

TFM - Análisis de Sentimiento de tuits en español

Curso 2018-2019

Paloma Ortiz Pérez de León



Indice

1. Contenido
2. Motivación
3. Los datos
4. Fases de trabajo
 - 4.1. Fase de extracción de tuits
 - 4.2. Fase de preprocesamiento de los datos
 - 4.3. Análisis exploratorio de los datos
 - 4.4. Fase de limpieza de los datos
 - 4.5. Fase de clasificación de tuits
 - 4.6. Vectorización de tuits
 - 4.7. Fase de entrenamiento de modelos de clasificación
 - 4.7.1. Conjuntos de entrenamiento y test
 - 4.7.2. Validación cruzada o cross validation
 - 4.7.3. Búsqueda de mejores parámetros con GridSearch
 - 4.7.4. Evaluación de resultados
 - 4.7.4.1. Support Vector Machine SVM
 - 4.7.4.2. Modelo de Regresión Logística
5. Conclusiones finales
6. Futuros trabajos

1. Contenido

Este trabajo es una aplicación de técnicas y algoritmos aprendidos durante el Máster de Data Science. En concreto, se centra en el campo del Procesamiento del Lenguaje Natural (PLN) que abarca parte de la Ciencia de datos, Inteligencia Artificial y Lingüística.

El reto de este estudio reside en hacerlo para textos escritos en español y lenguaje informal, ya que la mayoría de los trabajos de PLN están hechos en base a textos literarios o formales en los cuales la gramática, ortografía y expresiones son más correctas y esto es reflejado en los resultados obtenidos.

2. Motivación

El principal objetivo de este trabajo es realizar un análisis de sentimiento para predecir la polaridad de tuits en español procedentes de Twitter. Se trata por tanto de un caso de estudio de Aprendizaje automático supervisado para la clasificación en tres categorías o polaridades:

- Positiva
- Neutra
- Negativa

Para llegar a ese fin, ha sido necesario pasar por diferentes fases de trabajo:

- Fase de extracción de datos
- Fase de preprocesamiento de los datos
- Fase de limpieza de los datos
- Análisis exploratorio
- Fase de clasificación de polaridad de los tuits
- Fase de entrenamiento de modelos

Las herramientas y funciones utilizadas para cada fase han sido codificadas en Python 3 y se han usado diferentes librerías específicas de PLN como NLTK, Sklearn, etc.

3. Los datos

El fichero original **2_Input_PreprocessingTweets.csv** contiene una muestra de tuits escritos en español que fueron publicados antes, durante y después de las manifestaciones que tuvieron lugar el día 8 de marzo del 2019 con motivo del día Internacional de la Mujer. Todos los mensajes o tuits han sido recogidos bajo la etiqueta o *hashtag* **#8M**.

El fichero consta de 20 variables y un total de 33.010 registros, dichas variables son:

- Tweets: el texto del tuit.
- Tweet_long: la longitud del tuit.
- Id: identificador del tuit.
- Created_date: fecha de creación del tuit.
- Source: fuente origen de la que ha sido creado el tuit.
- Likes: número de "me gusta" que tiene el tuit.
- RTs: número de retuits realizados.
- Language: idioma en el que el tuit ha sido escrito.
- Place: localización desde donde el tuit ha sido publicado.
- User_id: identificador del usuario que ha publicado el tuit.
- User_name: nombre de usuario creador del tuit.
- User_description: descripción del perfil del usuario.
- Followers: número de seguidores del usuario que ha tuiteado.
- Followings: número total de usuarios a los que sigue el autor del tuit.
- User_lists_member: listas a las cuales pertenece el usuario.
- User_total_favourites_count: número de favoritos que posee la cuenta.
- User_statuses_count: número publicaciones.
- User_created_account: fecha de creación de la cuenta del usuario autor del tuit.

- User_location: localización origen del usuario autor del tuit.
- User_lang: idioma origen del usuario autor del tuit.

