

Paper. Asignatura Text Mining en Social Media. Master Big Data

Ramón Ferrer Mestre
rafermes@inf.upv.es

Abstract

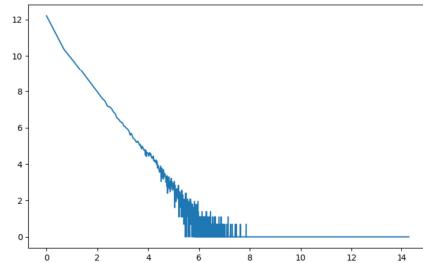
La tarea trabajada en este proyecto consiste en la identificación de género y nacionalidad a partir de un dataset compuesto por 300 autores y 7 variedades lingüísticas. Esta tarea se ha abordado mediante una metodología crisp-dm. Partiendo de la comprensión previa de los datos

1 Introducción

El problema de clasificación de autores mediante frases es complejo en comparación con otros problemas de clasificación debido a que el input es mayormente desconocido apriori. Por este motivo durante esta tarea se ha tenido en cuenta para la evaluación que esa va ha ser la situación natural del modelo en producción. Con el enfoque en ese punto, la distribución del dataset no es aleatoria como podría ser natural en otros problemas. En este caso la división es por autores, de forma que no solo contemplamos la realidad de no tener todas las expresiones y palabras sino también el reconocer que en la realidad del problema, los usuarios finales no tienen porque estar en el conjunto de entrenamiento.

2 Dataset

En este dataset contamos con 20063 palabras o tokens distintos después de pasar por la normalización que elimina los emojis y los caracteres extraños que quedan fuera del alfabeto castellano y no son números. De estas palabras aparecen 1014 grupos distintos en cuanto a numero de veces que aparecen en todo el dataset. En la siguiente gráfica podemos observar el numero de veces que aparecen y el numero de palabras pertenecientes a este grupo. Como se aprecia con facilidad, la gran mayoría de palabras aparecen en los grupos con menos apariciones.



Podría ser interesante observar el mismo comportamiento con los n-gramas de las m palabras con más apariciones. De esta forma eliminamos una gran cantidad de 0 en las matrices de incidencia de los n-gramas y concentrarnos las combinaciones de palabras más comunes de la combinación de autores del train. De todas formas como esto puede llevar a un problema de overfitting de la forma de escribir de estos autores, dado que sería sencillo que las palabras mas repetidas fueran muletillas o expresiones típicas de este autor o autores en el caso de expresiones regionales, esta parte se dejará a implementar y validar en trabajos futuros.

3 Propuesta del alumno

Dada la metodología explicada anteriormente y el expuesto punto de partida con un dataset semiestructurado, el primer enfoque tomado es la realización de una comparación entre distintos modelos, apartir de diferentes tecnologías y transformaciones.

En principio las tecnologías utilizadas serán para r las librerías qdat,tm,xml,splitstackshape y caret, para python las librerías sklearn, nltk,json,re, numpy y csv, y para java las librerías de org.w3c.dom y vertx.

El proceso de desarrollo consiste en la generación de tres scripts que se encarguen de los tres puntos claves para la modelización y evaluación de los modelos.

El primer script del proceso tiene la finalidad de

lectura de los xml, creación de la bolsa de palabras y guardado de esta en formato abierto para desacoplar de la tecnología utilizada para este punto del proceso.

El segundo script tiene como objetivo la vectorización de las frases y el guardado de los vectores con los ids de autor y sus etiquetas.

El tercer script se responsabiliza de leer los vectores de test y train, entrenar el modelo que se desea probar y evaluar este modelo utilizando los datos de test.

En este punto es interesante explicar la división entre train y test, esta división nos viene dada en el punto anterior al inicio de este proyecto. La división es de 200 autores para train y 100 usuarios para test, de forma que evaluación sea de frases desconocidas, con palabras desconocidas y se evaluará sobre estilos de escritura personales diferentes. De esta forma se busca que los mejorar la predicción sin tener que preocuparnos del sobreajuste por la forma de escribir de los autores del train.

Para el primer script se ha decidido usar java para la lectura de xml y conversión a json, y python para la creación de la bolsa de palabras. Una vez hecho esto se ha procedido a la vectorización de los conjuntos de train y test mediante python, dada la cantidad de memoria ram requerida para tener estos conjuntos en memoria, se ha decidido reducir el numero de palabras por vector a las 500 palabras más usadas. Por otro lado en R se ha creado un script único que realiza el trabajo de los tres scripts por motivos de eficiencia.

La primera decisión importante es el método para seleccionar las palabras que formarán parte de la bolsa de palabras. Una de las propuestas ha sido crear una bolsa de categorías gramaticales en lugar de una bolsa de palabras mediante un tagger TNT preentrenado y con un accuracy del 90 porciento. Esta propuesta fue descartada después de su implementación debido a que el tiempo de etiquetado de este tagger oscilaba entre una hora y dos horas por frase, siendo para el numero de frase una opción realmente inviable.

La segunda propuesta fue agregar un catalogador de emojis para traducirlos a palabras que expresarán sentimientos, pero en la fase de análisis de esta opción resultó que el tiempo de preparación de las frases incrementaría con la latencia de las peticiones web al servicio de catalogación y con el parseo del html respuesta. Ademas

esta propuesta corría el riesgo de ser baneados por denegación de servicio.

La tercera propuesta fue partir de una tokenización donde se normalizarán las palabras, eliminando acentos y caracteres extraños. En este punto se decidió no usar filtros de stop words ni lemmatización porque nuestra hipótesis se basa en que la mayor diferencia radica en la forma de usar los verbos y los conectores.

