MCDI
MAESTRÍA EN CIENCIA DE DATOS E INFORMACIÓN

INFOTEC CENTRO DE INVESTIGACIÓN E INNOVACIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

DIRECCIÓN ADJUNTA DE INNOVACIÓN Y CONOCIMIENTO
GERENCIA DE CAPITAL HUMANO
POSGRADOS

“CUANTIFICACIÓN Y PREDICCIÓN DE POSIBLES CASOS DE DEPRESIÓN EN USUARIOS DE TWITTER UTILIZANDO MACHINE LEARNING”

REPORTE ANALÍTICO DE EXPERIENCIA LABORAL
Que para obtener el grado de MAESTRO EN CIENCIA DE DATOS E INFORMACIÓN

Presenta:

Enrique Isidoro Vázquez Ramos

Asesores:

Dr. Sabino Miranda Jiménez
Dr. Eric Sadit Téllez Avila

Aguascalientes, noviembre, 2021.

CONACYT

INFOTEC POSGRADOS

Autorización de Impresión

INFOTEC

AUTORIZACIÓN DE IMPRESIÓN Y NO ADEUDO EN BIBLIOTECA

Maestría en Ciencia de Datos e Información, MCDI.

Ciudad de México, 1 de junio de 2022.

La Gerencia de Capital Humano / Gerencia de Investigación hacen constar que el trabajo de titulación intitulado:

"Cuantificación y predicción de posibles casos de depresión en usuarios de Twitter utilizando machine learning".

Desarrollado por el alumno: **Enrique Isidoro Vázquez Ramos** y bajo la asesoría del **Dr. Sabino Miranda Jiménez** y el **Dr. Eric Sadit Téllez Avila**; cumple con el formato de Biblioteca. Por lo cual, se expide la presente autorización para impresión del proyecto terminal al que se ha hecho mención.

Asimismo, se hace constar que no debe material de la biblioteca de INFOTEC.

Vo. Bo.

Lic. Juan Ramón Abarca Damián
Coordinador de Biblioteca

Anexar a la presente autorización al inicio de la versión impresa del trabajo referido que ampara la misma.

C.p. Servicios Escolares

Agradecimientos

Agradezco a Dios,
sin él nada sería posible.

Al amor de mi vida y esposa Izadora,
que siempre me apoya en los momentos más difíciles.

A mi papá, mamá y hermanas y familia en general,
espero que estén orgullosos.

A la sociedad en general,
esperando que en algún momento esta tesis les sea de ayuda.

Agradezco a mi amigo Víctor Silva Cuevas,
compañero de la maestría por su paciencia y ayuda.

También a todos mis profesores
por su gran trabajo y por su tiempo.

Tabla de contenido

Índice de figuras

Índice de cuadros

Glosario

“A”

Análisis sintáctico: Es el análisis de las funciones sintácticas o relaciones de concordancia y jerarquía que guardan las palabras cuando se agrupan entre sí en forma de sintagmas, oraciones simples y oraciones compuestas de proposiciones.

“L”

Lematización: Es un proceso lingüístico que consiste en, dada una forma flexionada, hallar el lema correspondiente. El lema es la forma que por convenio se acepta como representante de todas las formas flexionadas de una misma palabra.

“M”

Machine learning: Es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan.

“N”

Normalización: Segmentación de las palabras, formato de las palabras y segmentación de las oraciones en el texto.

“T”

Tokenización: Una instancia de un tipo en un texto dado.

Capítulo 1 - Planteamiento

Introducción

En el presente trabajo de investigación se abordarán conceptos enfocados a la creación y aplicación de un modelo de machine learning[1] alimentado por un corpus de tweets que dará como resultado una herramienta para poder predecir posibles casos de depresión en personas. Durante el transcurso del proyecto se estarán detallando las herramientas, teorías, análisis, procesos y resultados llevados a cabo para la realización de dicho modelo. En resumen, el trabajo se divide en 5 capítulos, en el primero se explica el planteamiento en general, la motivación y problemática, los objetivos y la contribución que se aportará al lector.

El segundo capítulo muestra los conceptos y definiciones básicas que se deben tener para la construcción del modelo de machine learning, aquí mismo se muestran algunos estudios previos que diversos investigadores han realizado similares al que se aplica en esta investigación. En el tercer capítulo se habla acerca de la metodología que se aplicará, haciendo énfasis en los datos, herramientas y procedimiento.

El cuarto capítulo define la estructura de los datos y da a conocer los resultados obtenidos de los experimentos realizados. Finalmente, en el quinto y último capítulo se podrán leer las conclusiones del proyecto, así como las áreas de oportunidad que surgieron al completarlo.

Capítulo 1 - Planteamiento

1.1 Motivación

El problema social referente a que las personas sufran depresión es un tema preocupante en nuestra sociedad actual, la cifra de personas que la padecen crece cada vez más. La vida acelerada, competitiva y estresante (entre más factores) que se vive hoy en día ha dado paso a que muchas personas caigan en depresión. La gente

[1] Un modelo de machine learning es la salida de información que se genera cuando entrena su algoritmo de machine learning con datos. Después del entrenamiento, al proporcionar un modelo con una entrada, se le dará una salida (Hastie, Trevor, Tibshirani, Robert y Friedman, 2009).

encuentra diversas válvulas de escape para desahogarse, entre ellas están las redes sociales, y dentro de las redes sociales más populares se encuentra Twitter. Es aquí donde el presente proyecto entra, haciendo uso de herramientas de inteligencia artificial y de un corpus (textos de Twitter) clasificado por expertos donde se identifican personas con depresión y sin depresión, se desarrollará una propuesta para predecir posibles casos de depresión en usuarios de Twitter localizados en México.

1.2 Problemática

Los gobiernos actuales, así como diferentes organizaciones alrededor del mundo están desarrollando propuestas. Un ejemplo de lo anterior se puede consultar en (Hosseinifard, Hassan y Rostami, 2013) que habla sobre salud mental, para poder prevenir y/o detectar casos de depresión en las personas.

El tema de la depresión está muy ligado al del suicidio (Kulasinghe, Jayasinghe, Rathnayaka, Karunarathne, Suranjini Silva y Anuradha Jayakodi, 2019) las personas que atentan contra su vida generalmente sufren algún tipo de depresión y/o problemas que creen no tienen solución. Es posible que no cuenten con personas cercanas a ellas y se sientan solos, tratando de desahogarse haciéndose notar en sus perfiles de redes sociales.

Dado el enorme abanico de herramientas y tecnologías que se tiene actualmente, como son las redes sociales (Twitter, Facebook, etc.) y la inteligencia artificial (machine learning), muchas propuestas para apoyar en el estudio de este problema se han desarrollado. Los resultados han sido variados (Joulin, Grave, Bojanowski y Mikolov, 2017). (Conneau, Kiela, Schwenk, Barrault y Bordes, 2017) como desarrollo de léxicos y predicción de desempeño. Se utilizarán técnicas de machine learning y Procesamiento de Lenguaje Natural (PLN) sobre datos de redes sociales para detectar a usuarios con posibles casos de depresión.

1.3 Objetivos

1.3.1 Objetivo General

Proponer un modelo de clasificación automática[2] basado en aprendizaje computacional que sea capaz de detectar posibles casos de depresión entre los usuarios de Twitter. En el caso particular de este proyecto, el enfoque será trabajar sólo en lo que se refiere a México.

1.3.2 Objetivos Específicos

- Obtención y caracterización de la base de conocimiento.
- Creación y prueba de modelos.
- Selección del modelo que se desempeñe mejor para su aplicación en un conjunto de datos grande de tweets geolocalizados en México.
- Análisis de los resultados.

1.4 Contribución

Este proyecto aportará al lector una herramienta para poder pronosticar posibles casos de depresión en las personas, lo anterior se logra haciendo uso de un algoritmo de machine learning. El algoritmo cuenta con varias implementaciones las cuales se explicarán más adelante en esta investigación. Los resultados, así como el algoritmo generado en este trabajo estarán disponibles a través de un repositorio de Github.

[2] Consiste en utilizar técnicas de Inteligencia Artificial sobre un conjunto de elementos para ordenarlos por clases o categorías (Hastie, Trevor, Tibshirani, Robert y Friedman, 2009).

Capítulo 2 - Marco Teórico

Capítulo 2 - Marco Teórico

A continuación, se abordarán los conceptos de las herramientas, librerías, y tecnologías usadas en este proyecto de investigación.

