UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MIARFID
Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital
Universitat Politècnica de València

Sistema de Sentiment Analysis

Aplicaciones de la Lingüística Computacional

Autor: Juan Antonio López Ramírez

Curso 2019-2020

Introducción

Este trabajo consiste en la realización de un sistema de para el análisis y reconocimiento de sentimientos.

Para ello, se ha utilizado el conjunto de datos de la tarea TASS20171 (Análisis de sentimientos en Twitter para el español), que está formado por un conjunto de entrenamiento de 1008 tweets, un conjunto de *development* de 506 tweets y un conjunto de test de 1899 tweets. Cada tweet tiene un identificador; una etiqueta de sentimiento, que puede ser positivo (P), negativo (N), neutro (NEU) o ninguno (NONE); y el texto que contiene.

Por lo tanto, el propósito es desarrollar un clasificador que etiquete los tweets, basándose en su contenido y haciendo uso de las etiquetas que contienen, tratando de minimizar el error de clasificación.

Preproceso, limpieza y vectorización de los datos

Los conjuntos de datos son ficheros XML que contienen las marcas indicadoras de cada campo (identificador, contenido y sentimiento) y otra información irrelevante.

Para deshacerse de esta última, hemos implementado un script en Python que hace la función de un parser, que extrae la información de peso del XML y la almacena en un fichero de texto. De esta forma, obtenemos tres ficheros de texto, uno para training, otro para development y otro para test.

En cada uno de estos, se tienen tantas filas como tweets tenga dicho conjunto de datos, y cada fila consta del id, el sentimiento del tweet y el texto, separados los tres por un espacio.

Una vez hecho esto, creamos otro script en Python llamado *sas.py* que implementará la mayor parte de nuestro sistema de reconocimiento.

En este, hemos realizado un proceso de limpieza sobre los textos extraídos de la red social. Esto se debe a que muchos elementos, como los usuarios, los hashtags o las direcciones web, tienen un significado muy similar, a pesar de que pueden ser muy variados. Por eso hemos implementado expresiones regulares que detecten estos tipos de contenido, reemplazándolos por un token que indique la categoría de cada uno.

Finalmente, hemos vectorizado el texto para poder entrenar un modelo. En nuestro caso, hemos usado la herramienta *TfidfVectorizer*, de la librería *Scikit-Learn*. Esta herramienta nos permite convertir un conjunto de texto plano a una matriz de TF-IDF. Con esta matriz podemos comprobar la relevancia de cada palabra en un documento determinado (en este caso, el dataset), y así obtener una matriz de características significativas (*features*), que pueden ser clave cuando clasifiquemos un texto según el sentimiento que transmite.

Modelos utilizados y resultados

Cuando ya tenemos el texto preprocesado, hemos probado distintos modelos de clasificación para encontrar el que mejor rendimiento nos proporcione para nuestra tarea. Hemos tenido en cuenta, como puntuación mas relevante, el *F1 score* que se obtiene al contar el número de palabras con sentimiento positivo y negativo que aparecen en un determinado lexicon proporcionado.

Todos los modelos utilizados han sido proporcionados, una vez más, por la librería *Scikit-Learn* y consisten en algoritmos de Machine Learning, como pueden ser el de vecinos más cercanos, vectores soporte, descenso por gradiente estocástico, etc.

Teniendo en cuenta los promedios macro, los resultados han sido los siguientes:

Modelo	Accuracy	Precisión	Recall	F1
Vectores soporte	0.55	0.38	0.36	0.32
Vectores soporte (versión lineal)	0.54	0.39	0.39	**0.38**
Naive Bayes Gaussiano	0.42	0.34	0.33	0.33
Gradient Boosting	0.53	0.42	0.37	0.36
Descenso por gradiente estocástico	0.49	0.36	0.36	0.36
Vecinos más cercanos	0.44	0.37	0.36	0.34

Como podemos apreciar, sin realizar ningún ajuste de sus parámetros (exceptuando quizá el hecho de que a los vectores soporte y vectores soporte lineal se le ha aplicado un C=1), las *scores* de F1 obtenidas han sido muy similares. Como el mejor modelo, teniendo en cuenta esta métrica, ha sido el clasificador de vectores soporte lineal, hemos hecho el ajuste de los hiperparámetros a partir de él.

Para ello, vamos modificando dichos parámetros del modelo, que son:

- El parámetro C, para la penalización de los errores. Hemos probado distintos valores (1000, 100, 1 y 0.01).
- El umbral de tolerancia para determinar la convergencia del algoritmo. Los valores que se han probado son 0.1, 0.01 y 0.001.
- La funcion de coste o *loss*. Por defecto, hemos usado siempre la función *hinge*, debido a que la variación de esta no supone una repercusión significativa en la métrica F1.
- El tipo de normalización aplicada a la penalización.
- El número máximo de iteraciones antes de parar el entrenamiento.

Dado que los hiperparámetros más importantes de las máquinas de vectores soporte son la tolerancia y C, se ha decidido variar estos dos, dejando como función de penalización la función *hinge* y l2 como la normalización de la penalización. Además, para asegurarnos de no finalizar el algoritmo antes de su convergencia, establecemos un valor grande para el número máximo de iteraciones (10^9).

Los valores de C probados han sido 1000, 100, 1 y 0.01; mientras que los de la tolerancia han sido 0.1, 0.01 y 0.001.

Los resultados obtenidos, ajustando dichos parámetros, se pueden observar en la tabla 1.

Por tanto, el modelo de clasificador basado en máquinas de vectores soporte lineal con el que se ha realizado la estimación de etiquetas de sentimiento sobre el conjunto de test se ha configurado con los parámetros:

- C = 100
- tol = 0.1
- loss = 'hinge'

- penalty = 'l2'
- max_iter = 1000000000

Con él, se ha conseguido una puntuación F1 de **0.3999**, siendo bastante superior a el resto de experimentos que hemos hecho.

C	**tolerancia**	**F1**
1000	0.1	0.3966
1000	0.01	0.3982
1000	0.001	0.3982
100	0.1	**0.3999**
100	0.01	0.3982
100	0.001	0.3982
1	0.1	0.3716
1	0.01	0.3716
1	0.001	0.3716
0.1	0.1	0.35
0.1	0.01	0.3528
0.1	0.001	0.3525

Tabla 1: Resultados de ajustar los hiperparámetros.