Asignatura Text Mining en Social Media. Master Big Data

Adrián Díaz Soler
adri_ds92@hotmail.com

Abstract

El Author Profilling es un método de análisis en plena evolución que consiste en extraer patrones de textos para tratar de proporcionarnos un perfil lo más exacto posible del autor, como pueda ser su edad, género, nacionalidad...etc.

En este caso vamos a tratar de pronosticar la variedad del lenguaje nativo de unos tweets determinades, pertenecientes al corpus del PAN-AP 2017, una competición internacional sobre perfilado de autores.

Nos centraremos en crear diccionarios locales de modismos para cada una de las 7 variedades de lenguaje que tenemos identificadas y, aplicando diversos algorítmos de machine learning, trataremos de encontrar el mejor índice de acierto en este problema de clasificación.

Una vez encontrado es modelo que nos da un mejor resultado, aplicaremos la bolsa de palabras que hemos modificado y observaremos que resultados obtenemos y lo compararemos con el inicial. Que nos permitirá exponer las conclusiones sobre la realización de la tarea.

1 Introduccin

El Author Profiling es un campo que ya está obteniendo buenos resultados a la hora de ser capaz de diferenciar entre géneros, pero que siguen en pleno desarrollo a la hora de analizar otras variables como la edad del autor o su nacionalidad. El porqué de que este método está tan en auge, se debe a sus aplicaciones de seguridad como pueda ser la detección de potenciales abusadores sexuales que tratan de captar a menores haciéndose pasar por otro género o tener otra edad, como la lucha antiterrorista tratando de obtener perfiles de posibles personas que encajen con el de un terrorista.

Obviamente para detectar esas cosas se requiere un nivel muy avanzado en este campo, en este caso práctico de Author Profiling, se ha reducido simplemente a una tarea de identificación de género y variedad lingüística del autor. Para ello disponemos de los datos de PAN-AP 2017, que se trata de una competición a nivel mundial donde el objetivo es resolver un problema que varía cada ao la clase de clasificación que hay que hacer, por ejemplo, puede ser solo de género, solo de variedad, de edad, combinación de variables...etc.

Los datos que nos proporcionan son un corpus reducido de tweets escritos por autores de siete países distintos de habla hispana, de los cuales deberemos conseguir clasificar a qué país pertenecen los autores y de qu género son, en base a lo que han escrito en cada tweet.

Podemos abordar el problema de distintas maneras,por un lado contamos con una bolsa de palabras que extraemos del propio corpus que utilizaremos para que el modelo aprenda, contrastando los textos a clasificar con textos ya clasificados correctamente. Por otro lado, está el tipo de modelo de machine learning que aplicamos para nuestro problema ya que cada uno puede darnos un mejor o peor resultado, pero no por ello ser fiable para nuestra tarea en concreto.

Para la tarea contamos con cinco horas para primero examinar nuestro dataset, y posteriormente plantear cómo vamos abordar el problema y llevarlo a cabo con las pruebas necesarias. Y finalmente comprobar los resultados obtenidos y aportar nuestras conclusiones al respecto.

2 Dataset

El dataset que utilizamos en la tarea como ya hemos comentado, son los mismos que los del

PAN-AP 2017, y se trata de un conjunto de archivos en formato XML, cada uno perteneciente a un individuo en concreto con cien tweets publicado por dicho individuo. En total se trata de 2800 archivos para el training, es decir, disponemos de 280 mil tweets(unos 34 MB de texto) para analizar y entrenar con nuestros modelos para poder realizar una correcta clasificación.

Para unir cada archivo individual en uno solo que es lo que necesitamos a la hora de trabajar, debemos hacer una función que lea primero todos los archivos del tipo XML, y posteriormente los metamos en una variable todos juntos de la siguiente manera:

files = list.files(pattern="*.xml")

corpus.raw ¡- NULL i ¡- 0 for (file in files) xmlfile ¡- xmlTreeParse(file, useInternalNodes = TRUE) corpus.raw ¡- c(corpus.raw, xpathApply(xmlfile, "//document", function(x) xmlValue(x))) i ¡- i + 1 if (verbose) print(paste(i, " ", file))

Una vez hecho esto, ejecutamos el modelo que se nos ha proporcionado con una bolsa de palabras dada, que se ha extraído del propio corpus, en base a las palabras más frecuentes y con un modelo predictivo de Super Vector Machine, para las 100 palabras más frecuentes. Y el resultado que obtenemos es el siguiente:

N	GENDER	VARIETY	TIME
10	0.5875	0.2608	3.62m
50	0.6850	0.3167	4.32m
100	0.7375	0.3383	5.36m
500	0.7358	0.5717	9.16m
1000	0.6983	0.6167	12.11m
5000	0.7550	0.6167	51.81m

Como observamos el accuracy para el género es relativamente alto, en cambio para la variedad del lenguaje es muy bajo, por ello nos hizo plantearnos en centrarnos en el problema de la variedad y tratar de mejorarla dado el tiempo del que disponíamos.