2.1 Aprendizaje Supervisado

Esta técnica es una rama de la inteligencia artificial usada para entrenar o crear modelos a partir de datos numéricos o textos que han sido previamente etiquetados, estas etiquetas son realizadas manualmente y por lo general son nombradas como las clases del problema, el aprendizaje supervisado cuenta con una gran variedad de aplicaciones como el uso de las redes sociales para monitorear la salud mental (McClellan, Ali, Mutter, Kroutil y Landwehr, 2016) por mencionar alguna.

En (Román, 2019) se expone que un problema de clasificación[3] es una subcategoría del aprendizaje supervisado en la que el objetivo es predecir las etiquetas de clase categóricas de las nuevas instancias, basándonos en observaciones pasadas.

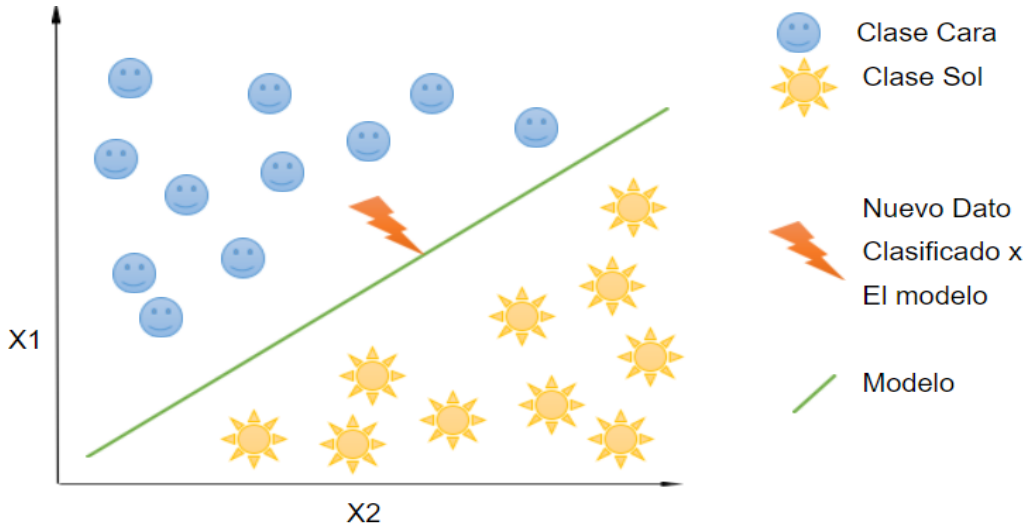

Figura 1. *Ejemplo bidimensional de un problema de clasificación.*

Fuente: Elaboración propia.

[3] Se caracterizan por tener una variable cualitativa 'Y' como respuesta.

Las clases identificarán a cada individuo de los datos, por ejemplo, supongamos que los datos son extraídos de un corpus de tweets que cuenta con textos escritos por usuarios seleccionados al azar, como es de esperarse habrá gran variedad de combinaciones de unidades léxicas con diversos significados, una de estas combinaciones podría ser el texto "*¿¡estoy muy feliiiiz!? #2021*" que, siguiendo la definición de aprendizaje supervisado se clasificaría manualmente con la clase "Positivo". Entre más grande sea el corpus mejor serán los resultados.

La finalidad de esta técnica es, que una vez entrenado los datos con algoritmos de machine learning, se podrán clasificar nuevos datos sin la necesidad de hacerlo manualmente.

2.2 Partición de los Datos y Métricas

Dentro del aprendizaje supervisado se requiere de la realización de pruebas a los modelos creados para de esta forma medir su eficiencia, en este proyecto se aplicará la estrategia de particionar los datos en dos[4], con el primero se entrenarán los modelos de machine learning y con el segundo se realizarán pruebas para generar las métricas de desempeño (véase 2.2.2).

2.2.1 Entrenamiento - Prueba

El conjunto de datos manejado en este proyecto proviene del INAOE, fue en este instituto donde se etiquetó a los usuarios de Twitter. La forma en que lo hicieron fue la siguiente: los usuarios etiquetados con depresión auto declararon esta condición y los usuarios etiquetados sin depresión son cuentas genéricas. Cabe señalar que en este proceso no intervinieron especialistas en el campo de la salud mental[5]. Los datos se encuentran en formato json mostrando la siguiente estructura (cada línea del archivo json contiene 3 campos: user, klass y tweets):

- User. Hay 317 usuarios (se debe tener en cuenta que las clases no están balanceadas, véase 4.1), 92 considerados con depresión y 225 sin depresión.

[4] Conjunto de entrenamiento y conjunto de prueba.
[5] Psiquiatras y psicólogos.

- Klass. El número 1 representa a los considerados con depresión y el número 0 a los considerados sin depresión.
- Tweets. El campo tweets es una lista por usuario de tamaño no constante donde cada elemento es un tweet que contiene todos los datos de éste: fecha, id, texto, fecha de creación, etc. En este proyecto se utilizará sólo la variable texto.

La partición de los datos tanto por usuarios como por tweets se realizará dividiendo la información (arreglos) en subconjuntos de pruebas y trenes aleatorios, quedando los porcentajes de la siguiente manera:

- Entrenamiento. Tendrá el 70% de los datos.
- Prueba. Tendrá el 30 %.

2.2.2 Métricas de desempeño

Las métricas que se utilizarán para calificar los resultados serán las siguientes:

- *Precision*. La precisión es la relación entre las observaciones positivas predichas correctamente[6] y el total de observaciones positivas predichas[7] (Renuka, 2016).

$$precisión = \frac{TP_{(k)}}{TP_k + FP_k}$$

- *Recall*. Es la proporción de observaciones positivas predichas correctamente a todas las observaciones en la clase real (Renuka, 2016).

$$recall = \frac{TP_{(k)}}{TP_k + FN_k}$$

[6] Son los aciertos que el modelo obtuvo.
[7] Es el total de todas las predicciones.

- *F1 Score Measure*. Es el promedio ponderado de Precision y Recall (Renuka, 2016).

$$F1 = \frac{2 * precisión_{(k)} * recall_{(k)}}{precisión_{(k)} + recall_{(k)}}$$

$$micro\ F1 = \frac{2 * precisión * recall}{precisión + recall}$$

$$(macro\ o\ ponderado)\ F1 = \sum_{K \in \Gamma} W_k * F1_k$$

$$W_k = \{macro \rightarrow 1/\Gamma\ ponderado \rightarrow \frac{|\{u \in D | u = k\}|}{|D|}\}$$

- *Accuracy*. Es simplemente una relación entre la observación predicha correctamente y el total de ellas (Renuka, 2016).

$$accuracy = \frac{TP + TN}{\#muestras}$$

Donde:

TP = número de verdaderos positivos.

TN = verdaderos negativos.

FP = número de falsos positivos.

FN = número de falsos negativos.

k = clase.

2.3 Algoritmos de Clasificación

Como se mencionó anteriormente la estrategia en este proyecto será utilizar algoritmos de clasificación supervisada (véase 2.1), pues se conoce la cantidad y el tipo de las clases asociadas a los textos que se trabajarán. En los puntos siguientes se

definen los algoritmos utilizados en este proyecto que serán requeridos para la clasificación automática de los tweets.

2.3.1 Soporte Lineal para la Clasificación de Vectores

LinearSVC (Linear Support Vector Classification) (Pedregosa, 2011) es una implementación de clasificación de vectores de soporte para el caso de un núcleo lineal[8].

Como otros clasificadores, LinearSVC toma como entrada dos matrices[9]: una matriz 'X' de forma (n_samples, n_features) que contiene las muestras de entrenamiento[10] y una matriz 'Y' de etiquetas de clase[11] (cadenas o enteros), de forma (n_samples).

2.3.2 Clasificador de Vecinos más Cercanos

La clasificación basada en los vecinos más cercanos es un tipo de aprendizaje basado en instancias o aprendizaje no generalizador[12]. La clasificación se calcula a partir de un voto de mayoría simple de los vecinos más cercanos de cada punto: a un punto de consulta se le asigna la clase de datos que tiene más representantes dentro de los vecinos más cercanos del punto (Pedregosa, 2011).

El nombre también se refiere a la clase de algoritmos llamados "no paramétricos". Estos son algoritmos basados en instancia, que se caracterizan por memorizar el conjunto de datos de entrenamiento, y el aprendizaje perezoso es un caso particular de estos algoritmos, asociados con coste computacional cero durante el aprendizaje (Román, 2019). El proceso que sigue el algoritmo es (Román, 2019):

- Escoge el número de k y la distancia.
- Encuentra el k vecino más cercano de la muestra que se pretende clasificar.
- Asigna la etiqueta de clase por votación mayoritaria.

