ISSN 1870-4069

Detección de humor en tweets en español utilizando clasificadores de Scikit-learn

Alexander Gelbukh, Francisco Hiram Calvo Castro,
Mitzy Gabriela Sánchez Sánchez

Instituto Politécnico Nacional,
Centro de Investigación en Computación,
México

gelbukh@gelbukh.com, hiramcalvo@gmail.com,
msanchezs1209@alumno.ipn.mx

Resumen. La identificación automática del humor resulta una tarea compleja, ya que lo que provoca el humor aún no está completamente caracterizado. Se han presentado varios enfoques para detectar humor siendo la mayoría en inglés [3]. Esta fue una de las razones por las que se presentó durante IberLEF2019 un reto que consistió en detectar humor en tweets en español [4]. En esta competencia se abordó la detección de humor como una tarea de clasificación de textos. Retomando algunas partes de este reto, en este trabajo se presenta la comparación del desempeño de varias técnicas de aprendizaje supervisado para detectar humor en tweets en español utilizando herramientas disponibles en scikit-learn.

Palabras clave: Detección de humor, aprendizaje supervisado, clasificación.

Humor Detection in Spanish Tweets Applying Scikit-learn Classifiers

Abstract. Automatic recognition of humor is a complex task, because humor is not yet fully characterized. Several approaches to detecting humor have been presented, most of them for English language [3]. This was one of the reasons, why a challenge was presented during IberLEF2019 that consisted of detecting humor in tweets in Spanish [4]. In this competition, humor detection was addressed as a text classification task. Taking up some parts of this challenge, this paper presents the comparison of the performance of various supervised learning techniques to detect humor in tweets in Spanish using tools available in Scikit-learn.

Keywords: humor recognition, supervised learning, classification.

1. Introducción

El humor puede definirse como la tendencia de ciertas experiencias que provocan diversión o risa. Identificar que provoca reír es una tarea compleja. A Platón y a Aristóteles, se les atribuye la "teoría de la superioridad", en la que establecen que la risa es provocada por un sentimiento de superioridad sobre otras personas o uno mismo en una situación pasada. En otras palabras, nos reímos con la desgracia ajena o propia. También, existe la "teoría del alivio" propuesta por Sigmund Freud, en la que explica que la risa es una forma de liberar energía reprimida de los pensamientos. Por otra parte, la "teoría de la incongruencia" propuesta por Blaise Pascal dice que la risa se provoca cuando las personas descubren una inconsistencia entre lo que esperan que pase y lo que en realidad pasa [1]. La teoría más reciente fue propuesta por el laboratorio de investigación sobre el humor de la universidad de Colorado, en la que explican que el humor solo ocurre cuando una situación que parece atemorizante o amenazante inmediatamente se vuelve confiable o segura. Esta teoría es conocida como la "teoría de la violación benigna" [2].

Desafortunadamente, no todas las personas se ríen en las mismas situaciones, ya que el humor es algo personal independiente del origen geográfico o cultural. Por esta razón, hacer que una computadora entienda lo que es el humor es complicado, ya que no queda claro como modelar las teorías descritas anteriormente. Se ha presentado evidencia de que se puede identificar humor con un enfoque computacional utilizando extracción de características en textos cortos en conjunto con métodos de clasificación [3].

Pero aún no está completamente especificado como tienen que ser estas características en especial para el idioma español. Esta fue una de las razones que motivo la competencia "Humor Analysis based on Human Annotation" (HAHA) en IberLEF2019 [4], la cual consistió en clasificar tweets en español como divertidos o no y asignarles un puntaje de diversión dado que el tweet fuera divertido.

Retomando la primera parte de la competencia, en este trabajo se presenta un análisis comparativo del desempeño de distintos clasificadores disponibles en la librería scikit-learn en la tarea de detección de humor en tweets en español.

