UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Lenguaje Natural y Representación de la Información

Memoria: Detección y Clasificación de Estereotipos Raciales en Español

Grado en Ciencia de Datos

DATTABAYÖ
Borja Esteve Molner
Jorge López Fresco

0. Índice de contenido

1. Preprocesamiento

Para realizar el preprocesamiento general, creamos una lista de palabras tokenizadas, con el texto original y su lema, teniendo en cuenta el tipo de elemento sintáctico al que se refiere. Nuestro objetivo era observar la estructura de las frases y su tipología. Así, prodíamos saber qué elementos mantener y cuáles eliminar u omitir. Por su parte, realizamos una normalización y convertimos todas las palabras a minúsculas. Además, eliminamos signos de puntuación, espacios y caracteres raros.

En lo que respecta a las StopWords, realizamos una prueba empírica para decidir si eliminarlas o no de la representación. Dicha prueba consistía en observar si existían diferencias significativas entre la representación de las oraciones al incluir o no las stopwords. Es decir, queríamos comprobar lo siguiente:

H_0: $\overline{SW} = \overline{No\ SW}$. *Media de vectores embedding son iguales incluyendo SW o no.*

H_1: $\overline{SW} \neq \overline{No\ SW}$. Media de vectores embedding son distintas incluyendo SW o no.

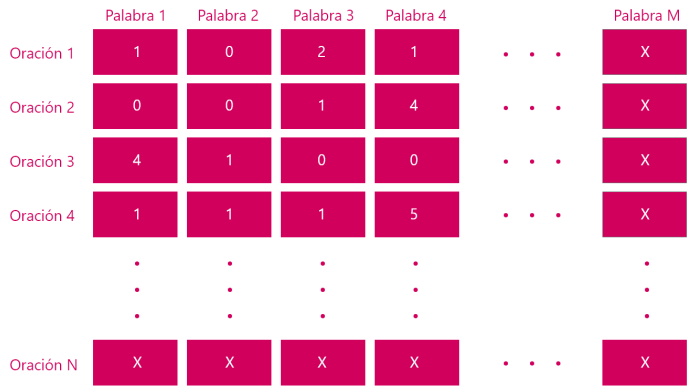
Usamos un valor muy pequeño (epsilon) para tomar la decisión, que medía la diferencia entre el valor máximo y el mínimo de los vectores medios del word embedding de las palabras. Dado que observamos que más de un 13% de las oraciones mostraban diferencias significativas (lo que implica que tienen importancia), decidimos tener en cuenta las StopWords, incluyéndolas en el estudio.

2. Representación de textos

2.1 Sistema de Ponderación

De cara a mejorar la representación posterior de los Word Embedding, creamos un sistema de ponderación basado en la asignación de pesos en función de las diferencias existentes en la distribución de frecuencias entre las palabras de la *clase 'estereotipo'* y las de la *clase 'no estereotipo'*.

Del mismo modo, obtuvimos para cada frase la cantidad de veces que una palabra ocurría en cada oración y lo representamos en una matriz, en la que cada fila era una oración del dataset y cada columna una palabra en el vocabulario general del dataset.



Por su parte, creamos una matriz de ceros con tantas filas como oraciones y tantas columnas como palabras y actualizamos los valores de la matriz en función de las frecuencias de cada palabra en la oración.

Por otro lado, añadimos la columna con los valores de los estereotipos al final de la matriz y de esta forma, podíamos

conocer la media y la frecuencia de la ocurrencia de una palabra específica en ambas clases. Para hacerlo más accesible, creamos un diccionario con cada palabra y una lista asociada con sus ocurrencias en cada clase, la proporción de no estereotipo, la proporción de estereotipo y el total en valor absoluto . El diccionario tenía la forma:

```
{ word: [no_estereotipo, estereotipo, prop_no_est, prop_est, total] }
```

Con estas variables hicimos un test para determinar si una palabra tenía una distribución diferente en una clase con respecto a la otra. De esta forma:

H_0: *La palabra w sigue la misma distribución en ambas clases.*
H_1: *La palabra w sigue una distribución diferente en cada clase.*