4. Fases de trabajo

4.1. Fase de extracción de tuits

Con el propósito de obtener un conjunto de datos propios, los tuits han sido extraídos utilizando Tweepy, la librería de Python para acceder a la API de Twitter. Para obtener las claves y tokens de acceso es necesario darse de alta en Twitter y crear una aplicación como desarrollador. Una vez realizado, se proporcionarán cuatro claves que deben incluirse en el notebook **O_Tweets_Extractor.ipynb** de [GitHub](#), el cual contiene el código para extraer los tuits. Las cuatro claves necesarias son las siguientes:

- consumer_key
- consumer_secret
- access_token
- access_token_secret

Dichas claves son privadas ya que con ellas es posible crear y borrar tuits procedentes de la cuenta de Twitter asociada, razón por la cual no están incluidas en el código. En el [link](#) de la aplicación se puede consultar más información acerca del proceso.

Tweepy proporciona varios métodos para obtener mensajes con posibilidad de añadir parámetros para extraer tuits con palabras claves, también permite especificar el idioma de los tuits, fijar la localización donde son publicados, excluir mensajes que han sido retuiteados, etc. En este trabajo se ha utilizado algunos de estos parámetros para filtrar tuits en español y con etiqueta específica *#8M*.

Además de incluir las claves, el notebook **O_Tweets_Extractor.ipynb** también necesita especificar la fecha (año, mes, día, hora, minuto y segundo) en que concluirá el proceso de extracción. Los datos de salida serán guardados en un fichero csv con los tuits recogidos, separados por el delimitador la barra vertical "|".

El proceso de extracción se llevó a cabo los días 8 y 9 de marzo de 2019 coincidiendo con el aumento de actividad en Twitter propiciado por las movilizaciones que tuvieron lugar en muchas localidades de España y Latino América. Al finalizar el proceso, se obtuvieron en total **33.301 registros y 20 variables** con información acerca de la publicación de los tuits.

4.2. Fase de preprocesamiento de los datos

El preprocesamiento de los datos se refiere a la fase de normalización y transformación del conjunto de datos. Para ello se realizaron diferentes tareas como:

- Normalización de los nombres de columnas.
- Normalización de los campos a formato fecha *dd-mm-yyyy hh:mm*.
- Eliminación de espacios en blanco a principio y fin de columnas.
- Eliminación de caracteres \n y \r de los registros.
- Eliminación de tuits incompletos, es decir, aquellos que tienen 139-140 caracteres.

Estos son producto de retuits que se dividen en dos mensajes de forma que el primero de ellos es el tuit original. Con este filtro, el conjunto de datos resultante se redujo a 11.924 registros, es decir, se descartaron un 64% del total.

- Eliminación de tuits repetidos, finalmente quedaron 5.301 tuits únicos que corresponden a un 16% del total de los tuits originales.

4.3. Análisis exploratorio de los datos

Con motivo de las manifestaciones del día Internacional de la mujer celebrado el 8 de marzo, se recogieron tuits que se utilizarán como corpus para hacer análisis de sentimiento. Por ello, es interesante conocer y explorar los datos con el fin de obtener toda la información posible. Algunos de los ejemplos más relevantes:





- El primer tuit recogido fue a las 2019-03-08 17:59:21, previo a la hora de inicio de la manifestación general convocada:

RT @FrasesRockNacOk: EL DÍA QUE NOS LAS MATEN... ese día seran FELICES. RESPETALA. CUIDALA. VALORALA. NO LAS ACOSES. #8M #NiUnaMenos



- El último tuit se recogió a las 2019-03-09 11:49:49:

Ser feminista y apellidarse Calvo Poyato. Y así todo. #8M

- El mensaje más retuiteado con un total de 9.467 retuits fue publicado a las 2019-03-08 17:59:23 en Córdoba, Argentina:

RT @FonziDolores: Mejor video en lo que va del día. #8M Lloré.     <https://t.co/Uzudpmhp6y>