2. Trabajos relacionados

Uno de los primeros trabajos para caracterizar e identificar humor, fue el presentado por Mihalcea y Strapparava [3]. Ellos trabajaron en textos breves conocidos como one-liners debido a la similitud estructural que existe entre los chistes cortos, los proverbios y encabezados de noticias. Hicieron su propio corpus extrayendo chistes de sitios de internet enfocados a humor, para conformar la parte de textos no humorísticos extrajeron títulos de noticias de Reuters, proverbios y enunciados del British National Corpus (BNC) y del Open Mind Common Sense (OMCS). Utilizaron tres técnicas para la clasificación automática: heurísticas basadas características de estilo específicas de humor, características basadas en el contenido y una combinación de las anteriores. En la primera técnica buscaron palabras específicas para identificar si alguno de los textos contenía antónimos, aliteraciones o groserías de modo que si se sobrepasaba de cierto umbral se le consideraba como humorístico o no.

Fig. 1. Metodología para la evaluación de clasificadores.

Tabla 1. Comparación de los resultados de los trabajos previos.

	Mihalcea y Strapparava [3]	**Castro et al. [5]**	**Adilism [6]**	**Kevin & Hiromi [7]**	**Bfarzin [8]**
Método	Extracción de características en conjunto con clasificadores de tipo NB y SVM	Extracción de características en conjunto con clasificadores de tipo SVM, DT, MNB y KNN	BERT ajustado en conjunto con un clasificador de tipo NB	Modelos utilizando BERT en conjunto con clasificadores de tipo NB y SVM	Modelo AWD-LSTM con QRNN
Datos	Encabezados de noticias, proverbios, oraciones del BNC y chistes de tipo *one-liners*	Corpus de tweets en español etiquetados como cómicos o no cómicos	Corpus de tweets en español etiquetados como cómicos o no cómicos proporcionado por la competencia		
F1-score	N/A	• 75.5% para SVM • 67% para DT • 70.3% para MNB • 73% para KNN	82.1%	81.6%	81%
Exactitud	• 96% para encabezados • 78.15% con oraciones del BNC. • 84.5% para proverbios	• 92.5% para SVM • 88.9% para DT • 91.4% para MNB • 91.7% para KNN	85.5%	85.4%	84.6%

En la segunda técnica abordaron la detección de humor como un problema de clasificación de textos utilizando un clasificador de tipo Naïve Bayes y uno de Máquina de Soporte Vectorial. Retomando las dos técnicas anteriores, en la última técnica crean un vector con las características identificadas en la primera técnica y lo utilizan para alimentar a un clasificador. En sus resultados, resaltaron que a pesar de que encontraron características que distinguen a los chistes de los textos no humorísticos, es necesario

profundizar en la búsqueda de estas características, por ejemplo, algunos chistes que contienen antónimos los tienen escondidos en el sentido de las frases y no solamente en palabras con significado contrario, algo similar sucede con frases ambiguas, incongruentes o que carecen de sentido común.

El primer intento de identificación automática de humor en español fue presentado por Castro et al. [5]. Ellos trabajaron con tweets en español asumiendo que la longitud de un tweet evita tener partes cómicas y no cómicas al mismo tiempo. El problema de detección de humor se abordó con un enfoque de aprendizaje supervisado, es decir que propusieron una función que identifica humor a partir de datos etiquetados. Probaron con Máquinas de Soporte Vectorial, Vecinos Cercanos, Árboles de Decisión y Naïve Bayes Multinomial. Recolectaron alrededor de 47,000 tweets provenientes de cuentas enfocadas a humor y cuentas de noticias o de datos curiosos, cada tweet fue etiquetado como cómico o no cómico. Sus resultados demostraron que utilizar Máquinas de Soporte Vectorial como la mejor técnica de clasificación obteniendo un 92.5% de exactitud.

Por otra parte, en la competencia presentada en IberLEF2019 [4], se consideró como ganador al equipo que consiguiera el mejor puntaje de F1. El primer lugar de la tarea de clasificación de humor utilizó un modelo mejorado de BERT en conjunto con un clasificador de tipo Bayesiano, obteniendo un puntaje de F1 de 82.1% y una exactitud de 85% [6]. El equipo quedo en segundo lugar presentó cinco modelos diferentes basados en redes neuronales utilizado la librería de fastAI y BERT para crear un modelo de lenguaje con clasificadores de tipo Naïve Bayes y Máquinas de Soporte Vectorial, obteniendo un puntaje de F1 de 81.6% y una exactitud de 85.4% [7]. Finalmente, en el equipo que ocupó el tercer puesto utilizó un modelo AWD-LSTM con redes cuasi recurrentes, obteniendo un puntaje F1 de 81% y una exactitud de 84.6% [8].

