Text Mining en Social Media // Master Big Data Analytics

Alberto Ferrer Pérez
alferpre@alumni.upv.es

Abstract

En este trabajo se ha creado un modelo de clasificación, predicción y evaluación del sexo y de la variedad del español de usuarios de twitter a partir de 100 de sus tuits. Se ha usado el DataSet PAN-AP'17, preparado y clasificado en dos particiones, train y test. La aproximación principal al problema ha sido el uso de una bolsa de palabras ponderada basada en TF-IDF (term frequency - inverse document frequency) que da más importancia a las palabras que se usan solamente en un documento, frente a las demás.

1 Introducción

Author Profiling es un clásico problema de inteligencia artificial que se basa en el reconocimiento de características como edad, sexo o variedad del lenguaje del autor de un texto a partir únicamente del uso de las palabras. En el caso presentado en clase, se dispone de 2800 autores clasificados en sexo (male o female) y variedad del español (argentina, colombia, chile, mexico, peru, spain, venezuela). El objetivo es conocer cuál es el sexo y la variedad de un autor, lo cual puede tener muchas implicaciones en campos tan diversos como sociología, politología, marketing, publicidad, etc.

Realizar este trabajo resultaría prácticamente imposible si se hace etiquetando cada autor a mano, por lo que se hace uso de los algoritmos de Machine Learning que automatizan el proceso con unos resultados que, como se verá más adelante, son bastante satisfactorios.

2 Dataset

Se cuenta con el dataset PAN-AP'17 que contiene 2800 usuarios de Twitter para train y 1400 usuarios para test.

- Para cada usuario, se dispone de un fichero id_usuario.xml que contiene 100 tuits escogidos al azar (no retuits) de dicho usuario.

- Tanto para la partición de train como la de test, se dispone de un fichero truth.txt tipo csv separado por el caracter ':::' que contiene la información del id del usuario, el sexo del usuario y la variedad del español del usuario.

Para obtener usuarios de Twitter que respondan a cada variedad, el criterio de extracción fue la localización geográfica de las capitales de cada país. De esa forma, los usuarios cuya localización responda a Bogotá corresponden a la variedad de Colombia, y así sucesivamente con las otras seis variedades del español.

La cantidad de usuarios que pertenece a cada variedad está repartida de forma ecuánime. De los 2800 usuarios de train, 400 corresponden a cada variedad. Y de esos 400, la mitad son hombres y la otra mitad mujeres. Lo mismo ocurre con los 1400 usuarios de test: 200 para cada variedad, de los cuales 100 son hombres y 100 mujeres.

3 Propuesta del alumno

3.1 Baseline

En clase se llevó a cabo una primera aproximación en lenguaje R basada en obtener el baseline. Para ello se creó una bolsa de palabras con los 1000 términos más frecuentes del corpus de entrenamiento. A partir de ahí, se genera una matriz donde cada fila corresponde a un usuario (sus 100 tuits) y cada columna es una palabra de las 1000 más frecuentes. El valor de cada elemento es la frecuencia relativa de aparición de la palabra dentro de los 100 tuits de cada usuario. Se trabaja, por tanto, con una matriz X para la partición de tamaño 2800x1000. Para la partición de test el tamaño de la matriz X será 1400x1000.

Con dicha matriz X de train se entrena un modelo SVM Lineal mediante Cross Validation con 10

folds. Con dicho modelo se efectúa la predicción a partir de la matriz X de test y se evalúan los resultados obteniendo una matriz de confusión. Los resultados de Accuracy obtenidos han sido:

- Sexo: 66.43 %.
- Variedad: 77.21 %.

3.2 TF-IDF

Como intento de mejora de los resultados del Baseline, se llevó a cabo la aproximación mediante bolsa de palabras ponderada con la métrica TF-IDF: *term frequency, inverse document frequency*. Dicha métrica obtiene la medida de la importancia de una palabra en un documento (en nuestro caso, los 100 tuits de un usuario), con respecto al resto de documentos. La hipótesis es que para diferenciar variedades del español, hay palabras que se utilizan mucho en una variedad y son prácticamente inéditas en otras. Palabras que, además, suelen ser las más coloquiales, pertenecientes al lenguaje hablado, que es el mayoritario en Twitter. Lo mismo puede suceder con la clasificación del sexo.