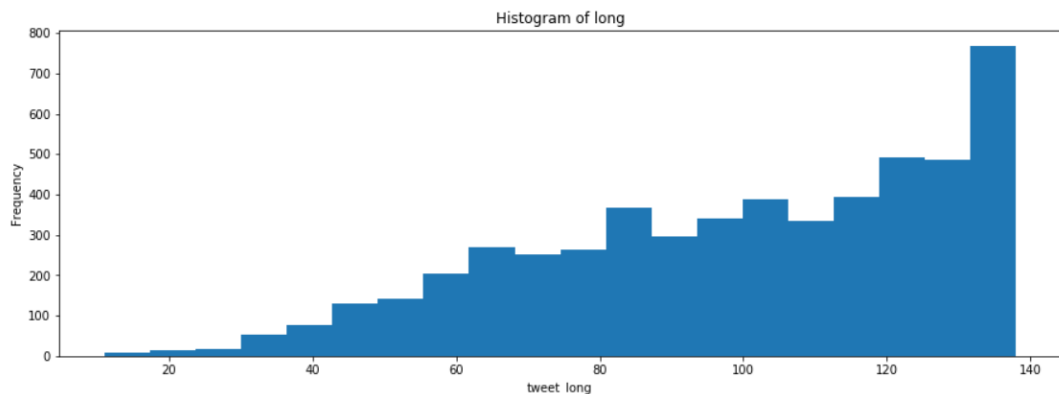
- El tuit más valorado con un total de 5 “me gustas” fue publicado el 2019-03-08 23:07:24 en América Latina:

¿Cuáles son tus mujeres favoritas en la música?   #DíaDeLaMujer #EllaMeInspira #8M

- Entre los tuits publicados, las fuentes más utilizadas fueron *Android*, *iPhone*, *Web* y la *App de Twitter*:

Number of Tweets	
source	
Twitter for Android	2918
Twitter for iPhone	1200
Twitter Web Client	524
Twitter Web App	201
Instagram	96
TweetDeck	78
Facebook	61
Twitter for iPad	48
Hootsuite Inc.	45
IFTTT	27

- Las longitudes de los tuits recogidos frecuentan entre 120 y 139 caracteres. Hay que tener en cuenta que con el filtro previo realizado se descartaron tuits con longitudes mayores a 139 caracteres por aparecer incompletos.



- Las localizaciones con más actividad en Twitter durante las movilizaciones fueron Madrid y Argentina.

Number of Tweets	
user_location	
Madrid	121
Argentina	98
España	79
Buenos Aires, Argentina	69
México	59
Chile	54
Madrid, Comunidad de Madrid	50
Santiago, Chile	44
Barcelona	34
Montevideo, Uruguay	28

Como se puede apreciar, el campo *user_location* no está normalizado ya que hay valores diferentes que se refieren al mismo lugar. Por ejemplo, para referirse a *Madrid* encontramos valores como *Madrid, Comunidad de Madrid* o simplemente *Madrid*.

- Para analizar los hashtags más utilizados, se ha realizado una limpieza previa de signos de puntuación, se han pasado todos los mensajes a minúsculas y se han eliminado tildes. Como era de esperar la etiqueta más común es *#8M*, ya que ha sido el elemento por el cual fueron recogidos los tuits:

Los anteriores resultados están recogidos junto al código en el notebook **3_Data_Exploration.ipynb** procedente de [GitHub](#).

Además, como parte del análisis exploratorio del conjunto de datos, se realizó un intento frustrado de clasificación de las palabras en diferentes grupos utilizando como herramienta la librería *SpaCy*, la cual permite diferenciar términos según se refieran a Personajes (PER), Localizaciones (LOC), Organizaciones (ORG) o Misceláneos (MISC). Para ello *SpaCy* cuenta con un diccionario de palabras en español *es_core_news_sm*. Finalmente, los resultados no se mostraron en este trabajo ya que no fueron demasiado precisos al no estar actualizado recientemente para español y por tanto no era posible reconocer algunos nombres de políticos relevantes de la actualidad, organizaciones conocidas, etc.