A continuación, en la tabla 1 se muestra una comparación de los resultados obtenidos en los trabajos mencionados previamente.

3. Clasificación

En este trabajo la detección de humor se abordó como un problema de clasificación de textos. Se proponen distintos clasificadores para identificar si un tweet es cómico o no y se compara su desempeño. En la figura 1 se presenta la metodología propuesta, ver por ejemplo [14].

3.1. Obtención de los datos

En la competencia HAHA se proporcionó un corpus con 27,000 tweets [9], los tweets provenían de cuentas humorísticas y cuentas normales. Cada tweet está etiquetado cómico o no cómico y tiene asignado un puntaje de diversión.

Las etiquetas fueron establecidas por un grupo de personas que identificaban según su criterio si un tweet parecía que tenía la intención de ser gracioso o no. Si resultaba ser gracioso le tenían que asignar un puntaje de 1 a 5 estrellas y con este número de estrellas se calcula con el promedio ponderado para obtener el puntaje de diversión. El fragmento que se utilizó contiene 6,000 tweets. En el cuál cada tweet cuenta con un id, el texto del tweet, la cantidad de estrellas que se le asignó a cada tweet y el puntaje de

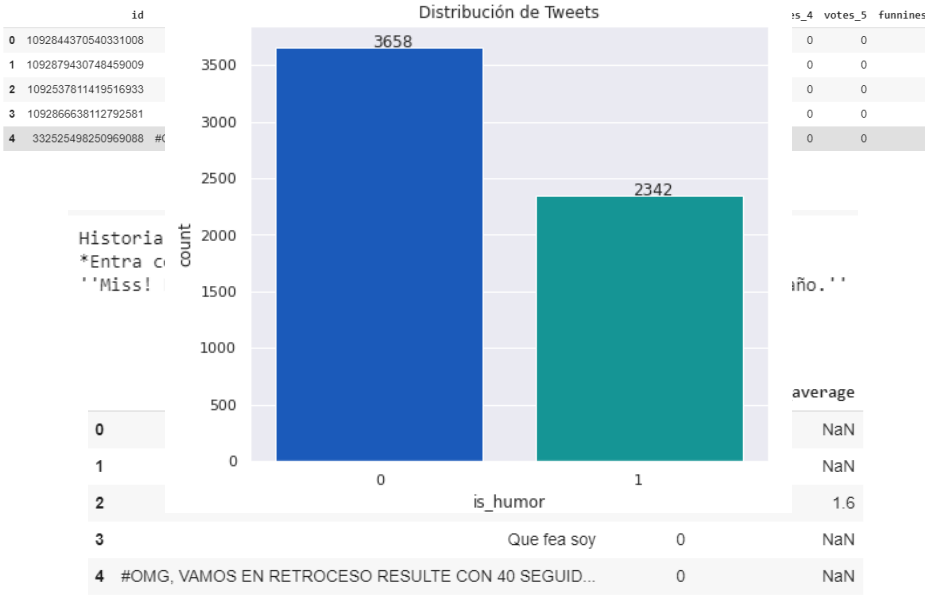

Fig. 4. DataFrame con los tweets y etiquetas.

diversión, como se observa en la figura 2. En la figura 3, se presenta un ejemplo de los tweets que contiene el corpus.

3.2. Preprocesamiento de los datos

El corpus se guardó como un DataFrame de pandas para poder manipular los datos. Pandas es una biblioteca que permite la manipulación y el análisis de datos en Python [10]. Se decidió trabajar con pandas ya que permite trabajar con distintos tipos de datos y almacenarlos en un DataFrame con el cual se pueden realizar varias operaciones.

Para este trabajo no son útiles las columnas que indican el "id" del tweet ni las que cuentan cuantas estrellas tuvo cada tweet. Por esta razón solo se conservan las columnas "text", "is_humor" y "funniness_average", como se observa en la figura 4.