El lenguaje de programación utilizado fue Python, mediante un notebook de Jupyter. Nos decidimos por Python debido a las dificultades para llevar a cabo el modelado de TF-IDF en R.

El primer paso debía ser generar una variable con la que trabajar de forma cómoda, por lo que se decidió generar dos Data Frame de la librería *pandas* de Python, uno para la partición de train y otro para test. Dicho Data Frame constaba de tres columnas: los tuits escritos por un usuario, el sexo y la variedad del español del usuario. A partir de la primera columna se construirá en los siguientes pasos la matriz *X* de entrada al modelo. Y las otras dos columnas supondrán los vectores *y* de soluciones que servirán para entrenar al modelo, en el caso de la partición de train, y para evaluarlo, en el caso de la partición de test.

Para empezar a construir el modelo, lo primero que había que decidir es el tokenizador utilizado. Para ello, la librería *nltk* de Python cuenta con la función *TweetTokenizer* que, como su nombre indica, está preparada exactamente para tokenizar tuits.

TweetTokenizer. Se usaron tres parámetros: *reduce_len = True* que elimina más de tres caracteres iguales juntos; *preserve_case = False* que elimina las mayúsculas; *strip_handles = True* que elimina espacios en blanco.

Tras tokenizar, se debía eliminar las stop words o palabras vacías, que son las más usadas (preposiciones, adverbios, pronombres, etc.) y las que menos información pueden añadir para clasificar la variedad del lenguaje o el sexo del usuario. Para ello, descargamos la librería *stop_words* de Python que contiene la función *get_stop_words*.

get_stop_words. Se le llama con el parámetro 'es' para obtener las stop words del español.

Con el tokenizador preparado y las stop words descargadas, se procede a crear la bolsa de palabras de TF-IDF mediante la función *TfidfVectorizer* de la librería *sklearn*. Tras diversas pruebas, se obtuvo el número de palabras óptimo con el que se consiguieron los mejores resultados.

TfidfVectorizer. Tres parámetros: *tokenizer*, en el que se usa el tokenizador preparado anteriormente; *stop_words*, al que se pasa la lista de stop words obtenida; *max_features*, el número máximo de palabras a utilizar. Tras muchas pruebas, se determinó que el número óptimo de palabras es 20000.

Con el objeto TfidfVectorizer creado, se deben hacer dos cosas. La primera es crear la bolsa de palabras mediante el método *fit*. Y después, a partir de dicha bolsa de palabras, generar las matrices X para la partición de train y para la partición de test con la función *transform*. Como la bolsa de palabras se basa en los usuarios de train, ambas funciones, fit y transform, se pueden ejecutar a la vez mediante la función *fit_transform*.

Tenemos, pues, las dos matrices X de entrada al modelo:

$$X_{train}\ 2800 \times 20000$$

$$X_{test}\ 1400 \times 20000$$

El siguiente paso es entrenar un clasificador, tanto para sexo del usuario como para variedad del español utilizada. Después de diversas pruebas con modelos como Naive Bayes, Suport Vector Machine o Random Forest, se concluye que los mejores resultados se obtienen con un clasificador basado en la técnica Random Forest con 500 árboles.

Random Forest. Es una técnica que se basa en Árboles de Decisión. Un árbol de decisión es un modelo de predicción que, a partir de los datos de las variables de entrada y en función de unas reglas de decisión, clasifica una nueva entrada en una de las categorías posibles, usando para ello nodos y flechas que finalmente componen diferentes caminos. Si se juntan muchos árboles de decisión

se tiene un modelo de agregación llamado Random Forest, que decide la categoría final de una nueva entrada como la categoría más votada de entre todos los árboles que lo componen.

