Detección de lenguaje ofensivo

09/05/2022

Jaime Rubio López

Procesamiento del Lenguaje Natural

Introducción

Para la resolución de este proyecto, he decidido realizar un clasificador híbrido, combinando un aprendizaje supervisado mediante el algoritmo de clasificación de Máquinas de Vectores de Soportes (SVM), con la detección de ofensividad mediante un lexicón. Entrenando al modelo mediante el SVM y haciendo luego una predicción completamente con el aprendizaje supervisado, y posteriormente la salida que me produce la predicción, comprobar para los valores de las frases que han salido como no ofensivas, comprobar esas frases mediante el lexicón para hacer un estudio de si esa predicción debe cambiar de no ofensivo a ofensivo.

Implementación y decisiones

Para la implementación de mi sistema, primero sigo unos pasos para la carga de los primeros archivos:

1. Primero mediante la librería Pandas cargo el archivo de entrenamiento y el del lexicón, además implemento una función para que todo el texto que le pase, sea preprocesado eliminando las palabras vacías, etc.
2. Una vez leído los archivos, recorro las tuplas de cada uno de ellos, en el caso del conjunto de entrenamiento, al recorrerlo, voy guardando en un array los comentarios que se van a usar para entrenar (siendo preprocesado antes por la función que he implementado), y en otro array distinto las etiquetas de ofensividad de su respectivo comentario, y en un tercer array las etiquetas del género.
3. Después guardo el contenido de los comentarios y de las etiquetas en un archivo de texto plano como paso opcional.

Tras estos pasos, empiezo con la implementación del algoritmo de clasificación, en este caso he usado SVM (Máquinas de Vectores de Soportes) para entrenar el modelo que voy a crear para las predicciones.

Para empezar, extraigo las características usando TF-IDF para generar una matriz frecuencia de término - frecuencia inversa de documento, y uso el método fit_transform() con el que obtengo el vector de características de los comentarios del conjunto de entrenamiento

He decidido combinar las predicciones con más de un tipo de etiquetado, en este caso, uso dos tipos de etiquetas, la de ofensividad y la de género, para ello no puedo usar un modelo de predicción linear, así que combino las distintas listas de etiquetas de entrenamiento en una matriz. Creo un objeto de la clase LinearSVC ya que soporta multiclass

Ahora convierto mi modelo a entrenar en un clasificador multi etiqueta pasándole el objeto LinearSVC, y entreno dicho modelo con la función fit() igual que en prácticas anteriores pero esta vez las etiquetas son una combinación de dos tipos de etiquetas.

He comprobado al hacer el entrenamiento con solo un tipo de etiqueta, usando las funciones vistas en prácticas anteriores, que me tarda la ejecución del entrenamiento del modelo alrededor de 2 minutos, mientras que usando el clasificador multietiqueta me hace una predicción igual de buena con la ofensividad, y el tiempo de ejecución tarda la mitad.

Una vez terminado el entrenamiento, he decidido hacer el modelo híbrido de aprendizaje supervisado y uso del lexicón haciendo primero la predicción del conjunto de prueba mediante el supervisado, y una vez que tengo los resultados, puede que algunas frases ofensivas me aparezcan como no ofensivas en el resultado de la predicción asique aquí es donde hago uso del lexicón, cojo los resultados de la predicción que hayan dado no ofensivos, y compruebo cuantas palabras del lexicón hay en cada uno de estos comentarios, y si supera un umbral establecido por el usuario (lo dejo como parametro en el codigo para que sea facil cambiarlo por el usuario) cambia la predicción de no ofensivo a ofensivo.

En este caso, dejo dicho umbral en valor 3, y me aparece la siguiente salida de la cantidad de comentarios que han sido cambiada su predicción de no ofensivo a ofensivo:

```
#AQUI COMIENZA LA PREDICCION
Y = vectorizer.transform(textosProcesados)
y_test_pred = multilabel_classifier.predict(Y)

itPredicc=0
for text in textos:
  if "NON" in str(y_test_pred[itPredicc]):
    var = procesarLexicon(itPredicc)  #LE PASO EL INDICE PARA SABER QUE FRASE ES LA QUE ME HA SALIDO COMO NO OFENSIVA
    if var: #SI SE CUMPLE, ESQUE SE HA CAMBIADO LA ETIQUETA DE NON A OFF AL COMPROBAR CUANTAS PALABRAS OFENSIVAS CONTIENE SEGUN EL LEXICON
      vecesCambiadoPredic += 1
      tagsGuardar.append("OFF")
    else:
      tagsGuardar.append(str(y_test_pred[itPredicc][0]))
  else:
    tagsGuardar.append(str(y_test_pred[itPredicc][0]))
  itPredicc+=1


print("SE HA CAMBIADO LA PREDICCION DE ", vecesCambiadoPredic, " COMENTARIOS")
```

```
SE HA CAMBIADO LA PREDICCION DE  23  COMENTARIOS
```

Al comprobar dichos comentarios con muestras por pantalla, he visto que efectivamente estos comentarios eran ofensivos y habían sido catalogados en un principio como no ofensivos. Si dejase el umbral con valor 1, me saldrían muchisimos mas comentarios entonces no serviria de nada tener un umbral bajo.