4.4. Fase de limpieza de los datos

El proceso de limpieza consistió en implementar funciones creadas específicamente para aplicarlas sobre la variable *Tweets* que contienen todos los mensajes o tuits. Los procesos realizados fueron los siguientes:

- Eliminar espacios a principio y final de tuits.
- Conversión a minúsculas.
- Eliminar signos de puntuación.
- Eliminar números.
- Eliminar menciones a usuarios.
- Eliminar hashtags.
- Eliminar links o URLs contenidos en los tuits.
- Eliminar la abreviatura "RT" al comienzo de los mensajes.
- Eliminar tildes.
- Normalizar expresiones de risa como "jajaja", "jejeje", "jijiji" o "hahaha", las cuales se sustituirán por "LOL".
- Eliminar tuits vacíos, resultado de toda la limpieza.

Todas estas transformaciones se llevaron a cabo con el fin de facilitar la fase posterior de clasificación y así tener un corpus lo más limpio posible con el que poder trabajar. No obstante, al tratarse de lenguaje informal hay casos de errores gramaticales que son difíciles de detectar y corregir, es por ellos que no han sido tratados en este trabajo.

Tras esta fase de depuración de los tuits, se obtuvieron **4.869 registros válidos que conforman la muestra final de datos**. El código correspondiente a este proceso de limpieza se encuentra en el notebook **2_Data_Cleansing.ipynb** de [GitHub](#).

4.5. Fase de clasificación de los tuits

Los tuits resultantes del proceso de depuración fueron clasificados manualmente según su polaridad. Han sido divididos en tres clases:

- *Positivos*: según fuesen mensajes de ánimo, alegría, apoyo, etc. Marcados con valor 1.
- *Neutros*: donde no se encontraron rasgos positivos o negativos. Marcados con valor 0.
- *Negativos*: se incluyeron mensajes ofensivos, de queja o lucha. Marcados con valor -1.

Para anotar la polaridad se creó una nueva columna *Polarity* en el conjunto de datos con sus posibles valores. Con el objetivo fundamental de incrementar la asertividad en la elección de la polaridad, se consideró la respuesta de tres individuos, donde uno de ellos evaluó el corpus total y en lo referente a los otros dos individuos, cada uno de ellos clasificó una mitad de la muestra. De esta forma cada tuit fue clasificado por más de un individuo, así, en los casos de discrepancia entre dos individuos la tercera opinión permitía la resolución del conflicto. Del corpus completo se evaluaron:

- 915 tuits negativos, 18.81% del total.
- 1.874 tuits neutros, 38.52% del total.
- 2.075 tuits positivos, 42.66% del total.

4.6. Vectorización de tuits

En los algoritmos de aprendizaje automático a menudo se toman vectores de valores numéricos como entrada, esto no ocurre cuando se trabaja con textos por lo que surge la necesidad de convertirlos en un vector numérico. Este proceso se conoce como vectorización y consiste en transformar textos en números con el fin de crear una *bolsa de palabras o términos (bag of words)*. Ésta indica cuando una palabra está en el conjunto de datos o no sin importar el orden en el que aparezca. Dicha técnica es básica en el campo del procesamiento de lenguaje natural (PLN), para ello se suele emplear *TfidfVectorizer* de *sklearn* que es una librería de uso frecuente en trabajos de clasificación. La ventaja de utilizar *TfidfVectorizer* frente a otras técnicas como *Word2Vec* es que la primera permite crear un vocabulario de términos calculando los pesos de todas las palabras que componen los mensajes, de esta forma los pesos serán mayores en palabras que aparecen pocas veces pero aportan mucho contenido, mientras que penalizan las que aparecen demasiadas veces en el conjunto de datos.

TfidfVectorizer permite usar parámetros más específicos, los utilizados para realizar el análisis de sentimiento de este trabajo son los siguientes:

- `ngram_range`: tuple (`min_n`, `max_n`).

Indica el límite inferior y superior del rango de valores de `n` para distintos `n`-grams. Permite agrupar los términos en el rango definido $min_n \leq n \leq max_n$.

En este trabajo se utilizarán `ngram_range` igual a (1,2) y (1,3) con el fin de comparar resultados, con ello se tendrán en cuenta tanto términos sueltos como agrupaciones de dos o tres términos.