La función en Python que construye un clasificador basado en Random Forest es *RandomForestClassifier* de la librería *sklearn*. Dicha función tiene un parámetro, que es el número de árboles de lo componen. Con *n=500* árboles se obtuvieron los mejores resultados. El método que construye el clasificador es *fit*, que debe recibir como entrada la matriz *X* de la partición de train y el vector *y* de soluciones para que el modelo sea entrenado. Si el vector *y* es el correspondiente a la columna *sexo* del Data Frame, entonces entrenaremos el modelo para clasificar el sexo del usuario. De lo contrario, si el vector *y* corresponde a la columna *variedad*, se entrenará el modelo para clasificar la variedad del español del usuario.

RandomForestClassifier.predict El método predict recibe como entrada la matriz X de la partición de test y, en base a lo entrenado anteriormente con la partición de train, clasifica cada valor de entrada en una de las categorías.

El resultado es un vector de 1400 elementos que pueden valer *male* o *female* en el caso de la clasificación del sexo o *argentina*, *colombia*, *chile*, *mexico*, *peru*, *spain* o *venezuela* en el caso de la clasificación por variedad.

Por último, se deben evaluar los resultados obtenidos. Para ello se usa el método *classification_report* de la función *metrics* de la librería *sklearn*.

metrics.classification_report Con esta función obtenemos un resumen de las métricas para cada categoría y una media final. Las métricas que nos muestra son *accuracy*, *recall* y *f1-score*. En el siguiente punto se presentarán los resultados obtenidos para cada clasificación.

3.3 Medida de la longitud de los tuits

Tras la aproximación con TF-IDF, se decidió probar qué sucede si se tiene en cuenta la longitud de los tuits. Como hipótesis, las mujeres escriben tuits más largos que los hombres. Para ello, partiendo de los dos Data Frames creados en el punto anterior, se añadieron cuatro nuevas columnas: media, mediana, desviación estándar y simetría de las longitudes de los 100 tuits de cada usuario. Tomando únicamente estas cuatro columnas, se introdujeron en el modelo las matrices X:

$$X_{train}\ 2800 \times 4$$

$$X_{test}\ 1400 \times 4$$

Usando un clasificador basado, como en el punto anterior, en Random Forest, se obtuvieron unos resultados algo por encima de lo puramente aleatorio, como se verá en el siguiente punto.

3.4 Bolsa de palabras ponderada por sexo

Como último trabajo, se realizó una exploración de las palabras más usadas por cada sexo, en relación al otro. Un cálculo de TF-IDF hecho a mano que nos condujo a resultados interesantes. Para cada palabra de un sexo, dividimos su frecuencia absoluta entre su frecuencia absoluta en el otro sexo.

$$TFIDF(palabra_{hombre}) = \frac{n_{hombre}}{n_{mujer}}$$

$$TFIDF(palabra_{mujer}) = \frac{n_{mujer}}{n_{hombre}}$$

Ordenando el diccionario de tuplas generado, y filtrando sólo aquellas palabras con un valor de TF-IDF superior a 3.2, se pueden extraer conclusiones interesantes en relación a la diferencia entre ambos sexos. Los resultados se muestran en el siguiente punto.

4 Resultados experimentales

Con la propuesta basada en TF-IDF, los resultados obtenidos fueron bastante satisfactorios si se tiene en cuenta la relación con los que constituyen la Baseline. En la siguiente tabla, se muestra la media de *accuracy* para cada tipo de clasificación:

	Baseline	TF-IDF	Longitud
Sexo	66 %	74 %	54 %
Variedad	77 %	94 %	18 %

Como se ve en la tabla, la bolsa de palabras ponderada de TF-IDF mejora sustancialmente el *accuracy* obtenido.

El primer análisis a la vista de los resultados es pensar que parece obvio que si se tienen en cuenta las palabras por su importancia para un usuario frente a todos los demás, se conseguirán establecer las diferencias entre sexos y entre variedades para una mejor clasificación.