En general, queríamos saber si una palabra tenía frecuencias distintas para cada clase, ya que eso podría ser un indicador de relevancia. Usando los valores de las proporciones de las frecuencias en cada clase (implementados anteriormente) creamos un sistema en el que las diferencias se calculan con valores en el intervalo [0,1].

```
# Ejemplo de funcionamiento de la ponderación
casa1 = 6/10 = 0.6
casa2 = 4/10 = 0.4
coche1 = 52/100 = 0.52
coche2 = 48/100 = 0.48
```

Dado que la diferencia entre `casa1` y `casa2` es > 0.05, ambas palabras siguen distribuciones distintas, es decir que hay diferencias significativas entre la clase estereotipo y no estereotipo. Por su parte, la diferencia entre `coche1` y `coche2` es ≤ 0.05, ambas palabras siguen distribuciones iguales, es decir que no hay diferencias significativas entre la clase estereotipo y no estereotipo.

Con respecto a la estructura, asignamos un peso a cada palabra. Así, aquellas palabras que no apareciesen etiquetadas como estereotipos y aquellas que aún estándolo, siguiesen una distribución semejante en ambas clases, tendrán un peso de 1. Por su parte, las que tenían una distribución diferente para cada clase, se guardaron con un peso de 1 más la diferencia asociada a esa palabra con respecto al estereotipo. Veamos un ejemplo:

Si la palabra *"islam"* tiene una distribución distinta en las clases Estereotipo y no Estereotipo, entonces: $p_w = 1 + |v[2] - v[3]|$.

Si la palabra *"español"* sigue la misma distribución ambas clases o si no aparece marcada como estereotipo, entonces: $p_w = 1.$

2.2 Extracción de características y representación

Para la extracción de características usamos 3 métodos:

- **N-gramas de palabras**: usamos *3-gramas*, dado que se trataba de una cantidad aceptable de datos y a nivel computacional ya suponía un coste bastante elevado.
- **Word Embedding Mediana**: usamos *word2vec* para representar cada oración como un vector mediana de los Word Embdding de las palabras que lo formaban. Usamos la mediana porque es una medida mucho más robusta que la media.
- **Word Embedding Ponderado**: usamos *word2vec* para realizar la representación de la ponderación llevada a cabo anteriormente. Representamos cada oración como un vector ponderado de los Word Embdding de las palabras que lo formaban.

3. Modelos entrenados y combinación de modelos

3.1 Resampling para balanceo

En la primera exploración de los datos, se observó que las clases estaban totalmente desbalanceadas (más observaciones de la clase *'no estereotipo'* que de la clase *'estereotipo'*).
Por este motivo, probamos dos opciones con el fin de solucionar este problema:

- Penalizar la clase mayoritaria, cambiando el parámetro *class_weight* al valor *balanced*.
- Uso de la técnica de resampling *SMOTeTomek*. Este es un método que combina *undersampling* y *oversampling*, generando en un primer paso muestras sintéticas (fase de *oversampling*) y eliminando las muestras creadas más ruidosas en una segunda instancia (fase de *undersampling*).

La segunda técnica nos ofreció mejores resultados, por lo que se seleccionó para balancear la muestra, dando como resultado una mejora sustancial en las métricas del modelo.

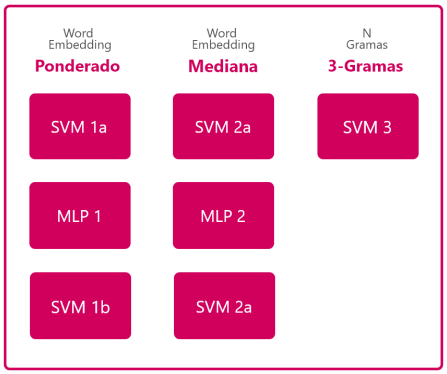
3.2 Elección de Modelos

Para elegir los mejores modelos nos basamos en 2 aspectos fundamentales:

