Práctica 2. Procesamiento de Lenguaje Natural

Fecha de entrega: 10 de mayo de 2020

Esta práctica vamos a realizar procesamiento de lenguaje natural con la aproximación de bolsa de palabras.

Se elaborará una memoria mediante notebooks de jupyter que incluirán los apartados debidamente señalizados, con su código, la salida del mismo cuando corresponda, y las respuestas a las preguntas planteadas.

IMPORTANTE: Usa la opción **random_state** en las funciones que generan las particiones de datos y que implementan los algoritmos de aprendizaje automático para que los resultados del notebook sean reproducibles en todo momento.

Parte 1. Análisis de sentimiento

En esta parte vamos a usar el fichero "yelp_labelled.txt" que contiene un conjunto de datos de opiniones de restaurantes del recomendador Yelp. Las opiniones está categorizadas como positivas o negativas y hay 500 de cada tipo.

El objetivo es comparar el funcionamiento de distintos clasificadores y entender cómo trabajan estos con texto.

Para ello, vamos a usar distintas opciones a la hora de transformar el texto en bolsa de palabras y observar su efecto en este caso concreto en el que los textos son cortos y escritos con lenguaje informal.

Apartado a)

Configura una partición train-test usando el 75% de los datos para entrenamiento y el 25% restante para test.

Vamos a estudiar varias representaciones de bolsa de palabras, pero todas ellas utilizarán **`countVectorizer`** con el diccionario que se crea a partir de los términos del propio corpus y la lista de palabras vacías (**`stop_words`**) que proporciona sklearn para el inglés. Las **4 posibilidades** que estudiaremos surgen de combinar los siguientes 2 parámetros:

- Bolsa de palabras binaria (usando el **`countVectorizer`** con el parámetro **`binary=True`** y sin usar **`TfidfTransformer`**) y bolsa de palabras con TF/IDF (usando primero el **`countVectorizer`** con el parámetro **`binary=False`**, y sobre el resultado el **`TfidfTransformer`**).
- Usando un rango de n-gramas de (1,1) y de (1,2) (parámetro **`ngram_range`** del **`countVectorizer`**). Es decir, haciendo que la bolsa de palabras se consideren solamente monogramas, o que se consideren monogramas y bigramas.

Para cada una de esas 4 combinaciones entrenaremos dos clasificadores:

1. Un clasificador naive bayes, eligiendo el más adecuado para cada caso.

2. Un árbol de decisión buscando un valor óptimo para uno de los siguientes parámetros para que se maximice la tasa de aciertos en el conjunto de test: **`max_depth`**, **`min_samples_leaf`** o **`max_leaf_nodes`** (siempre el mismo).

Analiza la tasa de aciertos de entrenamiento y test de los 2 clasificadores en las 4 representaciones de bolsa de palabras (8 configuraciones en total) y contesta a las siguientes preguntas:

- ¿Hay un clasificador que sea superior al otro? ¿por qué crees que sucede?
- Para cada clasificador, ¿tiene un efecto positivo el añadir "complejidad" a la vectorización? Es decir, añadir bigramas y añadir tf-idf. ¿Por qué crees que sucede este efecto positivo o la falta del mismo?

Selecciona el mejor árbol de decisión y obtén las 25 variables con más poder discriminante:

- ¿Predominan más las palabras de uno u otro sentimiento? ¿por qué? ¿hay ruido?

Selecciona el mejor clasificador naive bayes y obtén las 25 variables con más presencia en cada clase:

- ¿Tienen sentido las palabras seleccionadas? ¿hay ruido (palabras sin sentimiento o de sentimiento opuesto al esperado)? ¿por qué crees que suceden estos fenómenos?

Finalmente, explica de manera razonada las conclusiones que has extraído de todo el estudio realizado en este apartado.

Apartado b)

Toma el mejor clasificador Naive Bayes y el mejor árbol de decisión para todos los experimentos del caso anterior y analiza a fondo sus resultados en el conjunto de test.

1. Analiza la precisión y la exhaustividad de cada clasificador en cada una de las clases (opiniones positivas y negativas).
    - Para cada clasificador, ¿tiene un comportamiento homogéneo a la hora de clasificar ambas clases?
    - ¿Cuáles son las fortalezas y debilidades de cada uno de los clasificadores?
    - ¿Hay algún clasificador que sea mejor que el otro en todo?
    - ¿Coinciden ambos clasificadores a la hora de clasificar mejor una clase que la otra?
2. Pinta los 8 primeros niveles del árbol de decisión y comenta lo que ves.
    - ¿Qué estructura tiene el árbol?
    - ¿Cómo interpretas los niveles que has pintado? ¿tienen algún sentido con respecto a la tasa de aciertos, o la precisión y exhaustividad del clasificador?
    - ¿Hay nodos impuros?
3. Por cada clasificador identifica 2 críticas que hayan sido falsas positivas (malas críticas calificadas como buenas) y 2 críticas que han sido falsas negativas (buenas críticas clasificadas como malas). Analiza tanto su texto original, como el vector de palabras resultante (solamente los términos activos).
    - ¿Por qué crees que ha fallado el clasificador en cada uno de los casos?

- ¿Se te ocurre alguna idea sobre cómo mejorar el clasificador de sentimiento?

Parte 2. Recuperación de información

Para esta parte utiliza un notebook diferente. Aquí vamos a utilizar el conjunto de datos de 20 Newsgroups que se encuentra disponible en Scikit-learn y que hemos usado en el notebook de ejemplo.

El conjunto consiste en textos de un foro sobre diferentes temas, desde hardware hasta religión. Algunos temas están muy relacionados, p.ej. "IBM PC hardware" y "Mac hardware", mientras que otros son más diversos, p.ej. "religion" o "hockey").

El objetivo de esta parte es poner en poner en práctica los conceptos de recuperación de información para realizar un buscador de mensajes en un foro.

Apartado a)

El conjunto está dividido de forma predeterminada en entrenamiento y prueba en porcentajes de 60 y 40, respectivamente (como se puede ver en el notebook de prueba). Usaremos únicamente la parte de entrenamiento como los mensajes a recuperar por nuestro buscador.

Vamos a utilizar una representación de la bolsa de palabras de **`countVectorizer`** con las siguientes opciones:

- La bolsa de palabras tendrá en cuenta la frecuencia de las palabras en cada mensaje (**`binary=False`**)
- Usa el diccionario que se encuentra en la siguiente URL y que ya usamos en el notebook de prueba. https://github.com/dwyl/english-words/blob/master/words.txt
- Usa la lista de palabras vacías (parámetro **`stop_words)`** que proporciona sklearn para el inglés
- Usando un rango de n-gramas de (1,1) (parámetro **`ngram_range).`**

Para calcular la similitud entre dos mensajes usaremos la similitud del coseno (**`sklearn.metrics.pairwise.cosine_similarity`**) que es capaz de medir la similitud entre los elementos (es decir, entre las filas) de dos matrices de vectores de términos pudiendo ser estas matrices densas o dispersas.

Toma 3 mensaje del conjunto de prueba para cada clase (es decir, para cada tema). Vas a usar cada uno de dichos mensajes como consulta para recuperar los mensajes del conjunto de entrenamiento que más se parezcan a la consulta. Para ello sigue los siguientes pasos:

1. Usa la distancia del coseno entre el mensaje de consulta y los mensajes de entrenamiento.
2. Ordena los resultados de mayor a menor relevancia con la consulta.
3. Calcula la precisión de la lista de resultados con nivel de exhaustividad 3 y 10.
    - La precisión a un nivel de exhaustividad X es el número de resultados que son relevantes (es decir, de la clase buscada) de entre los X primeros recuperados.
4. Calcula los valores de precisión media (para cada nivel de exhaustividad) para cada clase del conjunto de datos.

Se valorará el uso de funciones y la claridad del código, así como sus comentarios.

Contesta a lo siguiente:

- ¿Hay muchas diferencias entre los valores de precisión medios para las distintas clases del conjunto de datos? ¿A qué crees que se deben?
- Identifica la clase que haya tenido peores resultados de precisión y para alguna de sus consultas muestra alguno de los mensajes que recuperó erróneamente en las primeras X posiciones.
    - ¿Con qué clases se ha confundido más dicha consulta?
    - ¿A qué crees que se deben los malos resultados?

Debes usar la parte de entrenamiento para construir la bolsa de palabras con frecuencia y bolsa de palabras con TF/IDF.

Apartado b)

Repite la secuencia de pasos descritos en el apartado a) pero ahora usa TF-IDF para ponderar el peso de los términos de la bolsa de palabras. Para usar TF-IDF primero debes transformar los textos usando **`countVectorizer`** con **`binary=False`** para obtener la frecuencia de palabras (exactamente igual que en el apartado anterior), y a continuación usar **`TfidfTransformer`** para modular dicha frecuencia según lo popular que sea cada término en el conjunto de mensajes de entrenamiento.

A continuación contesta a lo siguiente.

- ¿Han cambiado los valores de precisión media para las clases del conjunto de datos? ¿Qué clases han mejorado? ¿Cuáles han empeorado?
- Encuentra una consulta donde el uso de la ponderación TF-IDF haya sido efectivo y haya mejorado los resultados. Explica por qué ha sido efectivo.

Entrega

La entrega se realizará a través del campus virtual subiendo los dos notebooks de jupyter. En la primera celda de cada notebook debe aparecer el número de grupo y los nombres completos de sus integrantes. Además, el nombre del archivo será P2GXX, siendo XX el número de grupo.