Computational Linguistics: Development of a Sentiment Analysis System

Ramon Ruiz Dolz

April 2019

1 Introducción

En esta memoria se presenta el proceso mediante el cual se ha realizado el diseño y la implementación de un sistema para el reconocimiento de sentimientos. Para ello se ha dispuesto del *dataset* empleado en el TASS2017[1] compuesto por un conjunto de entrenamiento formado por 1008 *tweets*, un conjunto de *development* formado por 506 *tweets* y un conjunto de test formado por 1899 *tweets*. Cada *tweet* consta de 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, nuestro objetivo consiste en desarrollar un clasificador capaz de etiquetar *tweets* en función de su texto, haciendo uso de las etiquetas anteriores tratando de minimizar el error de clasificación.

Este documento se ha estructurado de la siguiente forma: en la section 2 se explica la obtención del texto a partir del *dataset* proporcionado, así como el preproceso aplicado a estos textos con el objetivo de tratar con el mínimo de información irrelevante posible; en la section 3 se presentan las distintas pruebas realizadas con distintos modelos de clasificación y se escoge un modelo de entre todas las pruebas; en section 4 se muestra la exploración realizada entorno al ajuste de los parámetros del modelo seleccionado, con el objetivo de incrementar todavía más la precisión obtenida; finalmente, en **??** se realiza una comparativa de los resultados obtenidos por el sistema implementado en este trabajo, y en section 5 se extraen las principales conclusiones alcanzadas durante la realización de este proyecto.

[1] `http://www.sepln.org/workshops/tass/2017/`

2 Preproceso

Los datos de partida provienen de ficheros XML extraidos directamente de la web. Estos ficheros contienen las marcas indicadoras de cada campo y otra información que no es útil a la hora de analizar el contenido de los *tweets*. Por lo tanto, todos estos datos existentes en el fichero XML podrian considerarse ruido y deben ser eliminados. Para ello se dispone de un *parser* implementado en Python que extraerá únicamente aquella información necesaria para la realización de esta tarea. En este caso, el identificador del *tweet*, el texto que contiene, y para los conjuntos de entrenamiento y desarrollo, las etiquetas del sentimiento asociado a cada *tweet*.

Una vez separada la información relevante también se ha considerado realizar un proceso de *limpieza* sobre los textos extraídos de la red social. Muchos elementos como los usuarios, los *hashtags*, las direcciones web, etc. Pueden ser muy variadas pero, sin embargo tener un significado similar en el texto. Es por ello que mediante expresiones regulares se han unificado todos estos tipos de contenido. Por lo tanto, se ha trabajado con un texto donde los usuarios, los *hashtags* y las direcciones web han sido reemplazados por un único token para cada una de sus categorías.

Finalmente, se ha procedido a la vectorización del texto con tal de poder entrenar un modelo adecuadamente. En este caso se ha hecho uso de la herramienta de sklearn TfidfVectorizer[2]. Esta herramienta nos permite convertir un conjunto de texto plano a una matriz de TF-IDF. Esta matriz nos indica cuan relevante es cada palabra en un documento determinado (el corpus en este caso), de esta forma se obtiene una matriz de características significativas que pueden ser determinantes a la hora de clasificar un texto según el sentimiento que transmite.

3 Selección del modelo

Una vez con el texto preprocesado y en un formato adecuado para entrenar un modelo se han probado distintos modelos de clasificación con tal de encontrar el que mejor rendimiento nos proporcionase. Se ha tomado como *baseline* la puntuación f1[3] obtenida al contar numero de palabras con sentimiento positivo y negativo que apareciesen en el lexicon proporcionado.

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html`

A partir de esta puntuación se han tenido en cuenta todos los modelos que fuesen como mínimo capaces de mejorar la puntuación alcanzada.

Se ha probado a lanzar experimentos con el clasificador Naïve Bayes, con el clasificador Gradient Boosting, también se ha probado con el clasificador por vectores de soporte (Support Vector) y su versión lineal (Linear Support Vector). Además también se ha probado con técnicas no supervisadas como vecinos más cercanos (Nearest Neighbors) y una aproximación más clásica como es el descenso por gradiente estocástico. A continuación se puede observar los resultados obtenidos tras esta primera fase de experimentación.

Model	**F1 Score (macro)**
Baseline	0.359
Naive-Bayes	0.336
Gradient Boosting	0.372
Suport Vector	0.151
Linear SVC	**0.385**
Nearest Neighbors	0.363
Stochastic Grad. Desc.	-*

Table 1: Score obtenida por cada modelo en la primera fase.