- `stop_words`: string {'english'}, lista. Por defecto es None.

Este parámetro permite ignorar las llamadas *stopwords* o *palabras vacías* ya que por sí solas carecen de significado y no aportan valor al conjunto de términos. Pueden ser artículos, preposiciones, pronombres, etc. Para este trabajo se ha utilizado la lista de *stopwords* en español de la librería *NLTK*.

- `use_idf`: Boolean. Por defecto es True.

Permite habilitar la ponderación inversa de frecuencias de las palabras que componen el corpus.

- `min_df`: float in range [0.0, 1.0] or int. Por defecto es 1.

Este parámetro ignora los términos que tengan una frecuencia estrictamente menor a la indicada, de esta forma los elimina del corpus final para que no interfieran en las predicciones. En este trabajo se ha fijado un valor `min_df` igual

a 3, como la frecuencia mínima para pertenecer al corpus final, el resto que sean menores serán descartados del entrenamiento.

- `sublinear_tf`: Boolean. Por defecto es False.

Aplica la escala *tf* sublineal, es decir, reemplaza *tf* por la función $1 + \log(tf)$ para calcular los pesos de los términos del corpus.

El código utilizado para el proceso de vectorización se encuentra en el notebook `4_Training_models.ipynb` de [GitHub](#).

4.7. Fase de entrenamiento de modelos de clasificación

En este apartado, se llevarán a cabo los principales pasos para llegar a encontrar el mejor modelo que sea capaz de predecir la polaridad de un tuit con máxima precisión posible. La idea es comparar varios clasificadores y seleccionar el mejor de ellos para así hallar los parámetros óptimos que se utilizarán en el entrenamiento final. También se analizará cómo pueden llegar a influir algunos factores en el modelo final como por ejemplo, la existencia o no de *stopwords* o *palabras vacías* en el corpus.

Los modelos de clasificación que se estudiarán en este trabajo son:

- Multinomial Naive Bayes.
- Support Vector Machine (SVM).
- Regresión Logística.
- Random Forest.

El código se puede consultar en el notebook `4_Training_models.ipynb` de [GitHub](#).

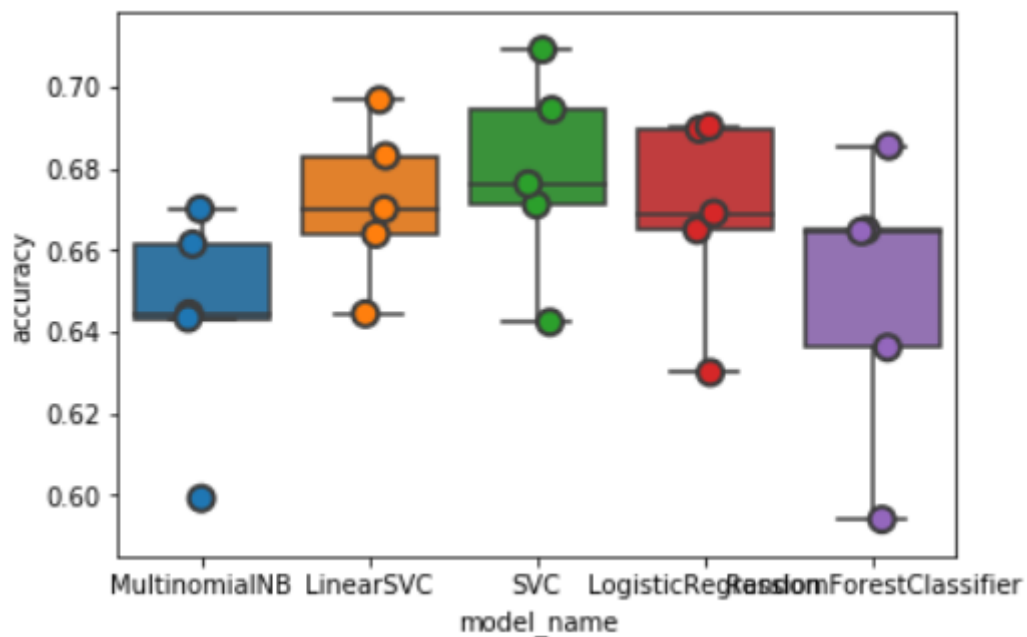
4.7.1. Conjuntos de entrenamiento y test

Como siempre que se trata un problema de Machine Learning, hacemos la división entre conjunto de entrenamiento y conjunto test, reservando el 80% para entrenamiento y 20% para test.