Después se examinó el DataFrame en busca de datos faltantes. En la columna "funniness_average" es evidente que faltan datos, esto se debe a que los tweets que son considerados como no cómicos no tienen un puntaje de diversión asignado. Para evitar complicaciones se le asignó un puntaje de diversión de 0 a todos los tweets no graciosos. La distribución de las clases se obtuvo contando cuantos elementos hay en cada clase.

En la columna "is_humor" se puede observar que existen dos clases, el 0 representa que un tweet no es cómico y el 1 que un tweet es cómico. En la figura 5 se muestra un histograma de las clases donde se observa que hay más tweets no graciosos en el corpus. Los autores del corpus explican que esto se debe a que gran parte de los tweets extraídos de las cuentas humorísticas son chistes malos o tweets con la intención de atraer seguidores a la cuenta. En total se tienen 3648 tweets no cómicos y 2342 tweets cómicos.

El corpus se dividió en 70% para entrenar y 30% para prueba. Es decir, el conjunto de entrenamiento se conformó por 4200 tweets y el de prueba por 1800 tweets. El conjunto de características son los textos de los tweets y las etiquetas son las proporcionadas en la columna "is_humor". Se utilizó train_test_split de scikit-learn para dividir el corpus. sckit-learn es una biblioteca de Python que incluye varias herramientas de preprocesamiento, algoritmos de clasificación y métricas de desempeño [11].

Desafortunadamente, no se puede clasificar una sucesión de letras y símbolos ya que a la entrada de un clasificador se espera un dato de tipo numérico de longitud fija. Por esta razón se vectoriza los textos del corpus, de esta forma el corpus de tweets puede representarse mediante una matriz con una fila por tweet y una columna por token (palabra) en el corpus. Primero se tokeniza las palabras, después se cuenta el número de ocurrencias de cada token y finalmente se normaliza el conteo para que palabras, a este procedimiento se le conoce como "Bag of Words".

Si bien contar palabras es útil, los textos más largos tendrán valores de conteo promedio más altos que los textos más cortos, aunque hablen del mismo tema. Para evitar esto, se reducen los pesos de las palabras que aparecen con mayor frecuencia en los textos ya que proporcionan menos información que la que proporcionan las palabras que aparecen menos veces, a esta reducción se le conoce como "Term Frequency times Inverse Document Frequency" (tf-idf). En scikit-learn se utilizó la función TfidfVectorizer.

3.3. Selección de clasificadores

Los clasificadores fueron seleccionados en base a trabajos mencionados previamente. Se eligieron 5 clasificadores que vienen incluidos en la librería scikit-learn: Naïve Bayes Multinomial, Máquina de Soporte Vectorial, Árboles de Decisión, Vecinos Cercanos y Perceptrón Multi-Capa.

- Naïve Bayes Multinomial

  Es un clasificador basado en el teorema de Bayes haciendo la suposición de que cada característica es independiente de las demás características y la posición de la palabra no importa. Naïve Bayes Multinomial es frecuentemente utilizado en clasificación de textos [11,12].

- Máquina de Soporte Vectorial

  Es utilizado cuando se tiene dos grupos de datos que son linealmente separables. El algoritmo busca un hiperplano que sirve de límite, de forma que todos los puntos estén en promedio igual de separados de él y lo más separado que se pueda. Los puntos más cercanos al hiperplano se llaman vectores soporte [11,12].

- Árboles de Decisión

  Los árboles de decisión aprenden de los datos para aproximar una función con un conjunto de reglas de decisión si-entonces-otro (if-then-else). Cuanto más

profundo es el árbol, más complejas son las reglas de decisión y mejor se ajusta el modelo [11,12].

– Vecinos Cercanos

  Dado un conjunto de datos etiquetados se estima la clase a la que pertenece un dato sin etiqueta comparando ese dato con cada dato existente. Después se toman los datos más parecidos al dato desconocido y se observan sus etiquetas. Se toman los K (K<20) datos más cercanos y se busca la clase que tiene más elementos dentro de la selección, a esa clase se asigna el dato desconocido. El método es preciso, pero requiere de bastante carga computacional [11,12].

– Perceptrón Multi-Capa

  Las redes neuronales son un paradigma biológicamente inspirado que permite que una computadora pueda aprender a partir de un conjunto de datos. La suma de las entradas multiplicadas por sus pesos asociados determina el impulso nervioso que recibe la neurona. Este valor, se procesa mediante una función de activación que devuelve un valor que se envía como salida de la neurona [11,12].

El proceso de clasificación fue idéntico para cada clasificador, a continuación, se muestra el proceso para el clasificador de Naïve Bayes:

1. Crear el pipeline

```
text_clf_nb = Pipeline([('tfidf', TfidfVectorizer()), ('clf',
MultinomialNB())])
```

2. Ajustar el modelo

```
text_clf_nb.fit(X_train, y_train)
```

3. Hacer las predicciones

```
predictions = text_clf_nb.predict(X_test)
```

3.4. Calculo de las métricas de desempeño

Se utilizó la matriz de confusión para comparar los resultados obtenidos con cada clasificador ya que permite visualizar el desempeño de cada algoritmo. Como se trata de un problema de clasificación binaria, cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real, como se muestra en la figura 6.

De esta forma el desempeño puede medirse en función del número de verdaderos positivos (VP), falsos positivos (FP), falsos negativos (FN) y verdaderos negativos (VN).

Al igual que en la competencia HAHA [4], se consideró como el mejor método clasificación al que tuvo el puntaje F1 más alto. Esta métrica porque combina precisión y sensibilidad, lo que resulta de utilidad cuando las clases no están bien distribuidas, ya que puede suceder que se obtenga un valor alto de precisión en la clase mayoritaria y sensibilidad baja en la clase minoritaria [13].

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Fig. 6. Matriz de confusión.

		Predicción	
		Positivo	Negativo
Real	Positivo	1051	51
	Negativo	460	238

Fig. 7. Matriz de confusión del clasificador de tipo Naïve Bayes Multinomial.

		Predicción	
		Positivo	Negativo
Real	Positivo	886	216
	Negativo	259	439

Fig. 8. Matriz de confusión del clasificador de tipo Máquina de Soporte Vectorial.

		Predicción	
		Positivo	Negativo
Real	Positivo	831	271
	Negativo	354	344

Fig. 9. Matriz de confusión del clasificador de tipo Árboles de Decisión.

		Predicción	
		Positivo	Negativo
Real	Positivo	1005	97
	Negativo	474	224

Fig. 10. Matriz de confusión del clasificador de tipo K Vecinos Cercanos.

		Predicción	
		Positivo	Negativo
Real	Positivo	825	277
	Negativo	267	431

Fig. 11. Matriz de confusión del clasificador de tipo Perceptrón Multicapa.

La matriz de confusión y el puntaje F1 se calcularon con las métricas de scikit-learn *"confusión_matrix"* y *"classification_report"*.

3.5. Comparación del desempeño

A continuación, en las figuras de la 7 a la 11 se muestran las matrices de confusión para cada clasificador. Se puede observar que con Naïve Bayes se pudieron identificar bien los tweets cómicos. De forma contraria, los tweets no cómicos fueron clasificados

Tabla 2. Comparación del desempeño de los clasificadores.

Clasificador	Exactitud	Precisión	Sensibilidad	F1-score
SVM	**0.736**	**0.67**	**0.63**	**0.648**
MLP	0.696	0.61	0.62	0.612
DT	0.654	0.56	0.50	0.514
NB	0.716	0.82	0.34	0.482
KNN	0.682	0.70	0.32	0.439

en su mayoría como cómicos, lo que provoca que la exactitud sea alta y el puntaje F1 sea bajo. Esto significa que este clasificador tiene problemas para diferenciar las clases, pero cuando identifica una es altamente confiable.

Con el clasificador de Máquina de Soporte Vectorial también se identifican bien los tweets cómicos y además también se clasificaron bien los tweets no cómicos. Con este resultado se espera que la precisión y la sensibilidad sean altas, lo que significa que el clasificador identifica bien ambas clases.

Con el clasificador de tipo Árboles de Decisión se observa un resultado similar al obtenido con SVM, pero no es tan bueno como el anterior ya que comente más errores lo que provoca que la exactitud y el puntaje F1 sean más bajos.

Con el clasificador de tipo Vecinos Cercanos se obtiene un resultado parecido al obtenido con Naïve Bayes, ya que el clasificador puede identificar bien los tweets cómicos, pero comete bastantes errores al identificar tweets no cómicos. Esto resulta en una precisión alta pero una sensibilidad baja lo que provoca un puntaje F1 bajo.

Con el Perceptrón Multicapa se pueden identificar bien ambas clases, desafortunadamente también se cometen varios errores al diferenciar las clases. Lo que provoca que el resultado de la clasificación sea bueno, pero no el mejor.

En la tabla 2, se muestra la comparación de las métricas de desempeño de cada clasificación. El clasificador de tipo Máquina de soporte vectorial resulto tener los mejores puntajes en todas las métricas, seguido por el Perceptrón Multicapa. Estos dos clasificadores diferencian bien ambas clases, a pesar de que el clasificador de tipo Naïve Bayes tiene una exactitud elevada, no se puede considerar como un buen clasificador porque detecta humor en tweets que no tenían la intención de ser cómicos.

4. Conclusiones y trabajo a futuro

De la comparación del desempeño de clasificadores para detección de humor, se observó que el clasificador de tipo Máquina de Soporte vectorial obtuvo el mejor puntaje de F1 con 64.8%. Este resulto ser el mejor clasificador ya que funciona bien para clasificadores binarios y tiende a no generalizar el error. Como el objetivo es identificar humor, según los resultados obtenidos no sería conveniente utilizar un clasificador de tipo Naïve Bayes, ya que clasifica una gran cantidad de tweets como cómicos cuando no lo son.

Es interesante que en este trabajo el clasificador bayesiano resultara entre los clasificadores de menor desempeño, cuando en los trabajos previos resulta ser uno de

los mejores clasificadores. Posiblemente se deba a que, en los otros trabajos a parte de la vectorización, se hace una extracción de características basada en el contenido.

Como trabajo futuro se propone trabajar la extracción de características, por ejemplo, tomar en consideración el número de veces que aparecen los signos de admiración, la cantidad de emojis utilizados o identificar las etiquetas con # que brinden más información acerca del contenido del tweet y a mejorar el desempeño de los clasificadores.

Referencias

1. Morreall, J.: Philosophy of humor. The Stanford Encyclopedia of Philosophy. Edward N. Zalta (ed.) (2016)
2. Humor Research: Benign Violation Theory. Lab at the University of Colorado. http://leeds-faculty.colorado.edu/mcgrawp/Benign_Violation_Theory.html (2016)
3. Mihalcea, R., Strapparava, C.: Learning to laugh (automatically): computational models for humor recognition. Computational Intelligence, 22(2), pp. 128–142 (2006)
4. HAHA@IberLEF: Human analysis based on human annotation (2016)
5. Castro, S., Cubrero M., Garat, D., Moncecchi, G.: Is This a Joke? Detecting humor in Spanish tweets. IBERAMIA, pp. 139–150 (2016)
6. Ismailov, A.: Humor analysis based on human annotation Challenge at IberLEF: First Place Solution (2019)
7. Bird, K.: High level HAHA architecture (2019)
8. Farzin, B., Czpla, P., Howard, J.: Applying a pre-trained language model to Spanish twitter humor prediction. IberLEF (2019)
9. Castro, S., Chiruzzo, L., Rosá, A., Garat, D., Moncecchi, G.: A crowd-annotated Spanish corpus for humor analysis. In: Proceedings of the 6th International Workshop on NLP for Social Media, pp. 7–11 (2018)
10. Pandas: The pandas development team. Zenodo, version 1.0.5 (2020)
11. Pedregosa, F. et al.: Scikit-learn: machine learning in python. JMLR, pp. 2825–2830 (2011)
12. Harrington, P.: Machine Learning in Action. NY, Manning (2012)
13. Barrios, J.: La matriz de confusión y sus métricas. Health Big Data (2019)
14. Sidorov, G.: Construcción no lineal de n-gramas en la lingüística computacional. SMIA, 166 p. (2013)