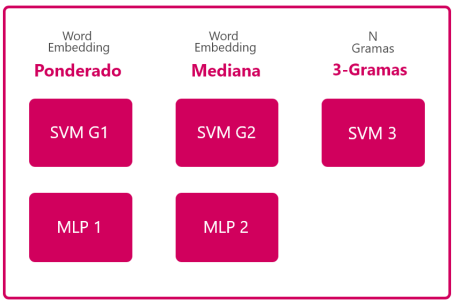
- El valor del *accuracy* en la clase target (stereotype). No buscamos un accuracy alto del modelo en general, sino que priorizamos que el accuracy sea alto en la detección de estereotipos. Esta métrica no es útil cuando los datos están desbalanceados. No obstante, como realizamos un balanceo con *SMOTeTomek*, la tuvimos en cuenta.
- Un *f1-score* alto, que hace patente la existencia de un buen ratio entre *precision* y el *recall* (cantidad de estereotipos que somos capaces de identificar) y no se ve afectado por el desbalanceo de los datos.

Por otro lado y con el fin de evaluar la calidad y consistencia del modelo, utilizamos una *Curva ROC* y la métrica *AUC* (área bajo la curva). El proceso de elección de modelos lo realizamos en diferentes fases:

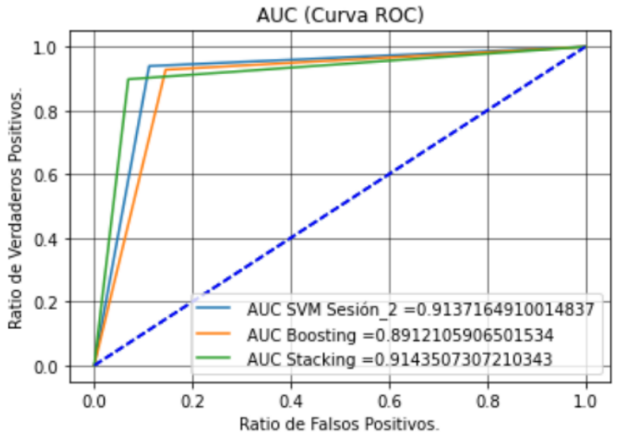
1. Construcción "manual" de los modelos, eligiendo aquellos que más consideramos que podrían proporcionar un mejor resultado:



2. Automatización de la optimización de hiperparámetros con *'GridSearch'* y *'RandomSearch'*. Después de esta fase, redujimos los modelos seleccionados a 5, experimentando una mejora significativa en todos estos modelos:



3. Aplicamos métodos de combinación de clasificadores (*bagging, boosting y stacking*), con el fin de incrementar la calidad de los modelos:



Finalmente, lo modelos seleccionados fueron los siguientes:

***#1** - STACK 1*

	precision	recall	f1-score
0.0	0.90	0.93	0.92
1.0	0.93	0.90	0.91

***#2** - SVM G1*

	precision	recall	f1-score
0.0	0.94	0.89	0.91
1.0	0.89	0.94	0.91

***#3** - BOOST 1*

	precision	recall	f1-score
0.0	0.92	0.85	0.89
1.0	0.86	0.93	0.89

Cabe mencionar que intentamos utilizar el algoritmo BETO para la construcción de nuevos modelos sin demasiado éxito. Entre los problemas que encontramos se destacan:

- La complejidad del sistema de transformers BETO.
- La dificultad para encontrar un modelo con métricas de calidad teniendo en cuenta nuestros limitados conocimientos sobre BETO.
- Ningún modelo entrenado con BETO proporcionó mejores resultados que los modelos creados anteriormente.

3.3 Problemas con los datos de Test

A la hora de usar los datos test, nos encontramos con un problema: nuestro sistema de ponderación usaba la etiqueta de clase para calcular los pesos, por lo que era imposible usar este método con los datos test. Así, y dado que los modelos creados mediante la representación con *Word Embedding Mediana* arrojaban resultados bastante buenos, decidimos realizar la representación de los datos test, principalmente, con este método.

De este modo, y dado que teníamos que presentar 3 ejecuciones para la tarea, decidimos realizar las siguientes adaptaciones:

- *1º Run*: Modelo *BOOST 2*, usando la representación con *Word Embedding Mediana*.
- *2º Run*: Modelo *XGBOOST*, usando la representación con *Word Embedding Mediana*.
- *3º Run*: Modelo *STACK 1*, usando la representación con *Word Embedding Ponderado*.