---

[8] Es el núcleo de la aplicación lineal definida por la matriz (Pedregosa, 2011).
[9] Tablas bidimensionales de números en cantidades abstractas que pueden sumarse y multiplicarse (Pedregosa, 2011).
[10] Datos que serán utilizados para el entrenamiento (Pedregosa, 2011).
[11] Etiqueta que se asigna a cada uno de los datos (Pedregosa, 2011).
[12] No intenta construir un modelo interno general, sino que simplemente almacena instancias de los datos de capacitación (Pedregosa, 2011).

El algoritmo encuentra las k[13] muestras que son más cercanas al punto que se quiere clasificar, basando sus predicciones en la distancia métrica (Román, 2019). La principal ventaja es que se adapta a los nuevos datos de entrenamiento, al ser un algoritmo basado en la memoria. La desventaja es que el coste computacional se incrementa linealmente con el tamaño de los datos de entrenamiento (Román, 2019).

2.3.3 Tf-Idf

Esta es una estrategia específica de tokenización, conteo y normalización, a la que se llama la bolsa de palabras o la representación de "bolsa de n-gramas". Los documentos se describen por las ocurrencias de las palabras mientras se ignora por completo la información de posición relativa de las palabras en el documento (Pedregosa, 2011).

En un corpus[14] de texto grande, algunas palabras estarán muy presentes, por lo tanto, llevarán muy poca información significativa sobre el contenido real del documento. Si tuviéramos que alimentar los datos de conteo directo directamente a un clasificador, esos términos muy frecuentes sombrearían las frecuencias de términos más raros aún más interesantes. Tf significa frecuencia de término, mientras que Tf-Idf significa frecuencia de término por frecuencia inversa de documento.

Originalmente, este era un esquema de ponderación de términos desarrollado para la recuperación de información que también ha encontrado un buen uso en la clasificación y agrupación de documentos (Pedregosa, 2011).

$$tfidf\,(t,d,D) = tf\ (t,d)\ * \mathrm{idf}\,(\mathrm{t},\mathrm{D})$$

$$tf\ (t,d) = \log log\,(1 + freq(t,d))$$

$$idf\,(t,D) = log\left(\frac{N}{|(d \in D : t \in d)|}\right)$$

[13] Variable que representa a cualquier valor posible.
[14] Es el total de los datos.

Donde:

t = término

d = documento

N = número total de documentos en el corpus

dƐD: tƐd = número de documentos en los que aparece el término “t”

2.3.4 Clasificador Naive Bayes para modelos multinomiales

MultinomialNB implementa el algoritmo Bayes ingenuo para datos distribuidos multinomial mente, y es una de las dos variantes Bayes ingenuas clásicas utilizadas en la clasificación de texto. La distribución está parametrizada por vectores (Pedregosa, 2011).

En este algoritmo se asume que las variables predictoras[15] son independientes entre sí. En otras palabras, que la presencia de una cierta característica en un conjunto de datos no está en absoluto relacionada con la presencia de cualquier otra característica (Román, 2019).

Lo consigue proporcionando una forma de calcular la probabilidad[16] ‘posterior’ de que ocurra un cierto evento *A,* dadas algunas probabilidades de eventos ‘anteriores’ (Román, 2019). Los pasos generales para un algoritmo Naive Bayes supervisado son: (Román, 2019):

-Convertir el conjunto de datos en una tabla de frecuencias[17].

-Crear una tabla de probabilidad calculando las correspondientes a que ocurran los diversos eventos.

-La ecuación Naive Bayes se usa para calcular la probabilidad posterior de cada clase.

-La clase con la probabilidad posterior más alta es el resultado de la predicción.

[15] Son las variables de interés en un experimento (Román, 2019).

[16] Es una medida del grado de certidumbre de que un suceso pueda ocurrir (Román, 2019).

[17] Agrupación de datos en categorías mutuamente excluyentes que indican el número de observaciones en cada categoría (Román, 2019).

2.3.5 MicroTC - TextModel

MicroTC (Tellez, Moctezuma, Miranda y Graff, 2018) es intencionalmente simple, por lo que solo se implementó una pequeña cantidad de funciones. Sin embargo, utiliza algunas herramientas complejas de numpy y scikit-learn. El número de dependencias está limitado y cumplido por casi cualquier distribución de Scientific Python, por ejemplo, Anaconda.

Está diseñado para abordar los problemas de clasificación de texto de forma agnóstica[18], ya que es independiente del dominio y del lenguaje.

El punto de entrada del núcleo de μTC es microtc.textmodel.TextModel, que puede verse como una función con la forma **m (texto)** → **R^*d*** donde '*d*' es el vocabulario, es decir, la dimensión del espacio vectorial. Como puede verse, '*m*' puede usarse para transformar un texto en un vector y, en consecuencia, puede usarse para transformar un conjunto de entrenamiento de pares, texto y etiqueta, en un conjunto de entrenamiento de pares, vectores y etiqueta, que puede ser utilizado directamente por cualquier algoritmo de aprendizaje supervisado para obtener un clasificador de texto (Tellez, Moctezuma, Miranda y Graff, 2018).

2.4 Estado del Arte

Es importante resaltar que una de las principales etapas que debe desarrollarse en un proyecto de titulación es la realización de su Estado del Arte. Ya que no sólo muestra un panorama de cómo se ha trabajado anteriormente el tema de este proyecto también ofrece información sobre los avances y las estrategias que se esperan en un futuro.

2.4.1 Aprendizaje Automático

Existe una gran cantidad de trabajos enfocados en el aprendizaje automático (Hastie, Trevor, Tibshirani, Robert y Friedman, 2009) hablan sobre como aprender de los datos, de cómo en un escenario típico, se tiene una medida de resultado, generalmente cuantitativa o categórica, que se desea predecir basado en un conjunto de

[18] Considera que la veracidad de ciertas afirmaciones son desconocidas.

características. Tienen un conjunto de datos de entrenamiento, en el que observaron el resultado y las mediciones de características para un conjunto de objetos. Usando esos datos construyeron un modelo de predicción, que permite predecir el resultado para nuevos objetos.

En (Pratama y Purwarianti, 2017) trabajan con técnicas de aprendizaje automático en sus dos variantes: supervisado y no supervisado. Ven al público de Bandung (ciudad indonesia) como un usuario activo de las redes sociales, en especial de Twitter, para que los ciudadanos denuncien sus quejas. Para gestionar las quejas denunciadas hacen uso del aprendizaje automático y así detectar los temas de los tweets. El enfoque de aprendizaje supervisado lo implementaron para clasificar el tema de los tweets en función de las quejas, mientras que el enfoque de aprendizaje no supervisado se utiliza para agrupar los tweets de quejas en función de la similitud de la información detallada contenida en las quejas. Su enfoque de aprendizaje supervisado lo evaluaron utilizando accuracy, precisión, recall y medida F1 (véase 2.2.2). Con su enfoque de aprendizaje no supervisado, utilizaron el valor del índice de agrupación para evaluar los grupos de temas detectados.

En (Larsen, Boonstra, Batterham, O'Dea, Paris y Christensen, 2015) describen que la mayoría de los datos de investigación sobre problemas de salud mental se recopilan a través de encuestas, esto trae como consecuencia que la información recolectada no proporciona una visión en tiempo real[19] de los individuos encuestados. Mencionan que una solución para obtener información en tiempo real es extraer ésta desde las redes sociales, pues los individuos participan en ellas continuamente. Describieron un sistema que llamaron "We feel" donde analizan variaciones globales y regionales de la expresión emocional. Su fuente de información fue Twitter, obteniendo así en un periodo de 12 semanas información en forma de textos y mediante el uso de PCA reconocer patrones que clasificaron como positivos y negativos.

En (Chopade, Edwards, Khan, Andrade y Pu, 2019). presentan un marco basado en el aprendizaje automático para la identificación de evidencias en las habilidades cognitivas y de aprendizaje socioemocional en la resolución colaborativa de

[19] Acontecimientos que están sucediendo en el momento presente.

problemas de la diádica. En otras palabras, demostrar las habilidades y la dinámica del trabajo en equipo como comportamientos verbales y no verbales y cómo éstos se pueden capturar y analizar mediante la recopilación pasiva de datos. Para lo anterior, analizaron datos de juegos multimodales y desarrollaron modelos de habilidades y técnicas para su medición, utilizaron técnicas de procesamiento de lenguaje natural (PLN) como la inserción de oraciones y la bolsa de palabras obteniendo resultados que muestran el desempeño de los jugadores clasificándolos como exitosos y no exitosos.