3 Propuesta del alumno

Una vez hemos decidido que optamos por centrarnos en mejorar la variable de variedad del lenguaje, debemos plantear cómo hacerlo.

La primera idea es centrarnos en la bolsa de palabras que tenemos, al tratarse de las palabras más frecuentes de los tweets y que se trata del mismo idioma, el castellano, aunque hayan 7 países distintos observamos que las palabras son de lo más común en todos los países y muy complicado que sirvan para diferenciar de qué país proviene cada uno. Por ello decidimos que debemos mejorar la bolsa incluyendo palabras típicas de cada país.

Para obtener estas palabras recurrimos a internet para encontrar los modismos o palabras más usadas en los países que tenemos que son: México, Colombia, Venezuela, Perú, Chile, Argentina y Espaa. Una vez obtenidas las palabras, las guardamos por separado en un csv para cada país y de esta manera poder cargarlo a nuestro dataset para generar la bolsa de palabras.

Pero no solo nos basta con guardarlas en csv conforme las encontramos, hemos tenido que aplicar un pequeo preproceso, eliminando modismos duplicados y también eliminando aquellos que se trataban de expresiones combinadas, quedándonos solo con palabras únicas.

Además, tuvimos que hacer una pequea transformación en estos diccionarios, cambiando todos los modismos para que empezaran por minúscula y adaptando estos nuevos datasets para que las variables tuvieran el mismo nombre y contaran con el mismo número de columnas.

Pero para ver que vamos bien encaminados en nuestra idea, hacemos la comprobacin con uno de nuestros particulares diccionarios para ver si el uso de este mejora la clasificación de los autores de dicho país. Por ejemplo lo hacemos con Chile.

Usando 100 palabras de Vocabulary:

STATITICS	CLASS
	CHILE
Sensitivity	0.49500
Specificity	0.88500
Pos Pred Value	0.41772
Neg Pred Value	0.91316
Prevalence	0.14286
Detection Rate	0.07071
Detection Prevalence	0.16929
Balanced Accuracy	0.69000

Accuracy : 0.5229

Usando listado de 102 palabras de Chile:

STATITICS	CLASS
	CHILE
Sensitivity	0.49500
Specificity	0.96083
Pos Pred Value	0.67808
Neg Pred Value	0.91946
Prevalence	0.14286
Detection Rate	0.07071
Detection Prevalence	0.10429
Balanced Accuracy	0.72792

Accuracy : 0.3121

Comprobamos que efectivamente el accuracy general disminuye a la hora de clasificar a las 7 pases pero en el caso de Chile mejora individualmente al usar el diccionario con los modismos que hemos obtenido de este pas.

Por tanto, decidimos seguir adelante con nuestra idea de unir todos los diccionarios que hemos buscado para que sea nuestra bolsa de palabras conjunta y mejore el nivel de clasificacin del modelo. Para crear la lista conjunta simplemente unimos cada lista con un rbind:

venezolano ¡- read.table(" /listas/venezolano.csv", quote="")
colnames(venezolano) ¡- c("WORD")
venezolano$FREQ ¡- 0
peruano ¡- read.table(" /listas/peruano.csv", quote="")
colnames(peruano) ¡- c("WORD")
peruano$FREQ ¡- 0
argentino ¡- read.table(" /listas/argentino.csv", quote="")
colnames(argentino) ¡- c("WORD")
argentino$FREQ ¡- 0
chileno ¡- read.table(" /listas/chileno.csv", quote="")
colnames(chileno) ¡- c("WORD")
chileno$FREQ ¡- 0
colombiano ¡- read.table(" /listas/colombiano.csv", quote="")
colnames(colombiano) ¡- c("WORD")
colombiano$FREQ ¡- 0
mexicano ¡- read.table(" /listas/mexicano.csv", quote="")
colnames(mexicano) ¡- c("WORD")
mexicano$FREQ ¡- 0
espannol ¡- read.table(" /listas/espannol.csv", quote="")
colnames(espannol) ¡- c("WORD")
espannol$FREQ ¡- 0

vocabularios7 ¡- rbind(venezolano, peruano, argentino, chileno, colombiano, mexicano, espannol)

Y si conseguimos tener un resultado positivo con la bolsa de palabras, pasaremos a probar distintos modelos de machine learning para ver si hay alguno que nos ofrezca tambin mejores resultados.

4 Resultados experimentales

Una vez vomprobado que las listas de los modismos si que mejoran la clasificación para cada país decidimos comprobar de varios modelos de machine learning, cual nos aporta un mayor accuracy para el vocabulario dado usando solo 100 palabras.

- Usamos primero Support Vector Machine, que es el modelo que tenemos cuando se nos proporciona el modelo. Obtenemos los siguientes datos:

  Accuracy : 0.5229.

  95% CI : (0.4963, 0.5493).

  No Information Rate : 0.1429.

  P-Value [Acc ¿ NIR] : ¡ 2.2e-16.

- A continuación usamos Random Forest:

  Accuracy : 0.5393.

  95% CI : (0.5128, 0.5656).

  No Information Rate : 0.1429.

  P-Value [Acc ¿ NIR] : ¡ 2.2e-16.

- Ahora usamos K-Nearest Neighbors:

  Accuracy : 0.345.

  95% CI : (0.3201, 0.3706).

  No Information Rate : 0.1429.

  P-Value [Acc ¿ NIR] : ¡ 2.2e-16.

- Usando Naive Bayes:

  Accuracy : 0.4214

  95% CI : (0.3954, 0.4478)

  No Information Rate : 0.1429

  P-Value [Acc ¿ NIR] : ¡ 2.2e-16

- Usando Nearest Neighbour:

  Accuracy : 0.43

  95% CI : (0.4039, 0.4564)

  No Information Rate : 0.1429

  P-Value [Acc ¿ NIR] : ¡ 2.2e-16

- Usando Classification Tree C5.0:

  Accuracy : 0.5071

  95% CI : (0.4806, 0.5337)

  No Information Rate : 0.1429

  P-Value [Acc ¿ NIR] : ¡ 2.2e-16

Como podemos observar, el modelo que mejor resultados nos ha dado de base usando únicamente 100 palabras del vocabulario base, es el Random Forest con un accuracy de 53,93%.

Una vez elegido el modelo,le aplicamos nuestra bolsa de palabras que consta de 416 palabras únicas propias de cada pas mas 500 palabras del vocabulario inicial, y de esta manera llegamos a conseguir con Random Forest un accuracy del 87,14%.

Accuracy : 0.8714
95% CI : (0.8528, 0.8885)
No Information Rate : 0.1429
P-Value [Acc ¿ NIR] : ¡ 2e-16

En definitiva, hemos conseguido mejorar la clasificación de la variedad del lenguaje desde un 33% hasta un 87% enriqueciendo la bolsa de palabras y el número de palabras usadas, como del modelo de machine learning utilizado para la casificación.

5 Conclusiones y trabajo futuro

Como concluisión de la tarea realizada, podemos decir qu eexisten muchas variables que nos permiten modificar el modelo de clasificación. Alugnas de ellas son más técnicas que solo requieren prueba y error, como la elección del modelo que utilizar, y otras variables que requieren imaginación e ideas originales que nos permitan enriquecer el modelo o ayude a entrenar el modelo, como el uso de una bolsa de palabras personalizada por cada país.

Como altenrativa y trabajo futuro, hubiera sido interesante aadir a los diccionarios nombres de personajes famosos y políticos de cada país, ya que al visualizar algunos tweets del corpus habían indiviuduos que nombraban a dichos personajes y podrá ser un factor que ayude a mejorar la clasificación del autor .

Complementariamente trabajaramos el aspecto gnero. Ya que, como comentamos inicialmente, hemos planteado el problema de clasificacin nicamente para el aspecto variedad, omitiendo el gnero ya que su accuracy inicial era ms alto. Pero podramos trabajarlo, planteando hiptesis como las siguientes, que tendramos que contrastar:

Los tweets ms largos estn escritos por mujeres. Los tweets con ms nmero de emoticonos estn escritos por mujeres.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.