4. Estrategia de validación

Como estrategia de validación utilizamos el método de validación cruzada con *k-fold cross-validation*, más específicamente con $k = 10$.Elegimos este método por la calidad de los resultados y por su coste computacional, que no es excesivamente elevado.

En todos los modelos validados y finalmente seleccionados, los valores obtenidos en *accuracy y F1-score* para los distintos folds resultaron ser muy parecidos entre ellos. Esto quiere decir que los modelos eran independientes de la muestra y que no estábamos sufriendo problemas de sobreajuste.

Es importante destacar que al realizar *k-fold*, además de ver si el modelo sobreajusta o si es independiente de la muestra, también te permite estimar el parámetro real, es decir, que podíamos estimar (de manera aproximada) cuáles eran los posibles valores reales de estos parámetros.

5. Resultados

Finalmente, y teniendo en cuenta los problemas con la ponderación de los datos de test, los modelos que presentamos a la tarea DETESTS fueron los siguientes:

#1 - Modelo con *AdaBoost y una representación con Word Embedding Mediana* (*BOOST 2*):

```
      precision   recall   f1-score
0.0        0.87     0.89       0.88
1.0        0.89     0.87       0.88
```

#2 - Modelo con *XGBoost y una representación con Word Embedding Mediana* (*XGBOOST*):

```
      precision   recall   f1-score
0.0        0.88     0.89       0.89
1.0        0.89     0.88       0.89
```

#3 - Modelo con *Stacking y una representación con Word Embedding Ponderada* (*STACK G1*):

```
      precision   recall   f1-score
0.0        0.90     0.93       0.92
1.0        0.93     0.90       0.91
```

6. Análisis de errores

El objetivo en este punto es detectar posibles causas por las que el algoritmo erra ciertas predicciones. Para ello hemos analizado los resultados del modelo que mejores métricas obtenía y que fue usado para el envío de los resultados finales (el Modelo construido con *AdaBoost*). Mediante la observación de la *Matriz de Confusión*, pudimos hacernos a la idea de si era posible realizar una mejora sustancial del modelo o si la corrección de estos errores no serviría para mejorar en gran medida el modelo.

		Real	
		1	**0**
Predicho	**1**	762	114
	0	89	801

El tipo de error que nos interesa analizar son los falsos negativos. Estos errores serán *casos en los que el algoritmo ha predicho que no existen estereotipos cuando en realidad sí existen*. Los falsos negativos conforman el *43% de los errores totales*. Pese a esto, podemos observar que solamente conforman un *0.5% de los casos totales* y un *10% de los estereotipos*.

Por otro lado, tras analizar los 89 casos mal clasificados en busca de algún patrón solucionable en los errores, no encontramos ninguno. Algunos de los errores eran frases como: *'600.000 ilegales'*, *'Pues, muy bien. Alguien tendrá que recoger la fruta o dar yeso en las obras'* o *'Son incapaces de gestionar el país para los españoles y quieren nacionalizar a los que no lo son'*.

Esto nos hizo pensar que estos errores simplemente se debían a la limitación del algoritmo y que era una cantidad de error asumible por lo que no veíamos sentido a seguir invirtiendo esfuerzos en intentar perfeccionar todavía más los resultados pensando que se había llegado al límite de precisión.

7. Conclusiones sobre la tarea de DETESTS

Finalmente, del trabajo realizado, extraemos las siguientes conclusiones:

- Podríamos haber tenido en cuenta que los datos de test no están etiquetados a la hora de crear el sistema de ponderación.
- Ser conocedores de otros métodos de embedding como FastText para la inclusión de nuevas palabras o el entrenamiento de modelos más "personalizados" de representación.
- Tener más tiempo para la elaboración de las tareas y el entrenamiento de los modelos. El deadline ha limitado bastante el desarrollo de algunos apartados. Pensamos que sería positivo invertir más tiempo en la representación de los textos.
- La tarea ha sido muy motivadora. La participación en un concurso sale un poco de los trabajos típicos y ayuda a despertar motivación por el trabajo.