Sigo los siguientes pasos para hacer las predicciones sobre el conjunto de entrenamiento del corpus:

1. Declaro una función procesarLexicon() que me recorre todas las palabras del lexicón y comprueba con cada una si está en el comentario que le pasamos, si es así, aumenta un contador que en el caso de pasar el umbral, la función devuelve true, en otro caso devuelve false.
2. Declaro dos arrays de comentarios, uno que tendrá los comentarios sin procesar para mostrarlos por pantalla (también se mostrará debajo de cada comentario, las predicciones de los dos tipos de etiquetas), y otro con los comentarios procesados para hacer las predicciones. Leo el conjunto de prueba rellenando dichas listas.
3. Hago la predicción con multilabel_classifier.predict() lo que me devuelve una matriz donde me devuelve la predicción de las etiquetas de ofensividad y de género. Después me recorro el array de comentarios buscando si es no ofensivo, para que en ese caso compruebe con la función antes declarada, si usando el lexicon, cambiará su predicción.
4. Finalmente guardo los resultados en un archivo y compruebo los resultados con el gold_label_test del corpus.

Ejecución de la aplicación

Durante el desarrollo de este proyecto, la ejecución me tardaba bastantes minutos en terminar sobre todo en el aprendizaje del modelo y en las predicciones, finalmente tras adoptar el sistema de clasificación multi etiqueta me ha descendido el tiempo de ejecución del programa a la mitad siendo así más eficiente a la vez que me mantiene la misma eficacia que cuando usaba el modelo lineal.

Al ejecutar mi código, muestro por pantalla todos los comentarios y sus predicciones, además de un mensaje de cuantos de estos comentarios han sido cambiadas sus predicciones.

Tras comprobar mis resultados con el conjunto de evaluación dado en el corpus, obtengo unos porcentajes de aciertos elevados, la mayoría cerca del 1:

```
{'maf': 0.7565527033938739, 'map': 0.8351596369523862, 'mar': 0.7192060438337871,
```

```
'mif': 0.8759370865794502, 'mip': 0.8759370865794502, 'mir': 0.8759370865794502,
```

```
'avgf': 0.8631065231472472, 'avgp': 0.8680227369929996, 'avgr': 0.8759370865794502}
```

Parámetros de la aplicación

Los parámetros que uso a lo largo de todo el código de mi aplicación son una lista para guardar todas las palabras contenidas en el lexicón, tres listas para el conjunto de entrenamiento del corpus (uno para guardar los comentarios, otro para guardar las etiquetas de ofensividad, y el último para guardar las etiquetas del género), y tres listas más para guardar los comentarios sin procesar, en otro los comentarios procesados de mi conjunto de test del corpus, y en el último el id de dichos comentarios.

Además tengo un parámetro a cambiar a gusto del usuario, llamado umbralLexicon para comprobar distintos resultados según el resultado de las predicciones con dicho umbral (inicialmente está a 3).

También se puede cambiar al principio y final del código a gusto la ruta donde está el directorio de archivos.

Resultados obtenidos

Al comprobar los resultados haciendo la ejecución sin contar el lexicón, y luego tras hacer el sistema híbrido con el lexicon, he notado una mejoría en el resultado de las predicciones al comprobarlo con el conjunto de evaluacion del corpus, dandome los siguientes valores:

1. **MAF**: 0.75
2. **MAP**: 0.83
3. **MAR**: 0.71
4. **MIF**: 0.87
5. **MIP**: 0.87
6. **MIR**: 0.87
7. **AVGF**: 0.86
8. **AVGP**: 0.86
9. **AVGR**: 0.87

Estos valores varían según el umbral que seleccione en el lexicón, pero poniendo un umbral entre 5 y 7 me da unos resultados cercanos al 1, por lo que en general obtengo un porcentaje de acierto bueno.

Conclusión y valoración personal

En general, mi algoritmo de clasificación híbrido proporciona buenos resultados. Buscando información en otros trabajos de otras universidades por internet, he encontrado sistemas en los que se usaba el lexicon además con palabras positivas no solo con palabras negativas, lo que daría una fluidez mucho mayor a la hora de seleccionar pesos a cada palabra de un comentario, por lo que se podría hacer predicciones más exactas que teniendo en cuenta solamente las palabras negativas, ya que a veces puede haber ironías no detectables por el ordenador, de esta manera se podría aclarar mejor un umbral para decidir si un comentario es ofensivo o no. En este caso no tenemos claro cuál podría ser el umbral así que haciendo pruebas de ejecución he encontrado que da mejores resultados con un umbral de entre 3 y 5 palabras ofensivas en un comentario para clasificarlo como ofensivo según el lexicon. En conclusión es una manera eficiente de calcular una buena predicción aunque se podría mejorar mucho más teniendo en cuenta palabras positivas e incluso neutras en el lexicón.