4.7.2. Validación cruzada o cross validation

Uno de los problemas al que nos enfrentamos siempre que entrenamos un modelo de Machine Learning con varios parámetros es el sobreajuste de los datos, esto suele pasar cuando el volumen es pequeño y los parámetros del modelo tienden a ajustarse a los datos de test igual de bien que a los de entrenamiento. Para que esto no ocurra, una buena técnica es realizar una validación cruzada o *cross validation* la cual consiste en dividir el conjunto de datos en un grupo de entrenamiento y un grupo de test varias veces. El resultado será la media aritmética obtenida en cada una de las particiones evaluadas por modelo, en base a ello se elegirán los dos mejores modelos descartando así al resto.

El código utilizado para esta sección está disponible en el notebook **4_Training_models.ipynb**. Los resultados obtenidos fueron los siguientes:



```
model_name
LinearSVC          0.671672
LogisticRegression 0.668796
MultinomialNB      0.643713
RandomForestClassifier 0.649057
SVC                0.678663
Name: accuracy, dtype: float64
```

Como se puede apreciar, los modelos con mejor evaluación fueron **Support Vector Machine (SVM)** con una media de *accuracy* del **0.6786**, seguido del modelo de **Regresión Logística** con una media de *accuracy* del **0.6687**. El resto de los clasificadores serán descartados en las siguientes fases por tener niveles más bajos.

4.7.3. Búsqueda de mejores parámetros con GridSearch

Una vez que tenemos los mejores clasificadores, es necesario investigar el rendimiento de cada uno, para ello utilizaremos *GridSearchCV*. Éste realizará todas las pruebas necesarias con los parámetros de entrada que le indiquemos.

En el caso del modelo SVM los parámetros de prueba fueron:

```
parameters_svc = {'kernel': ('linear', 'rbf'), 'C': [1, 10]}
```

Obteniendo como resultado óptimo: $\{C = 1, kernel = 'linear'\}$

En el caso del modelo de Regresión Logística los parámetros de prueba fueron:

```
parameters_logreg = {'C': (0.25, 0.5, 1.0), 'penalty': ('l1', 'l2')}
```

El resultado obtenido fue: $\{C = 1.0, penalty = 'l2'\}$

Una vez se han obtenido los parámetros óptimos se procederá a entrenar los modelos con dichos valores.

El entrenamiento se llevará a cabo en el notebook **4_Training_models.ipynb** de [GitHub](#).

5. Evaluación de resultados

La evaluación de los clasificadores se realiza atendiendo a varias métricas como el valor de *accuracy*, *precision*, *recall* y *F1-score*. Además, es posible generar la *matriz de confusión* que muestra el número de positivos verdaderos, falsos negativos, falsos positivos y negativos verdaderos. *Precision* y *recall* se pueden hallar a través de la *matriz de confusión*.

- *Accuracy*: es la precisión del clasificador, indica la dispersión del conjunto de valores obtenidos, dicho de otra forma, se refiere a lo cerca que está el resultado de una medición del valor verdadero.
- *Precision*: es la exactitud del clasificador, coincide con el porcentaje de los casos que se han clasificado correctamente.
- *Recall*: es la sensibilidad del clasificador o tasa de verdaderos positivos, es decir, la proporción de casos positivos que fueron identificados correctamente por el modelo.
- *F1-score*: es la media armónica de *precision* y *recall*. Es una buena forma de resumir la evaluación en un valor único, no obstante, es mejor atender a éstas por separado para entender mejor el comportamiento del clasificador.
- *Matriz de confusión*: es una matriz cuadrada de orden n , donde n corresponde al número de clases que participan. Las filas de la matriz corresponden a las clases reales y las columnas las que predice el modelo. Se utiliza en gran medida para ver los casos en los que el clasificador acierta o falla.