2.4.2 Minería de Texto

Dada la enorme cantidad de información[20] que actualmente se genera en el mundo, es necesario contar con técnicas avanzadas de minería de texto, así lo hacen notar (Hogenboom, Frasincar, Kaymak y Jong, 2011) al mencionar que la utilización de información extraída en los procesos de toma de decisiones[21] se convierte en un problema cada vez más urgente y difícil. Un problema omnipresente es el hecho de que la mayoría de los datos inicialmente no están estructurados y se describen usando un lenguaje natural, comprensible para el ser humano, que hace que los datos estén limitados en el grado en que son interpretados por las máquinas. Este problema frustra la automatización de, por ejemplo, los procesos de recuperación de información vital (IR) y extracción de información (IE), utilizados para la toma de decisiones.

En (Tekiner, Tsuruoka, Tsujii y Ananiadou, 2009) mencionan que las tareas de etiquetar datos de secuencia, la fragmentación y el reconocimiento de entidades nombradas son una de las tareas más importantes dentro de la minería de textos.

Cuando se trata de grandes cantidades de datos, Text Mining se refiere al aprendizaje de información a partir de texto preprocesado.

En (Agnihotri, Verma y Tripathi, 2014) analizan que, debido al uso intensivo de dispositivos electrónicos en la actualidad, la mayor parte de la información está

[20] Grandes cantidades de textos que se pueden recolectar de todas las redes sociales públicas.
[21] Conjunto de estrategias, aplicaciones, datos, productos, tecnologías y arquitectura técnicas, los cuales están enfocados a la administración y creación de conocimiento sobre el medio, a través del análisis de los datos existentes (Hogenboom, Frasincar, Kaymak y Jong, 2011).

almacenada en formato digital y gran parte de ella se encuentra en forma de texto. Mencionan que extraer el conocimiento, como la búsqueda de patrones o agrupaciones de palabras similares, es uno de los temas más importantes en la actualidad[22]. El documento se centra en extraer la información más importante de los textos, utilizaron el lenguaje de programación R para sus experimentos, su algoritmo utiliza la similitud coseno para medir la distancia entre palabras y de esa manera agruparlas usando k-medias.

Por medio de la minería de texto, a menudo utilizando el procesamiento del lenguaje natural, la información se extrae de textos de diversas fuentes, como noticias, mensajes, tweets o blogs, y se representa y almacena de forma estructurada.

2.4.3 Clasificación de pacientes con depresión usando machine learning

Existe una amplia variedad de aplicaciones[23] donde se puede utilizar machine learning, como es conocido, en el presente proyecto se aplicará esta técnica para detectar a usuarios de Twitter con posible depresión. Este mismo argumento también lo plantean (Hosseinifard, Hassan y Rostami, 2013) pues estudiaron el análisis no lineal de la señal EEG (electroencefalograma) para discriminar pacientes con depresión y controles normales. Cuarenta y cinco pacientes deprimidos no medicados y cuarenta y cinco sujetos normales participaron en este estudio.

Para discriminar los dos grupos[24], utilizaron el vecino K más cercano, el análisis discriminante lineal y la regresión logística[25] como los clasificadores. Todas las características no lineales y el clasificador LR lograron una precisión de clasificación del 90%. Su estudio muestra que el análisis no lineal de EEG puede ser un método útil para discriminar pacientes deprimidos y sujetos normales.

En (Hooda, Saxena, Madhulika y Yadav, 2017) abordan el tema de que la depresión es un problema crítico que puede afectar a las personas de diversas formas. Su objetivo es predecir a aquellas personas que ni siquiera saben que sufren del padecimiento. Para lo anterior usaron tres modelos: uso de clasificadores de

[22] Los patrones y similitudes de palabras dan un contexto natural al problema a resolver.
[23] Seguridad de datos, comercio financiero, cuidado de la salud, marketing personalizado, etc.
[24] Pacientes con depresión y controles normales (Hosseinifard, Hassan y Rostami, 2013).
[25] Clasificadores utilizados por los investigadores (Hosseinifard, Hassan y Rostami, 2013).

aprendizaje automático y WEKA[26], uso de métodos de imágenes y aprendizaje automático y uso de factores de riesgo.

2.4.4 Análisis de Texto

En (Hakdağli, Özcan y Oğul, 2018) mencionan que, con la tecnología actual en desarrollo, el acceso de las personas a la información[27] y su producción ha alcanzado un nivel muy rápido. Esta información generada y obtenida se crea instantáneamente, se ingresa en sistemas de datos y se actualiza. Describen que estas fuentes de transmisión de datos se pueden transformar en valiosos resultados de análisis cuando se manejan con métodos específicos. Determinaron un campo de datos de texto para realizar análisis de datos generados instantáneamente y utilizaron Twitter, la plataforma más rica para datos de texto instantáneos. Twitter genera instantáneamente una variedad de datos en grandes cantidades y los presenta como código abierto. Utilizaron el entorno de análisis de transmisión de Apache Spark para analizar los recursos, el análisis de la situación lo realizaron utilizando la máquina de vectores de soporte, los árboles de decisión y los algoritmos de regresión logística de Apache Spark[28]. De esta manera, con la información generada instantáneamente y enfocando estos recursos y estrategias al análisis de sentimientos se pueden detectar posibles casos de depresión en usuarios de Twitter en tiempo real y de esta forma reducir el tiempo de diagnóstico.

En (Xiaomeng, Chong y Zipei, 2020) analizan que, con el desarrollo de la tecnología de comunicación móvil, el aumento de las redes sociales en línea brinda oportunidades para que las personas expresen sus puntos de vista[29], lo que juega un papel importante en el análisis y monitoreo de la opinión pública. El tema principal de su estudio es el análisis emocional de los usuarios de Twitter en inglés. Aplican tecnología Reptile para obtener y limpiar tweets relacionados con China, luego de

[26] Plataforma de software para el aprendizaje automático y la minería de datos escrito en Java y desarrollado en la Universidad de Waikato (Hooda, Saxena, Madhulika y Yadav, 2017).
[27] Internet, medios de comunicación, globalización, etc.
[28] Motor ultrarrápido para el almacenamiento, procesamiento y análisis de grandes volúmenes de datos.
[29] Cualquier persona con acceso a internet puede expresar su opinión en cualquier red social.

etiquetar datos referidos al diccionario de palabras fuertes que se compone de frecuencias estadísticas de palabras, usan GloVe[30] para generar vectores de palabras y usan redes neuronales convolucionales[31] (CNN) para clasificar tweets automáticamente.

Así establecen un modelo de CNN preciso para analizar los tweets populares relacionados con China de 2014 a 2018. Sus resultados muestran que China ocupó una posición importante en la política mundial en 2018 con grandes comentarios negativos en Twitter. El número de simpatizantes mayor se dio en 2014, pero tiene un descenso repentino en 2015 y se mantiene en torno al 45% de 2016 a 2018. Al igual que en la anterior investigación (Hakdağli, Özcan y Oğul, 2018), si se cambia el enfoque hacia el análisis de sentimientos enfocado a la depresión, éste estudio puede arrojar nuevos resultados y nuevas métricas para la posible detección de personas con depresión.

2.5 Representaciones semánticas / Vectores de palabras (word embeddings)

El lenguaje natural[32] que usamos los humanos no puede ser procesado literalmente por las máquinas, dado que los textos trabajados en este proyecto están realizados por personas y escritos en lenguaje natural, se deben buscar estrategias que le faciliten a la máquina la manipulación e interpretación de éstos.

Word Embedding es una técnica utilizada para modelar textos escritos en lenguaje natural, su finalidad consiste en crear vectores numéricos que representen a los textos relacionándolos semánticamente. Es muy eficiente al momento de entrenar grandes cantidades de datos por tal motivo es utilizado en este proyecto.

Cada vector captura la información semántica[33] de la palabra o frase asociada según convenga y es visto en un plano multidimensional, es decir, la longitud

[30] Algoritmo de aprendizaje no supervisado para obtener representaciones vectoriales de palabras (Xiaomeng, Chong y Zipei, 2020).
[31] Tipo de rede neuronal artificial donde las “neuronas” corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria (V1) de un cerebro biológico (Xiaomeng, Chong y Zipei, 2020).
[32] Lenguaje utilizado por los humanos de manera natural (español, inglés, francés, etc.).
[33] Significado de las palabras.

y la orientación que tiene cada vector puede determinar el contexto de su palabra o frase asociada, esto se da gracias a que, si dos o más vectores presentan comportamientos parecidos, las palabras o frases que representan cuentan con significados similares.

