Trabajo Práctico: Filtrado de SPAM

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 Enron1, Enron2,… 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 instalar 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 Lectura de los correos

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 que realiza la carga de un subconjunto de correos electrónicos del dataset Enron-Spam. Para ejecutar el script de ejemplo escribe en la ventana de comandos `run load_mails.py`

En el trabajo, el entrenamiento y validación del clasificador se realizará con los correos de Enron1 a Enron5, y el conjunto Enron6 se reservará como datos de test.

2.4 Entrenamiento del clasificador

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. Más abajo se indica un conjunto de funciones útiles para extraer la bolsa de palabras y entrenar el clasificador.

[1] http://www.aueb.gr/users/ion/data/enron-spam/
[2] http://scikit-learn.org/stable/
[3] https://www.anaconda.com/download/

Realiza el entrenamiento del clasificador usando K-fold cross-validation para elegir el clasificador que funcione mejor. Para elegir el mejor se usará la tasa de aciertos (accuracy) o el f1-score. Se pide comprar al menos los siguientes casos:

A. Clasificador Naive Bayes con distribución Multinomial y con Bernoulli.
B. Distintos valores del hiperparámetro del suavizado de Laplace. Observa que en la definición de la clase de 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. Observad que se pueden probar valores mayores o menores que 1.
C. **(opcional)** El modelo de bolsa de palabras básico y el modelo de bolsa de palabras y bigramas. La opción para inicializar un modelo de bolsa de palabras y bigramas se encuentra en el constructor de la clase CountVectorizer.
D. **(opcional)** Cualquier otra opción que se os ocurra.

2.5 Evaluación del mejor clasificador con los datos de test.

Para evaluar el mejor clasificador obtenido deberemos predecir **tanto las clases como las probabilidades de pertenecer a cada clase** con las funciones indicadas en 2.6.1. Las métricas que se usarán para la evaluación son:

- f1-score
- matriz de confusión
- curva precision-recall

Aunque la programación para extraer las métricas a partir de la salida del clasificador no es difícil, el paquete scikit-learn ya incluye funciones para calcularlas. Para dibujar la curva precision-recall a partir de los datos se puede utilizar el paquete de dibujo de Python matplotlib.pyplot.

A. Observando las métricas, selecciona un umbral adecuado para el clasificador de spam y justifica la respuesta.
B. Por último, selecciona de entre los correos electrónicos de test algunos ejemplos de spam y ham clasificados correcta e incorrectamente, y discute los resultados.

2.6 Funciones útiles

Escribe help(function_name) en línea de comandos o busca en la web para obtener información sobre los argumentos de las funciones.

2.6.1 Funciones para el entrenamiento

- CountVectorizer.fit(...) inicializa la estructura del modelo de bolsa de palabras, siendo CountVectorizer 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 Bernoulli. 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.

2.6.2 Funciones para la evaluación

- 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()

3 Entrega y defensa del trabajo

El trabajo práctico se entregará vía moodle en un fichero comprimido que contendrá:

1. Un informe de un máximo de 4 páginas en un fichero pdf, que debe centrarse sobre todo en los aspectos teóricos del problema y la evaluación de los resultados obtenidos.

2. Los ficheros programados para la resolución de la práctica, con un documento README.txt con instrucciones para ejecutarlos