Atendiendo a estas métricas, a continuación, se muestran los resultados obtenidos al entrenar cada clasificador.

5.1. Support Vector Machine

- Incluyendo *stopwords* en el corpus, se obtiene un *accuracy* del 69.98%:

SVM Accuracy Score -> 0.6998972250770812

	predicted_negative	predicted_neutral	predicted_positive
negative	94	53	45
neutral	12	300	64
positive	21	97	287

	precision	recall	f1-score	support
-1	0.74	0.49	0.59	192
0	0.67	0.80	0.73	376
1	0.72	0.71	0.72	405
micro avg	0.70	0.70	0.70	973
macro avg	0.71	0.67	0.68	973
weighted avg	0.71	0.70	0.70	973

- Eliminando *stopwords* del corpus se obtiene un accuracy del 66.39%:

SVM Accuracy Score (without stopwords) -> 0.6639260020554985

	predicted_negative	predicted_neutral	predicted_positive	
negative	60	76	56	
neutral	13	294	69	
positive	20	93	292	
	precision	recall	f1-score	support
-1	0.65	0.31	0.42	192
0	0.63	0.78	0.70	376
1	0.70	0.72	0.71	405
micro avg	0.66	0.66	0.66	973
macro avg	0.66	0.61	0.61	973
weighted avg	0.66	0.66	0.65	973

- Considerando la vectorización en *TfidfVectorizer* con parámetro *n_gram* igual a (1,3) y dejando *stopwords* en el corpus de términos, el resultado de *accuracy* baja ligeramente hasta 69.04% si lo comparamos con el resultado obtenido en el punto primero:

SVM Accuracy Score (with n_gram = (1,3)) -> 0.6906474820143885

	predicted_negative	predicted_neutral	predicted_positive	
negative	90	54	48	
neutral	13	291	72	
positive	20	94	291	
	precision	recall	f1-score	support
-1	0.73	0.47	0.57	192
0	0.66	0.77	0.71	376
1	0.71	0.72	0.71	405
micro avg	0.69	0.69	0.69	973
macro avg	0.70	0.65	0.67	973
weighted avg	0.70	0.69	0.69	973

- Similar al apartado anterior, pero eliminando *stopwords* del corpus final se obtiene un *accuracy* del 66.70%.

SVM Accuracy Score (with n_gram = (1,3) and stopwords) -> 0.6670092497430626

	predicted_negative	predicted_neutral	predicted_positive	
negative	62	75	55	
neutral	14	295	67	
positive	20	93	292	
	precision	recall	f1-score	support
-1	0.65	0.32	0.43	192
0	0.64	0.78	0.70	376
1	0.71	0.72	0.71	405
micro avg	0.67	0.67	0.67	973
macro avg	0.66	0.61	0.62	973
weighted avg	0.67	0.67	0.65	973

5.2. Regresión Logística

- Dejando *stopwords* en el corpus de términos el resultado es un *accuracy* del 68.13%.

Logistic Regression Accuracy Score -> 0.6813977389516958

	predicted_negative	predicted_neutral	predicted_positive	
negative	78	61	53	
neutral	13	288	75	
positive	14	94	297	
	precision	recall	f1-score	support
-1	0.74	0.41	0.53	192
0	0.65	0.77	0.70	376
1	0.70	0.73	0.72	405
micro avg	0.68	0.68	0.68	973
macro avg	0.70	0.64	0.65	973
weighted avg	0.69	0.68	0.67	973

- Eliminando *stopwords* se obtiene un *accuracy* del 64.95%:

Logistic Regresion Accuracy Score (without stopwords) -> 0.6495375128468653

	predicted_negative	predicted_neutral	predicted_positive
negative	78	60	54
neutral	12	287	77
positive	14	93	298

	precision	recall	f1-score	support
-1	0.66	0.26	0.37	192
0	0.63	0.77	0.69	376
1	0.67	0.72	0.70	405
micro avg	0.65	0.65	0.65	973
macro avg	0.65	0.58	0.59	973
weighted avg	0.65	0.65	0.63	973

- Considerando la vectorización en *TfidfVectorizer* con parámetro *n_gram* igual a (1,3) y dejando *stopwords* en el corpus, el resultado es un *accuracy* del 67.72%.

