anteriores y otros muchos relacionados con matemáticas, ingeniería y tratamiento de datos.

2.3 Entrenamiento del clasificador

El primer paso consistirá en la lectura de los ficheros que contienen los correos electrónicos y su carga en una estructura tipo lista, que es admitida como entrada por posteriores funciones.

Se adjunta a este enunciado un script de ejemplo (`load_mails_example.py`) que realiza la carga de un subconjunto de correos electrónicos del dataset Enron-Spam. Observa que la función que carga

[1] `http://www.aueb.gr/users/ion/data/enron-spam/`
[2] `http://scikit-learn.org/stable/`
[3] `https://www.continuum.io/downloads`

los ficheros, definida al comienzo del script, separa en datos de entrenamiento, validación y test. Las etiquetas también se definen por separado. Observa como se extraen a la salida de la función. Puedes utilizar este script para añadir a continuación las instrucciones que realizan el entrenamiento y el test. Para ejecutar el script escribe en la ventana de comandos `run load_mails_example.py`

Teniendo los correos electrónicos almacenados en una estructura tipo lista el siguiente paso es construir los descriptores de bolsa de palabras para cada correo. Con los descriptores de bolsa de palabras se entrena el clasificador Naive Bayes.

A continuación se proporciona un conjunto de funciones útiles para extraer la bolsa de palabras y entrenar el clasificador.

2.3.1 Funciones útiles

Listado de funciones útiles para esta parte de la práctica (escribe `help(function_name)` en línea de comandos para obtener ayuda sobre el uso de la función). Consultar la ayuda en la ventana de comandos o en la web para obtener información sobre los argumentos de las funciones.

- `CountVectorizer.fit(...)` inicializa la estructura del modelo de bolsa de palabras (siendo `CountVectorizer` es la clase que implementa la conversión de documentos en descriptores de bolsa de palabras).
- `cv.transform(...)` construye los descriptores de bolsa de palabras para los correos contenidos en `mails`.
- `TfidfTransformer().fit(...)` inicializa una estructura para un modelo de bolsa de palabras normalizado conteniendo la frecuencia de aparición en lugar del número de apariciones (`TfidfTransformer` es la clase que implementa dicha normalización).
- `classifier = MultinomialNB().fit(...)` entrena un clasificador Naive Bayes suponiendo una distribución multinomial. `MultinomialNB` es la clase.
- `classifier = BernoulliNB().fit(...)` entrena un clasificador Naive Bayes suponiendo una distribución de BernoulliNB. `BernoulliNB` es la clase.
- `classifier.predict(...)` predice la clase a la que pertenece un correo electrónico de acuerdo a su bolsa de palabras y habiendo sido entrenado previamente el clasificador `classifier`.
- `classifier.predict_proba(...)` predice las probabilidades de pertenecer a cada clase de acuerdo a su bolsa de palabras y habiendo sido entrenado previamente el clasificador `classifier`.

Se pide realizar el entrenamiento del clasificador usando *K-fold cross-validation* para asignar el hiperparámetro del suavizado de Laplace. Observa que en la definición de la clase de la distribución multinomial `MultinomialNB(alpha=1.0, fit_prior=True, class_prior=None)` el primer parámetro (cuyo valor por defecto es 1.0) corresponde al suavizado de Laplace. Prueba también a construir el clasificador con la distribución multinomial y de Bernoulli y compara sus resultados. Evalúa también la mejora que ofrece un modelo de bolsa de palabras y bigramas con respecto a uno de palabras (la opción para inicializar un modelo de bolsa de palabras y bigramas se encuentra en el constructor de la clase `CountVectorizer`).

2.4 Evaluación del clasificador

Se pide evaluar las prestaciones del clasificador utilizando varias métricas. Para ello, en primer lugar deberemos predecir tanto las clases como las probabilidades de pertenecer a cada clase con las funciones del apartado anterior.

Las métricas que se piden evaluar son la curva precision-recall, f1-score y la matriz de confusión. Observando las métricas selecciona un umbral adecuado para el clasificador de spam y justifica la respuesta.

Aunque la programación para extraer dichas métricas a partir de la salida del clasificador no es muy difícil, el paquete `scikit-learn` tiene funciones para extraer dichas métricas. Para dibujar la curva precision-recall a partir de los datos se puede utilizar el paquete de dibujo de Python `matplotlib.pyplot`. Las funciones para realizar dichas métricas y gráficas se enumeran a continuación, de nuevo consulta la ayuda para obtener información sobre su uso.

2.4.1 Funciones útiles

- `metrics.confusion_matrix(...))`
- `metrics.precision_recall_curve(...)`
- `metrics.f1_score(...)`
- `plt.clf()` borra la figura actual
- `plt.plot()` realiza el dibujo
- `plt.show()` muestra la figura
- `plt.xlabel()`
- `plt.ylabel()`
- `plt.xlim()`
- `plt.ylim()`

Por último, selecciona de entre los correos electrónicos de test ejemplos de spam y ham clasificados correcta e incorrectamente.

3 Informe

El informe del trabajo práctico se someterá via moodle en un fichero comprimido. En el mismo fichero se adjuntarán también los archivos utilizados para la resolución de la práctica, que incluirán un documento `README.txt` con instrucciones para ejecutarlos. La fecha límite de entrega es el **lunes 18 de enero de 2016**. Se realizará una presentación de la práctica de 12 minutos en horario a convenir con el profesor los días 15 o 18 de enero.

El informe y la presentación deben centrarse sobre todo en los aspectos teóricos del problema y evaluación de los resultados, y menos en la parte técnica de la resolución que se evaluará con el resto de archivos sometidos. La longitud máxima del informe será de 4 carillas (2 hojas por las 2 caras).