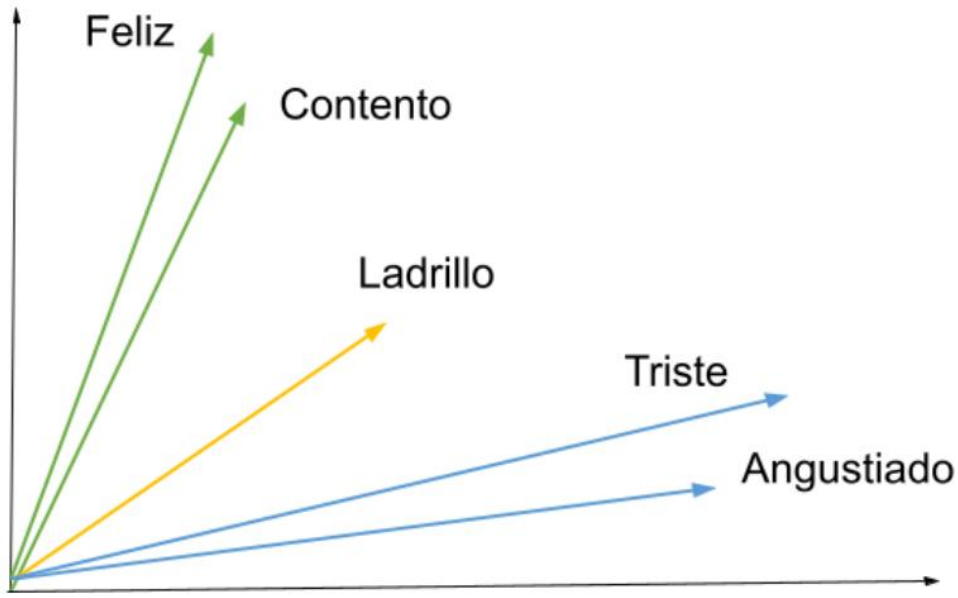

Figura 2. *Ejemplo bidimensional de una representación vectorial*

Fuente: Elaboración propia.

La figura 2 muestra un ejemplo de cómo los vectores pueden relacionarse entre sí dada su longitud y dirección, se aprecia que las palabras “feliz” y “contento” tienen similitud, pero están muy alejadas de las palabras “triste” y “angustiado” pues su contexto es diferente, en cambio el vector que representa la palabra “ladrillo” permanece neutral a los dos contextos anteriores.

2.5.1 FastText

FastText es una biblioteca de código abierto[34], gratuita y liviana que permite a los usuarios aprender representaciones de texto y clasificadores de texto. Funciona en hardware estándar y genérico (Joulin, Grave, Bojanowski y Mikolov, 2017).

Para calcular vectores de palabras[35], se necesita un gran corpus de texto. Dependiendo del corpus, los vectores de palabras capturan información diferente (Joulin, Grave, Bojanowski y Mikolov, 2017).

FastText proporciona dos modelos para calcular representaciones de palabras: skipgram y cbow ('bolsa de palabras continúa'). El modelo de skipgram aprende a predecir una palabra objetivo gracias a una palabra cercana. Por otro lado, el modelo cbow predice la palabra objetivo según su contexto. El contexto se representa como una bolsa de palabras contenidas en una ventana de tamaño fijo alrededor de la palabra de destino (Joulin, Grave, Bojanowski y Mikolov, 2017).

Los parámetros[36] más importantes del modelo son su dimensión y el rango de tamaño de las subpalabras. La dimensión controla el tamaño de los vectores, cuanto más grandes son, más información pueden capturar, pero requiere que se aprendan más datos. Pero, si son demasiado grandes, son más difíciles y lentos de entrenar. De forma predeterminada, se usan 100 dimensiones, pero cualquier valor en el rango de 100 a 300 es igual de popular. Las subpalabras son todas las subcadenas contenidas en una palabra entre el tamaño mínimo y el tamaño máximo. Por defecto, se toman todas las subpalabras entre 3 y 6 caracteres, pero otro rango podría ser más apropiado para diferentes idiomas (Joulin, Grave, Bojanowski y Mikolov, 2017).

Una forma sencilla de comprobar la calidad de un vector de palabra es mirar a sus vecinos más cercanos. Esto da una intuición del tipo de información semántica

[34] Software cuyo código fuente y otros derechos que normalmente son exclusivos para quienes poseen los derechos de autor, son publicados bajo una licencia de código abierto o forman parte del dominio público (Joulin, Grave, Bojanowski y Mikolov, 2017).
[35] Listas que contienen a un determinado número de palabras (Joulin, Grave, Bojanowski y Mikolov, 2017).
[36] Elemento de un sistema que es útil o crítico al identificar el sistema o al evaluar su rendimiento, estado, condición, etc. (Joulin, Grave, Bojanowski y Mikolov, 2017).

que los vectores son capaces de capturar (Joulin, Grave, Bojanowski y Mikolov, 2017). Por ejemplo:

[(0.891, ‘tomate’), (0.889, ‘espinacas’), (0.892, ‘zanahoria’)]

2.5.2 InferSent

InferSent es un método de incrustación de oraciones que proporciona representaciones semánticas para oraciones. Está capacitado en datos de inferencia de lenguaje natural y se generaliza bien para muchas tareas diferentes (Conneau, Kiela, Schwenk, Barrault y Bordes, 2017).

Para su funcionamiento se requieren descargar los vectores GloVe o fastText. Se debe tener en cuenta que infersent1 se entrena con GloVe e infersent2 se entrena con fastText. Éste último también elimina el relleno de ceros con agrupación máxima (Conneau, Kiela, Schwenk, Barrault y Bordes, 2017).

Infersent genera una matriz con n vectores de dimensión 4096. La velocidad es de alrededor de 1000 frases por segundo con un tamaño de lote de 128 en una sola GPU[37].

[37] Unidad de procesamiento de gráficos.

Capítulo 3 - Metodología

Capítulo 3 – Metodología

3.1 Descripción General

A continuación, se detallará el procedimiento[38] llevado a cabo en la investigación, el ambiente de desarrollo, el servidor[39] utilizado, el ecosistema donde se trabajará y finalmente las tecnologías y librerías utilizadas.

3.2 Conjunto de Datos

Para la elaboración de los experimentos propuestos se hizo uso de un corpus compuesto de 1,046,582 tweets realizados por 317 usuarios, éstos últimos están clasificados con dos etiquetas: “con depresión” (1) y “sin depresión” (0) por un grupo de expertos en el tema (INAOE: “usuario_depresion. json”). El análisis se divide en dos grandes grupos:

- **Por Usuario.** A cada uno de los 317 usuarios se le concatenan todos sus tweets creando así un único texto por usuario, el cual, contiene todos los textos que el usuario escribió, este texto en consecuencia tiene una dimensión relativamente grande (el promedio de tweets escritos por usuario es de 3,302), pero en total son pocos textos generados (317 textos). Como la clasificación de los expertos se hizo por usuario (INAOE: “usuario_depresion. json”), entonces la clasificación de los textos es la misma.
- **Por Tweet.** En este caso a cada tweet se le considera como un texto independiente, la única relación que tiene con los usuarios es la clasificación, es decir, la clasificación de cada tweet es la misma a la que tiene el usuario que lo escribió. Cabe mencionar que para este análisis las dimensiones de los textos son pequeñas, pero el total de textos generados es muy alto, pues son 1,046,582 tweets en total.

[38] Conjunto de acciones que tienen que realizarse todas igualmente, para obtener los mismos resultados bajo las mismas circunstancias (Hastie, Trevor, Tibshirani, Robert y Friedman, 2009).
[39] Sistema que proporciona recursos, datos, servicios o programas a otros ordenadores, conocidos como clientes, a través de una red (Hastie, Trevor, Tibshirani, Robert y Friedman, 2009).

3.3 Preprocesamiento

El preprocesamiento que se le aplicó a los textos fue el siguiente: se eliminaron los acentos, signos, puntuaciones, tildes, dígitos, letras repetidas (oooyeeeeee => oye); se transformaron todos los caracteres a minúscula, se eliminaron las siguientes palabras puntuales: http, rt. Finalmente se aplicó una eliminación de stopwords (véase Glosario) en español. Cabe mencionar que se omitió la limpieza de emojis y la aplicación de stemming, lo anterior para no perder peso en las emociones de los usuarios.

3.4 Procedimiento

Con la finalidad de seguir una secuencia estructurada de los pasos a desarrollar en este proyecto de investigación y de llevar un orden en los experimentos a realizar se propone la siguiente línea de trabajo:

- Extraer y leer el corpus (INAOE: “usuario_depresion. json”) mediante el uso de la librería de python. (Tellez, Moctezuma, Miranda y Graff, 2018).
- Modelar el problema de identificación de posible depresión, como uno de aprendizaje supervisado.
- Se cuenta con un corpus anotado con posibles usuarios con depresión en investigaciones paralelas por investigadores nacionales.
- Preparación y procesamiento del corpus para su explotación; preparación y procesamiento de los datos a ser anotados de manera automática.
- Preparación y procesamiento del corpus para su explotación; preparación y procesamiento de los datos a ser anotados de manera automática.
- Creación y selección de modelos para la solución del problema; se usará validación cruzada. (Tellez, Moctezuma, Miranda y Graff, 2018), (Hastie, Trevor, Tibshirani, Robert y Friedman, 2009), (Hogenboom, Frasincar, Kaymak y Jong, 2011).
- Análisis de los modelos (Hastie, Trevor, Tibshirani, Robert y Friedman, 2009), los modelos utilizados en este proyecto (Soporte Lineal para la Clasificación de Vectores, Clasificador de Vecinos más Cercanos, Tf-Idf, Clasificador Naive Bayes para modelos multinomiales y microTC – TextModel) son tan

solo una pequeña fracción de los que están actualmente disponibles queda a criterio del lector utilizar el modelo que más le convenga (véase 2.3).

- Etiquetado masivo y análisis de los resultados. Esto con el propósito de crear un dataset funcional para su correcto uso en el aprendizaje automático que se le aplicará, así mismo, al tener la información correctamente etiquetada se podrán obtener resultados sin sesgos (Hastie, Trevor, Tibshirani, Robert y Friedman, 2009), (Hogenboom, Frasincar, Kaymak y Jong, 2011).

Con la anterior secuencia se espera tener un orden en el proyecto y evitar así retrabajos y confusiones, para de esta forma presentar resultados lo más claros posibles sin dificultades de apreciación.

3.5 Ambiente de Desarrollo

3.5.1 Servidor utilizado para correr el código

El servidor utilizado para la ejecución del código fue proporcionado por INFOTEC[40] campus Aguascalientes[41]. Este proyecto se llevó a cabo haciendo uso de la terminal en un sistema operativo Ubuntu, en éste se corrieron los scripts de Python correspondientes a esta investigación. Las especificaciones del servidor son las siguientes:

- Servidor Intel® Xeon®[42]
- CPU[43] E5-2680 v4, 2.40 GHz[44]
- Sistema Operativo Ubuntu[45] 16.04.2 LTS
- 256 GB de memoria Ram[46].
- 24 procesadores[47]

[40] Centro de Investigación e Innovación en Tecnologías de la Información y Comunicación.
[41] Ciudad de la zona central de México.
[42] Xeon es una familia de microprocesadores Intel para servidores.
[43] Unidad Central de Procesamiento, procesa todas las instrucciones del hardware y software.
[44] Mide la frecuencia de un chip electrónico, el más conocido es el procesador.
[45] Sistema operativo de software libre y código abierto.
[46] Donde se almacenan de forma temporal los datos de los programas que son utilizados en tiempo real.
[47] CPU.

A continuación, se muestra un diagrama que resume la arquitectura del proyecto:

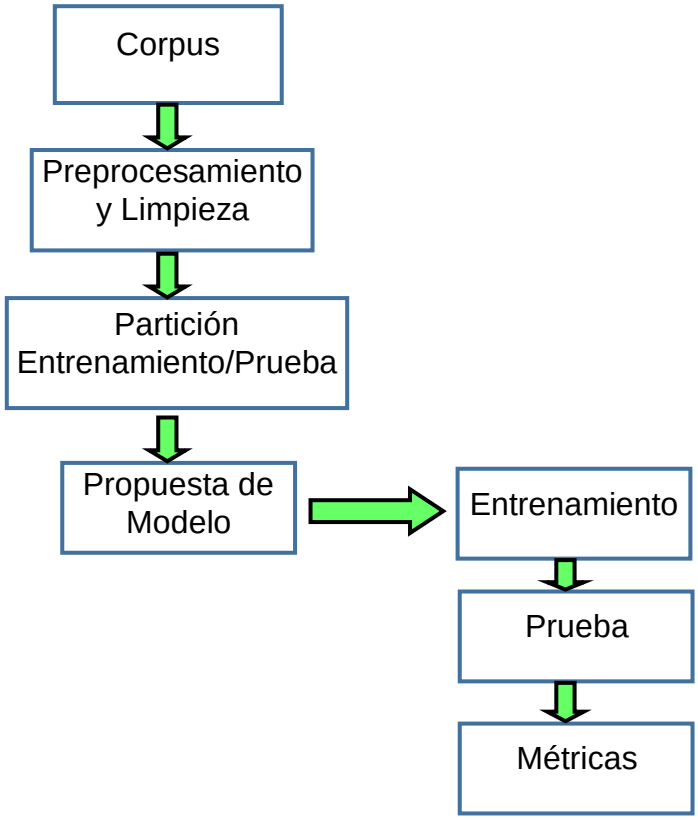

Figura 3. *Arquitectura general del proyecto*

Fuente: Elaboración propia.

3.5.2 Tecnologías y librerías utilizadas.

El lenguaje de programación usado en este proyecto es Python (versión 3.7.6), el cual cuenta con librerías instaladas por default que son utilizadas en este proyecto. Sin embargo, es necesario la instalación de otras para el correcto funcionamiento del código aplicado en este trabajo, las librerías que se deben instalar son:

- conda 4.8.3
- scikit-learn 0.24.2
- microtc 2.2.5
- fasttext 0.9.2
- torch 0.4.1
- infersent

Capítulo 4 - Análisis y Evaluación

Capítulo 4 - Análisis y Evaluación

Este capítulo muestra los experimentos realizados, los resultados logrados y la configuración detallada de cada uno de éstos. En los siguientes puntos se detalla la estructura, partición, modelos realizados y los resultados de los experimentos.

4.1 Estructura de los Datos por Usuario

El siguiente cuadro detalla la estructura de los datos por Usuario, así como su partición de entrenamiento y prueba.

Estructura	Clases	Textos	Usuarios	FastText	InferSent
Completo	NEG	225	225	225	225
	POS	92	92	92	92
Entrenamiento	NEG	159	159	159	159
	POS	62	62	62	62
Prueba	NEG	66	66	66	66
	POS	30	30	30	30

Cuadro 1. *Distribución de las clases.*

Fuente: Elaboración propia.

Como se puede observar en el cuadro anterior no existe un balance entre las clases, con dominio claro en la clase NEG, es decir, en los usuarios sin depresión. Esto se puede explicar debido a que de manera natural la mayoría de la gente y por consecuencia la mayoría de las personas que escriben en Twitter no sufren de depresión[48].

La partición de los datos fue de 70% para el conjunto de entrenamiento[49] y el 30% para el conjunto de prueba[50]. La estructura de las particiones se puede observar en el cuadro 1.

[48] Existen más personas sin depresión que con ella (Fernández 2019).
[49] Datos que se usarán para entrenar el modelo de machine learning.
[50] Datos que se usarán para probar el modelo de machine learning.

4.2 Estructura de los Datos por Tweet con y sin preprocesamiento

Para el análisis de los datos por tweet se encontró que debido a la alta cantidad de textos no se pudieron generar los experimentos con FastText y con InferSent (aún con el uso del Clúster), la solución fue aplicar un preprocesamiento de limpieza a los textos (véase 3.3) para disminuir sus dimensiones, de esta manera la experimentación con FastText se pudo lograr, aunque con InferSent persistió el problema de no poder generar los experimentos por la alta cantidad de textos.

Estructura	Clases	Textos	FastText
Completo	NEG	745,783	745,783
	POS	300,799	300,799
Entrenamiento	NEG	522,048	522,048
	POS	210,559	210,559
Prueba	NEG	223,735	223,735
	POS	90,240	90,240

Cuadro 2. *Distribución de las clases con preprocesamiento*

Fuente: Elaboración propia.

Estructura	Clases	Textos
Completo	NEG	746,271
	POS	300,923
Entrenamiento	NEG	522,421
	POS	210,614
Prueba	NEG	223,850
	POS	90,309

Cuadro 3. *Distribución de las clases sin preprocesamiento*

Fuente: Elaboración propia.

Así que se generaron dos corpus por tweet, uno con preprocesamiento (véase 3.3) donde se incluye la experimentación con FastText (cuadro 2) y el otro sin éste, donde no se toma en cuenta FastText ni InferSent (cuadro 3).

4.3 Experimentos y Resultados

En el proceso de experimentación se usaron varias combinaciones de modelos[51] para de esta forma tener una visión más amplia[52] de los resultados que se pueden presentar en problemas de clasificación enfocados a la detección de depresión. En los siguientes puntos están los cuadros con los resultados arrojados por los experimentos, las métricas mostradas son precisión, recall y F1 en sus tres formas (selección automática de parámetros, macro y micro) y accuracy. Para una mejor redacción se abreviará la selección automática de parámetros como **SAP** y la selección editada de parámetros como **SEP**. La partición seleccionada fue con una proporción de 70% para entrenamiento y 30% para prueba.

Todos los experimentos fueron realizados en el lenguaje de programación Python y ejecutados en el clúster (véase 3.5.1). Como se mencionó anteriormente, el uso de Word embeddings en este proyecto se llevó a cabo con FastText y con InferSent, los vectores resultantes fueron de dimensión 300 en el caso de FastText y de dimensión 4096 en el caso de InferSent. Los vectores fueron entrenados de la siguiente manera:

- FastText (véase 2.5.1): crawl-300d-2M-subword.bin
- InferSent1 (véase 2.5.2): GloVe.840B.300d.txt

  Parámetros del modelo de InferSent:
- params_model (véase 2.5.2) = {'bsize': 64, 'word_emb_dim': 300, 'enc_lstm_dim': 2048,'pool_type': 'max', 'dpout_model': 0.0, 'version': model_version}

[51] Véase cuadro 8.
[52] Al contar con varias combinaciones de modelos se tiene un mayor rango de resultados que pueden compararse.

4.3.1 Métricas por usuarios con preprocesamiento

Modelo	Métrica / Test	SAP	Macro	Micro
TfIdf / MultinomialNB	Precision	0.72	0.36	0.72
	Recall	1	0.50	0.72
	F1	0.84	0.42	0.72
	Accuracy	0.72	-	-
MicroTc / SAP	Precision	0.78	0.89	0.80
	Recall	1	0.65	0.80
	F1	0.88	0.67	0.80
	Accuracy	0.80	-	-
MicroTc / SEP	Precision	0.76	0.88	0.78
	Recall	1	0.66	0.78
	F1	0.86	0.67	0.78
	Accuracy	0.78	-	-
SVM / FastText	Precision	0.72	0.36	0.72
	Recall	1	0.50	0.72
	F1	0.84	0.42	0.72
	Accuracy	0.72	-	-
SVM / InferSent	Precision	0.90	0.92	0.91
	Recall	0.99	0.85	0.91
	F1	0.94	0.87	0.91
	Accuracy	0.91	-	-
KNN / FastText	Precision	0.87	0.82	0.84
	Recall	0.93	0.78	0.84
	F1	0.90	0.79	0.84
	Accuracy	0.84	-	-
KNN / InferSent	Precision	0.80	0.70	0.76
	Recall	0.90	0.65	0.76
	F1	0.84	0.67	0.76
	Accuracy	0.76	-	-

Cuadro 4. *Desempeños por usuario con preprocesamiento*

Fuente: Elaboración propia.

En el cuadro 4 se puede observar que se corrieron 7 diferentes modelos donde la combinación SVM / Infersent obtuvo los mejores resultados, las métricas más bajas son para la combinación de SVM / FastText y para Tfldf. En esta etapa, todos los experimentos se corrieron sin problema en el clúster.

4.3.2 Métricas por Usuarios sin preprocesamiento

Modelo	Métrica / Test	SAP	Macro	Micro
MicroTc / SAP	Precision	0.85	0.92	0.88
	Recall	1	0.80	0.88
	F1	0.92	0.83	0.88
	Accuracy	0.88	-	-
MicroTc / SEP	Precision	0.83	0.91	0.85
	Recall	1	0.77	0.85
	F1	0.90	0.80	0.85
	Accuracy	0.85	-	-

Cuadro 5. *Desempeños por usuario sin preprocesamiento*

Fuente: Elaboración propia

En el cuadro 5 se aprecia que se corrió solo 1 modelo que fue microTc, esto debido a que en la prueba anterior (véase 4.3.1) se corrieron todos los modelos con un preprocesamiento (véase 3.3) pero en este caso no existió tal limpieza, así que se probó el modelo de MicroTc en una forma pura con dos configuraciones de parámetros diferentes: SAP y SEP (véase 4.5) para de esta forma poder comparar resultados. Los mejores resultados se obtuvieron con la configuración SAP. En esta etapa, los experimentos se corrieron sin problema en el clúster, dando prioridad al modelo microTc y observando cómo se comporta cuando se utilizan sus parámetros automáticos y cuando se les edita sin un preprocesamiento previo.

Lo anterior pues microTc cuenta con su propia limpieza (véase 4.5 microTc Parámetros), dado esto, en el actual proyecto se puede observar que las métricas arrojadas por microTc con una limpieza tradicional antes de aplicar la suya (cuadro 4) son menores a las arrojadas solo aplicando la limpieza de microTC.

4.3.3 Métricas por Tweet con preprocesamiento

Modelo	Métrica / Test	SAP	Macro	Micro
SVM / FastText	Precision	0.72	0.61	0.71
	Recall	0.97	0.52	0.71
	F1	0.83	0.48	0.71
	Accuracy	0.71	-	-
MicroTc / SAP	Precision	0.89	0.87	0.88
	Recall	0.95	0.83	0.88
	F1	0.92	0.85	0.88
	Accuracy	0.88	-	-
MicroTc / SEP	Precision	0.89	0.86	0.88
	Recall	0.94	0.83	0.88
	F1	0.91	0.84	0.88
	Accuracy	0.88	-	-

Cuadro 6. *Desempeños por tweet con preprocesamiento*

Fuente: Elaboración propia

El cuadro 6 muestra sólo 3 modelos, esto por el hecho de que el clúster no corrió las demás pruebas, la causa fue el gran volumen de textos que se manejaron al trabajar por tweet y no por usuario. Sólo hay resultados para SVM / FastText y microTC, siendo este último en su versión SAP el que obtuvo mejores métricas.

4.3.4 Métricas por Tweet sin preprocesamiento

Modelo	Métrica / Test	SAP	Macro	Micro
MicroTc / SAP	Precision	0.84	0.77	0.81
	Recall	0.90	0.74	0.81
	F1	0.87	0.75	0.81
	Accuracy	0.81	-	-
MicroTc / SEP	Precision	0.84	0.79	0.82
	Recall	0.92	0.74	0.82
	F1	0.88	0.76	0.82
	Accuracy	0.82	-	-

Cuadro 7. *Desempeños por tweet sin preprocesamiento*

Fuente: Elaboración propia

En el cuadro 7 al igual que en el cuadro 5 sólo se corrió microTc, esto para poder ver cómo se comporta este modelo con sus propios parámetros sin que haya un preprocesamiento previo en los datos que trabajará, al contrario de lo que sucedió antes, en esta prueba, se lograron mejores resultados cuando se realizó un preprocesamiento previo al que microTc realiza, mejorando de esta forma sus métricas.

4.4 Discusión de los resultados obtenidos

El actual proyecto aplicó varios experimentos con diferentes combinaciones de clasificadores, embeddings y modelos para la resolución de un problema de clasificación aplicando machine learning.

La investigación está basada en dos enfoques que a su vez están ramificados en múltiples experimentos todos con el mismo fin pero que dieron resultados diferentes. Los dos enfoques y sus experimentos correspondientes se muestran en los cuadros 8 y 9 respectivamente.

Por Usuario		Por Tweet	
Con Prepro-cesamiento	**Sin Preprocesa-miento**	**Con Preprocesa-miento**	**Sin Preprocesa-miento**
MicroTC SAP	MicroTC SAP	MicroTC SAP	MicroTC SAP
MicroTC SEP	MicroTC SEP	microTC SEP	MicroTC SEP
SVM/FastText	*	SVM/FastText	*
TfIdf	*	*	*
SVM/Infer-Sent	*	*	*
KNN/FastText	*	*	*
KNN/Infer-Sent	*	*	*

Cuadro 8. *Enfoques y Modelos*

Fuente: Elaboración propia

* No se pudieron correr debido a su alto consumo de recursos computacionales.

Del análisis realizado los mejores resultados fueron:

Por Usuario				
Modelo	**Métrica** / **Test**	**Resultado**	**Macro**	**Micro**
SVM/Infer-Sent con pre-procesa-miento	Precision	0.90	0.92	0.91
	Recall	0.99	0.85	0.91
	F1	0.94	0.87	0.91
	Accuracy	0.91	-	-

Por Tweet				
Modelo	**Métrica** / Test	**Resultado**	**Macro**	**Micro**
MicroTC SAP con preproce-samiento	Precision	0.89	0.87	0.88
	Recall	0.95	0.83	0.88
	F1	0.92	0.85	0.88
	Accuracy	0.88	-	-

Cuadro 9. *Mejores Resultados*

Fuente: Elaboración propia

4.5 MicroTC Parámetros:

Los siguientes parámetros de MicroTC fueron los que se utilizaron en los experimentos SAP y SEP correspondientemente.

MicroTc SEP	
docs	None
text	'text'
num_option	'delete'
usr_option	'delete'
url_option	'delete'
emo_option	'group'
hashteg_option	'delete'
ent_option	'none'
lc	True
del_dup	True
del_punc	True
del_diac	True
token_list	[[2,1], - 1,3,4]
token_min_filter	0
token_max_filter	1
select_ent	False
select_suff	False
select_conn	False
weighting	'tfidf'

MicroTc SAP	
docs	None
text	'text'
num_option	'group'
usr_option	'group'
url_option	'group'
emo_option	'group'
hashteg_option	'none'
ent_option	'none'
lc	True
del_dup	True
del_punc	False
del_diac	True
token_list	[-1]
token_min_filter	0
token_max_filter	1
select_ent	False
select_suff	False
select_conn	False
weighting	'tfidf'

Cuadro 10. *Parámetros MicroTc*

Fuente: Elaboración propia

Para una mejor redacción se abreviará la selección automática de parámetros como SAP y SEP. A continuación, se muestran los parámetros que contiene el clasificador microTC y su significado:

- **docs**: Debe ser una lista que contendrá el corpus.
- **text** : En el caso de que el corpus sea un diccionario, el nombre del campo será la clave del texto .
- **num_option**: Transformación de números, posibles valores: none, group y delete.
- **usr_option:** Transformación de los usuarios, posibles valores: none, group y delete.
- **url_option**: Transformación de las url, posibles valores, posibles valores: none, group y delete.

- **emo_option**: Transformaciones de emojis y emoticones, posibles valores: none, group y delete.
- **hashtag_option:** Transformaciones de hashtags, posibles valores: none, group y delete.
- **ent_option**: Transformación de entidades, posibles valores: none, group y delete.
- **lc**: Minúsculas
- **del_dup**: Remover símbolos duplicados.
- **del_punc**: Elimina los símbolos de puntuación.
- **del_diac**: Elimina símbolos diacríticos.
- **token_list**: Tokens[53] > 0 qgrams[54] < 0 word-grams[55].
- **token_min_filter**: Conserva los tokens que aparecen más veces que el parámetro (utilizado en la clase de ponderación).
- **token_max_filter**:Conserva los tokens que aparecen menos veces que el parámetro (utilizado en la clase de ponderación).
- **select_ent**: Booleano.
- **select_suff**: Booleano.
- **select_conn**: Booleano.
- **weighting**: Esquema de ponderación, posibles valores: tfidf, tf y entropy.

[53] Conjunto de elementos seleccionados y separadas del texto original (Tellez, Moctezuma, Miranda y Graff, 2018).
[54] Conjunto de elementos seleccionados y separados por ‘q’ posiciones del texto original (Tellez, Moctezuma, Miranda y Graff, 2018).
[55] Conjunto de palabras seleccionadas y separadas del texto original (Tellez, Moctezuma, Miranda y Graff, 2018).

Capítulo 5 – Conclusiones y Trabajo Futuro

Capítulo 5 – Conclusiones y Trabajo Futuro

5.1 Conclusiones

En la investigación realizada se cumplieron los objetivos planteados logrando crear una herramienta de machine learning capaz de detectar posibles casos de depresión entre los usuarios de Twitter.

El proyecto trabajó dos tipos de pruebas: una por usuario y otra por tweet, en cada una se aplicaron experimentos y se obtuvo igual cantidad de resultados. La cantidad de datos por usuario eran pocos, pero con textos de dimensiones enormes y la cantidad de datos por tweet eran muchos, pero con textos cortos.

Los experimentos realizados mostraron que el modelo SVM/InferSent por usuario con preprocesamiento y el modelo microTC SAP por tweet con preprocesamiento fueron los que mejores métricas obtuvieron:

Por Usuario				
Modelo	**Métrica / Test**	**Resultado**	**Macro**	**Micro**
SVM / Infersent con preprocesamiento	Precision	0.90	0.92	0.91
	Recall	0.99	0.85	0.91
	F1	0.94	0.87	0.91
	Accuracy	0.91	-	-

Por Tweet				
Modelo	**Métrica / Test**	**Resultado**	**Macro**	**Micro**
MicroTC SAP con preprocesa-miento	Precision	0.89	0.87	0.88
	Recall	0.95	0.83	0.88
	F1	0.92	0.85	0.88
	Accuracy	0.88	-	-

Cabe mencionar que los resultados de los modelos restantes son cercanos a los dos antes mencionados, teniendo generalmente la métrica más alta en el recall. Es importante destacar que microTc se comportó de manera robusta en todos los experimentos, arrojando no sólo buenos resultados en las métricas, sino también fue el único modelo que se pudo correr en todas las pruebas, mostrando que es un clasificador confiable y robusto, como se puede observar en la siguiente tabla:

Por Usuario		**Por Tweet**	
Con Preproce-samiento	**Sin Preprocesa-miento**	**Con Preprocesa-miento**	**Sin Preprocesa-miento**
MicroTC SAP	MicroTC SAP	MicroTC SAP	MicroTC SAP
MicroTC SEP	MicroTC SEP	microTC SEP	MicroTC SEP
SVM/FastText	*	SVM/FastText	*
TfIdf	*	*	*
SVM/InferSent	*	*	*
KNN/FastText	*	*	*
KNN/InferSent	*	*	*

Un punto interesante descubierto en los experimentos fue el hecho de que, por usuario, microTc se comportó de mejor manera cuando realizó su propia limpieza sin aplicarle una previa. Y por tweet, microTc se comportó de mejor manera cuando se realizó un preprocesamiento previo al de su propia limpieza. Como aporte final el corpus utilizado, así como los experimentos realizados facilitan nuevos resultados al análisis de predicciones en problemas de salud mental.

5.2 Trabajo Futuro

Algunas propuestas de posibles trabajos futuros para continuar la presente investigación son:

- Realizar el análisis de los datos por tweet con word embedding aplicando InferSent. Lo anterior ya que en el presente proyecto se propuso este análisis, pero la capacidad del clúster y los recursos de cómputo disponibles entre otros factores no dejaron realizar el experimento.
- Se puede mejorar el corpus del proyecto, involucrando a expertos en salud mental al momento de clasificar a los usuarios, ya que en el presente proyecto la clasificación con depresión fue auto declarada por éstos.
- En el actual proyecto se trabajó con un solo corpus el cual estaba enfocado en la clasificación de personas con depresión. Pero existen más desórdenes mentales que aquejan a las personas y que igualmente se pueden analizar con los métodos propuestos en esta investigación.
- Visualizar de manera georreferenciada a los usuarios clasificados con y sin depresión en un mapa, para de esta manera poder identificar las zonas con más densidad de población que cuenten con población afectada por este u otro problema mental, llámense, por ejemplo, zonas rurales, urbanas, nuevos asentamientos, etcétera. El código y el trabajo propuesto en este proyecto será publicado y puesto al alcance del público para que de esta manera ayude a trabajos futuros.

Bibliografía

Agnihotri, D., Verma, K., & Tripathi, P. (2014). Pattern and Cluster Mining on Text Data. *IEEE Xplore*. https://ieeexplore.ieee.org/document/6821432

Chopade, P., Edwards, D., Khan, S., Andrade, A., & