

PRACTICA 4: Clasificador de tweets

- **Twitter** es actualmente una dinámica e ingente fuente de contenidos que, dada su popularidad e impacto, genera enorme interés para realizar distintos estudios de Social Media Analytics. Análisis de reputación de empresas, productos o personalidades, estudios de impacto relacionados con el marketing, extracción de opiniones y predicción de tendencias son sólo algunos ejemplos de aplicaciones que en los últimos años han venido utilizando Twitter como fuente de referencia.

En esta práctica extraeremos contenidos de Twitter, procesaremos los tweets extraídos y realizaremos experimentos básicos de clasificación y análisis de sentimientos.

Pasos a seguir:

1) utilizaremos python-twitter (<https://code.google.com/p/python-twitter/>) que es un wrapper en Python para acceder al API de Twitter. Por tanto, el primer paso consiste en que instaléis la citada librería y os familiaricéis con su documentación.

2) para poder extraer datos de Twitter tenemos que tener una cuenta en Twitter (si no la tenéis o no queréis utilizar vuestra cuenta actual de Twitter para estos propósitos, cread una nueva) y, a partir de esa cuenta de Twitter, tenemos que crear una “Twitter App” asociada a nuestra cuenta Twitter. Una Twitter App es esencialmente un mecanismo que proporciona Twitter para desarrolladores que quieran acceder a los contenidos de Twitter a través de programas. Al crear una Twitter App, Twitter nos proporciona una serie de claves que son las que tenemos que usar en el programa para identificarnos de cara a Twitter cuando queremos extraer tweets de la red social.

Para crear una Twitter App, una vez identificados con vuestra cuenta Twitter, podéis hacerlo en: <https://apps.twitter.com/app/new>. Una aplicación tiene una consumer key, una consumer secret y luego te permite generar “access tokens” (y access token secrets). Estas 4 claves son las claves que tienes que proporcionar en las llamadas al API para obtener datos de Twitter. Nota: la creación de una Twitter App obliga a configurar un número de móvil en la cuenta twitter asociada a la aplicación.

Revisad también los términos de uso de Twitter para desarrolladores (<https://dev.twitter.com/overview/terms/agreement-and-policy>).

Es fundamental respetar los “rate limits” establecidos por Twitter (<https://dev.twitter.com/rest/public/rate-limiting>). Si hacemos más solicitudes por minuto de las permitidas por Twitter, nuestro acceso será bloqueado. Por tanto, al realizar esta práctica asegurarnos de introducir retardos en el código (mediante `time.sleep`) para no sobrecargar a Twitter con más peticiones de las permitidas.

3) El primer objetivo de la práctica consiste en, para un par de personalidades a vuestra elección, descargar el máximo número de tweets posibles de su timeline. Escoged 2 personas con amplia actividad en Twitter y cuyos tweets sean en inglés (porque luego haremos análisis de sentimientos a partir de un lexicon en inglés). Tendréis que realizar varias peticiones al API de twitter (con retardos entre ellas) puesto que en cada petición Twitter te recupera un máximo de 200 tweets. Para realizar esto revisad el método `GetUserTimeline` de python-twitter.

4) Experimento de clasificación de tweets. Para familiarizarse con las estrategias de clasificación de texto, vamos a construir un clasificador automático de tweets que pueda predecir quien ha escrito un determinado tweet. Por ejemplo, si habéis escogido las celebridades Le Bron James y Stephen Curry, se trataría de entrenar un clasificador con un subconjunto de la colección y luego, para un conjunto de datos separados de tests, ver como funciona la clasificación. Por supuesto, es de esperar que en función de lo parecidos que sean los personajes que escojáis, la clasificación sea más fácil o difícil. Por ejemplo, es de esperar que James y Curry, por compartir la misma profesión, hablen habitualmente de baloncesto y, por tanto, la predicción de autoría de un tweet sea más difícil que si hubiésemos escogido, por ejemplo, Barack Obama vs Le Bron James. Pasos a seguir:

- separad el último 20% de tweets de cada personaje como el conjunto de test, que sólo se usará para medir el rendimiento del clasificador.

- con el 80% restante hay que construir el clasificador. Para ello: preprocesad los tweets para extraer una representación “bag of words” de la colección. Al igual que en la práctica anterior, podéis usar scikit-learn (<http://scikit-learn.org/stable/>) y, en particular, sus posibilidades para extraer características a partir de texto (sección 4.2.3 de la página http://scikit-learn.org/stable/modules/feature_extraction.html#feature-extraction) y el Tfidf Vectorizer:

http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

También podéis ayudaros de herramientas como

<https://github.com/myleott/ark-twokenize-py>.

Nota: la estrategia de vectorización ha de ser homogénea en los subconjuntos training-test. Para ello, va a ser necesario quedaros con el objeto vectorizer que uséis sobre el training set para luego aplicarle la misma vectorización a los tweets de test.

Una vez los tweets de training están vectorizados, la representación de la colección es una matriz $N * D$, donde N es el conjunto de tweets de training y D es el número de palabras tras la vectorización. Además, como cada tweet de training sabemos de quien es, podemos construir un vector binario $y=[0,0,1,...]$ (por ejemplo un 0 indicaría un tweet de James y un 1 indicaría un tweet de Curry) y utilizar cualquier estrategia de clasificación en 2 clases. Con esta matriz y este vector “y” hay que construir un clasificador utilizando un clasificador Support Vector Machine (<http://scikit-learn.org/stable/modules/svm.html#classification>). Probad distintos ajustes (distinto parámetro C y distintos kernels), construyendo varios clasificadores y comparando su efectividad en la predicción de los tweets de test. Para predecir la etiqueta de los tweets de test basta con aplicarles la vectorización utilizada en el training e invocar al método predict del clasificador creado en el training. Dadas las predicciones emitidas por el clasificador y las etiquetas reales del test (de quien es realmente cada tweet de test), mostrad una matriz de confusión (http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion_matrix/confusion_matrix.html) y el valor de accuracy (% de decisiones correctas del sistema). Esto debéis hacerlo para las distintas variantes que hayáis creado con la colección de training (distintos C, distintos kernels, etc).

5) análisis básico de sentimiento. Utilizando el lexicon de términos positivos y negativos creado por Bing y Liu (<http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar>), realizad un análisis básico de positividad y negatividad en los tweets de los dos personajes que habéis descargado. Debéis computar al menos el porcentaje de términos positivos y negativos en los tweets de cada uno de los personajes.

Entregables:

- 1) Guión python (.py)

2) Python Notebook (.pynb) (sed particularmente cautos en detallar los resultados de los experimentos, etc)

- Valoración y Fecha de Entrega:
 - Esta práctica tiene una valoración de 4 puntos (sobre el total de 7 puntos de la parte práctica de la materia), que se dividen de la siguiente forma:
 - 3 puntos: programa python correcto y con las funcionalidades arriba indicadas
 - 0.75 punto: extracción de datos de Twitter
 - 1.5 punto: clasificación automática
 - 0.75 puntos: análisis básico de sentimiento
 - 1 punto: python notebook

Fecha entrega: 22 de Diciembre, a las 20h.

Se permiten entregas retrasadas pero se reducirá la puntuación del siguiente modo:

- Entrega entre 1 y 2 semanas tarde: la puntuación recibida será el 70% de la obtenida al corregir la práctica
- Entrega entre 2 y 4 semanas tarde: la puntuación recibida será el 60% de la obtenida al corregir la práctica
- Entrega más de 4 semanas tarde: la puntuación recibida será el 50% de la obtenida al corregir la práctica