Logistic Regresion Accuracy Score (withput n_gram = (1,3)) -> 0.6793422404933196

	predicted_negative	predicted_neutral	predicted_positive
negative	75	59	58
neutral	11	287	78
positive	14	92	299

	precision	recall	f1-score	support
-1	0.75	0.39	0.51	192
0	0.66	0.76	0.71	376
1	0.69	0.74	0.71	405
micro avg	0.68	0.68	0.68	973
macro avg	0.70	0.63	0.64	973
weighted avg	0.69	0.68	0.67	973

- Con parámetro *n_gram* igual a (1,3) y eliminando *stopwords* del corpus final se obtiene un *accuracy* del 65.57%.

Logistic Regresion Accuracy Score (withput n_gram = (1,3) and without stopwords) -> 0.6557040082219938

	predicted_negative	predicted_neutral	predicted_positive
negative	48	78	66
neutral	10	292	74
positive	15	92	298

	precision	recall	f1-score	support
-1	0.66	0.25	0.36	192
0	0.63	0.78	0.70	376
1	0.68	0.74	0.71	405
micro avg	0.66	0.66	0.66	973
macro avg	0.66	0.59	0.59	973
weighted avg	0.66	0.66	0.64	973

A continuación, se resumen los resultados obtenidos en términos de *accuracies*:

Clasificador	Accuracy N_gram=(1,2) Con Stopwords en corpus	Accuracy N_gram=(1,2) Sin stopwords en corpus	Accuracy N_gram=(1,3) Con stopwords en corpus	Accuracy N_gram=(1,3) Sin stopwords en corpus
SVM	0.6998	0.6639	0.6906	0.6670
Logistic Regresion	0.6813	0.6495	0.6793	0.6557

6. Conclusiones finales

Tras el entrenamiento de los dos clasificadores y las métricas analizadas se llega a la conclusión que **el mejor modelo capaz de predecir la polaridad de un tuit es Support Vector Machine con un accuracy del 69.98%**. Además, los **parámetros óptimos** calculados para SVM son C igual a 1.0 y kernel del tipo “linear”.

Por otro lado, el **modelo de Regresión Logística** ha conseguido un *accuracy* del 68.13% con **parámetros C** igual a 1.0 y **penalty** del tipo “l2”.

Por detrás quedan otros clasificadores que han obtenido **bajos resultados en cuanto a *accuracy* como el modelo Multinomial de Naive Bayes y Random Forest.**

Otra conclusión obtenida es **la influencia de *stopwords* o palabras vacías en el modelo**. En términos generales, para ambos clasificadores los valores de *accuracies* son más bajos quitándolas del corpus total, lo cual quiere decir que su presencia es importante en la predicción de la polaridad de los tuits.

En cuanto a la técnica de vectorización con *TfidfVectorizer* hay una **ligera mejoría al utilizar el parámetro *n_gram* con valor (1,2) en lugar de (1,3)**, de forma que se consideran grupos de uno y dos términos en el corpus.

7. Futuras líneas de exploración

Tras los resultados obtenidos y dado el amplio campo que suponen la exploración en el procesamiento del lenguaje natural en español, se plantean como nuevas líneas de trabajo las siguientes:

- Obtener mayor volumen de datos para mejorar el entrenamiento de los clasificadores. Teniendo en cuenta que muchos de ellos son descartados en las fases de limpieza.
- Lematizar los términos del corpus de palabras con vista a obtener mejores resultados en los clasificadores estudiados.
- Tratamiento de la ironía y sarcasmo en los tuits.
- Aplicación de técnicas de Deep Learning.