Una vez decidido el método para la creación de palabras, la selección se ha decidido por numero de apariciones por palabra en python y por la frecuencia del termino en r.

Después de finalizar esta etapa, se decidió buscar los modelos que utilizado normalmente en este tipo de problemas. De esta búsqueda encontramos el artículo Improving Gender Classification of blogs authors[1] donde se comparan el SVM con el naive bayes, en este artículo se vectoriza usando el algoritmo EFS pero en nuestro caso se ha considerado que la prueba de esta vectorización como un trabajo futuro. Otro artículo que hemos encontrado, Automatic Turkish text Categorization Terms of Author, Genre and Gender[0], utiliza los anteriores y el random forest. Por otro lado, se ha decidido buscar modelos que funcionen bien en problemas con gran cantidad de dimensiones, en este caso se han seleccionado el extra trees, el kmeans y el ranger[2]. Por otra parte, se ha decidido probar un KNN para diversificar las pruebas.

4 Resultados experimentales

Se ha tomado el accuracy como medida de accuracy para el proyecto y común para evaluar las propuestas de todos los miembros del equipo. Además se ha probado a utilizar una medida de accuracy mediante crossvalidation de 10 particiones, para comprobar el grado de overfitting de los modelos kmeans y extratree sobre los autores de train. El primer dato interesante es que en ambos modelos se ha llegado a un ochenta porciento de accuracy cuando las muestras de test y train pertenecen a los mismos autores, en cambio cuando aplicamos estos modelos al test real, donde los autores son distintos, el accuracy baja hasta el cincuenta por ciento de acierto aproximadamente, de forma que se ha dado verificado la necesidad de testear con autores externos al train.

5 Conclusiones y trabajo futuro

En cuanto a resultados prácticos, los mejores resultados han sido dados por el aumento de palabras introducido en los modelos, aunque las mejoras mas remarcables han sido en los modelos de random forest y ranger[2]. Por lo tanto para este tipo de input y este problema resultan ser, para el criterio de evaluación utilizado, los mejores modelos.

Por otra parte como trabajo futuro a corto plazo es interesante el realizar pruebas con diferentes métodos para reducir la dimensionalidad y observar los nuevos resultados en los modelos probados.

Como trabajo futuro a medio plazo se debería poder mejorar el rendimiento del tagger o usar otro tagger de lengua española, en el peor de los casos se podría intentar usar un tesauro para realizar esta prueba de concepto. Dado que esto excede en el tiempo y los recursos actuales, se plantea a medio plazo como una prueba de concepto. Por otra parte el uso de los emojis y caracteres eliminados debería revisarse para poder enriquecer los vectores en base a estos valores. Ya sea usando una traducción de emoji/caracter especial a palabra a la que se refiere o clasificando por abstracción de su significado(es decir si corresponden a términos abstractos o no).

En cuanto a trabajo a largo plazo, dada su complejidad y la necesidad de afrontar como un proyecto nuevo. Se enmarcan las siguientes ideas:

1. Generar un buscador de sujetos y objetos para las frases teniendo en cuenta las oraciones pasivas. Esto puede realizar un prefiltro útil si el sujeto de la oración es el autor o el autor es el objeto de esta.
2. Construir la matriz de incidencia de todas las palabras, que se valoren como importantes mediante el método que se decida, afrontando el problema de la gran cantidad de columnas que se van a generar y la gran cantidad de memoria que será necesaria para el entrenamiento del modelo
 - Para esta idea se pensó en distribuir los datos y generar bolsas de palabras locales a las muestras de test de cada nodo, después se genera la vectorización imitando el protocolo gossip de cassandra, de forma que todas las muestra de train pasen por todas las bolsas locales.

Esto tendría el problema del orden de las columnas pero es fácilmente resoluble si fijamos un orden de las columnas en el momento en el que se acaba de montar el vector. Este planteamiento no resolvería el problema de memoria a la hora de entrenar solo a la hora de vectorizar.

- Otra idea que se ha propuesto es reducir la matriz de incidencia una vez calculada al numero máximo de unos por vector y tomar este numero como el numero de columnas y entregar a las columnas los valores de las posiciones en el vector. El problema de este planteamiento es que se necesita calcular la matriz y conocer el numero maximo de incidencias en un vector de toda la matriz.

Finalmente es necesario exponer los problemas encontrados en el proyecto. Entre los problemas encontrados, durante la fase de pruebas se intento utilizar el fit-transform del lda de la librería scikit-learn pero no se pudo aplicar por problemas de compatibilidad con la infraestructura para el desarrollo del proyecto. Otro problema importante es la limitación de la consola de r para tratar los volúmenes de datos impidiendo pasar de vectores de 1000 palabras. Y la necesidad de obligar a python a utilizar el garbage collector para mejorar la cantidad de ram que se podria ocupar para las operaciones de training.

References

M. Fatih AmasyaliBanu Diri 2006. *NLDB 2006: Natural Language Processing and Information Systems pp 221-226 Automatic Turkish text Categorization Terms of Author, Genre and Gender*

Arjun Mukherjee y Bing Liu University of Illinois at Chicago, Chicago, IL 2010. *EMNLP '10 Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, Pages 207-217 Improving Gender Classification of blogs authors*

Marvin N. Wright [aut, cre], Stefan Wager [ctb], Philipp Probst [ctb] 2018. <https://cran.r-project.org/web/packages/ranger/ranger.pdf> A Fast Implementation of Random Forests