Como podemos observar a excepción del modelo Naïve Bayes y el SVC no lineal, todos los demás han conseguido superar la baseline propuesta como punto de partida. También veo relevante destacar la variabilidad mostrada por el modelo de descenso por gradiente. Sin realizar un ajuste de sus hiperparámetros, en varias ejecuciones ha mostrado comportamientos muy variados, desde superar la baseline con holgura hasta quedar sustancialmente por debajo. Sin embargo, lo que si que ha quedado claro es que el modelo con mejor comportamiento ha sido el SVC lineal. Por lo tanto se realizará el ajuste de hiperparámetros a partir de este modelo.

4 Ajuste de hiperparámetros

Ya con el modelo escogido para clasificar, en esta sección nos centraremos en encontrar los parámetros del modelo que mejor rendimiento obtengan entrenando con el conjunto de entrenamiento y evaluando sobre el conjunto de desarrollo. El modelo Linear SVC[4] viene definido por una serie de parámetros. En este apartado se ha realizado el ajuste de los siguientes parámetros relacionados con el modelo escogido:

[4]https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

- **tol:** umbral de tolerancia para determinar la convergencia del algoritmo.
- **C:** parametro de penalización de los errores.
- **loss:** especifica la función de coste.
- **penalty:** define el tipo de normalización aplicada en la penalización.
- **max_iter:** especifica el número máximo de iteraciones antes de detener el entrenamiento.

Las combinaciones que mejor resultado han obtenido se muestran en la Table 2. Se han probado distintos valores de tolerancia (0.1, 0.01, 0.001, ...), como se puede observar, cuando menor el valor de tolerancia, el resultado obtenido mejora hasta cierto punto pero complica la convergencia por lo que puede tomar más tiempo obtener nuestro resultado. De hecho, a partir de tol = 0.01 ya no se aprecia ninguna mejora significativa en la medida F1. Para el parámetro C, se han probado desde valores muy elevados (1000) hasta valores muy pequeños (0.01). Por la naturaleza de los datos, con C = 1000 y C = 100 se ha conseguido obtener el mejor resultado sobre desarrollo. Puesto que es interesante que el modelo entrenado sea robusto a otros conjuntos diferentes a desarrollo se ha preferido C = 100 frente a C = 1000. En cuanto a los parámetros tanto loss como penalty no se ha observado una repercusión significativa sobre la métrica F1. Finalmente, para no detener el algoritmo antes de su convergencia, se ha configurado un numero de iteraciones máximas muy elevado.

C	tol	loss	penalty	max_iter	F1 Score
1000	0.1	hinge	l2	∞	0.392
100	0.1	sq_hinge	l2	∞	0.394
1	0.1	hinge	l2	∞	0.377
0.1	0.1	sq_hinge	l2	∞	0.348
0.01	0.1	hinge	l2	∞	0.297
1000	0.01	sq_hinge	l2	∞	0.396
100	**0.01**	**hinge**	**l2**	∞	**0.396**
1	0.01	sq_hinge	l2	∞	0.385
0.1	0.01	hinge	l2	∞	0.354
0.01	0.01	sq_hinge	l2	∞	0.223

Table 2: Ajuste de hipperparámetros.

Por ello, el modelo Linear SVC 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 = 0.01, loss = 'hinge', penalty = 'l2', max_iter $\approx \infty$. Se ha conseguido alcanzar una puntuación F1 de 0.396, siendo esta significativamente superior a todos los demás experimentos realizados.

5 Conclusiones

Mediante la realización de este trabajo ha sido posible observar la importancia de cada una de las tres tareas descritas. El preproceso, la selección de un modelo adecuado y el ajuste de sus parámetros. De hecho en este trabajo se ha llegado a un punto en el cual ya no era posible seguir mejorando los resultados obtenidos. Sin embargo, enfocando el problema con un preprocesado y una extracción de características más elaborada es posible alcanzar resultados incluso mayores, tal y como se puede observar en [1]. Es por ello que familiarizarse con las distintas técnicas de extracción de características y conocer gran cantidad de modelos y sus propiedades relacionadas con la naturaleza de los datos es algo de gran importancia para la realización de este tipo de trabajos o competiciones.

References

[1] E Martınez-Cámara et al. "Overview of TASS 2017". In: *Proceedings of TASS* (2017), pp. 13–21.