Asimismo, vemos que la clasificación por sexo es bastante más complicada que por variedad del

lenguaje. Cabe pensar que, como se comenta en el siguiente punto, se deben añadir nuevas características al Data Frame de entrada al sistema para hacerlo más rico y así contribuir a la mejora de los resultados.

La propuesta de la longitud de los tuits arrojó unos resultados algo por encima de lo puramente aleatorio. Para sexo, 54 % está por encima de 1/2; y para variedad, 18 % es ligeramente superior a 1/7, que es 14 %.

Por último, cabe mencionar cuáles son las palabras más usadas por cada sexo que se calculó en la seccin 3.4. A continuación, eliminando hashtags o menciones de usuarios, se muestran las palabras más comunes en cada sexo:

Hombres	Mujeres
equipo, 823	mamá, 353
partido, 817	emoticono corazón, 310
fútbol, 361	amo, 263
gol, 301	novio, 214
jugadores, 291	amigas, 199

5 Conclusiones y trabajo futuro

5.1 Conclusiones

Algunas de las conclusiones que extraemos del trabajo realizado es que el problema de Author Profiling es muy interesante, muy bonito y no es en absoluto baladí. Al comenzar, uno se puede encontrar con múltiples problemas que son esenciales para el desarrollo de la tarea. Se ha demostrado que la identificación del sexo del autor de un documento se puede realizar con precisiones por encima del 70%, pero aumentar dicha precisión puede ser un trabajo muy arduo, muy laborioso y casi diría que artesanal. Con la identificación de la variedad del español sí que se pueden obtener grandes resultados y predecirla con una altísima probabilidad.

Además, hemos trabajado mejor y con más facilidad usando el lenguaje Python, por encima de R. Parece que Python, con librerías como *nltk* o *sklearn* se adapta mejor al trabajo con textos.

En cuanto a trabajos futuros o mejoras que se pueden añadir a nuestra tarea, podemos enumeramos:

- En lugar de trabajar con propuestas separadas, es mejor **unirlas**. La hipótesis es que existe una relación de proporcionalidad directa: cuantas más propuestas se añadan, la mejora se agregará y se obtendrá un mejor resultado, siempre y cuando no se mezclen conjuntos de números con medias o desviaciones muy diferentes, para lo cual se deben escalar los valores. Escalar todos los valores en torno a una misma media y con una misma desviacion, se puede realizar a mano o se puede usar una función adaptada para ello como *StandardScaler* de la librería *sklearn*.

- Se pueden usar **n-gramas** de caracteres para identificar el sexo de una persona: frases como ”estoy cansado/a” o ”estoy enamorado/a” se diferencian en un único caracter, que puede ser identificado mediante n-gramas de 2 ó 3 caracteres.

- Añadir **bolsas de palabras temáticas**. Bolsas con palabras propias de *deportes* pueden ayudar a diferenciar el sexo, con la hipótesis de que a los hombres les interesa más el deporte, o la variedad, ya que en España se habla de temas deportivos que seguramente no coinciden con los que se hablan en Colombia o Perú. Además de *deportes*, se pueden añadir bolsas de palabras de *política*, de *sociedad*, de *relaciones humanas*, y cualquier otra que pueda aadir elementos de diferenciación entre sexos o entre variedades lingüísticas.

- Añadir **Emoticonos**. Los emoticonos no fueron tenidos en cuenta pero son un elemento interesante de estudio que pueden ayudar a añadir información en el modelo de clasificación, aunque posiblemente son más importantes si realizáramos un análisis de sentimiento.

Todas estas mejoras unidas en una sola matriz X, escalada, junto con las dos propuestas realizadas en clase, TF-IDF y longitud de los tuits, seguramente mejorarían la precisión de nuestro modelo de clasificación.

References

Jake VanderPlas 2017. *Python Data Science Handbook.*. O'Reilly, Sebastopol.

Henri Laude 2017. *Data Scientist y lenguaje R. Guía de autoformación para el uso de Big Data.*. Ediciones ENI, Barcelona.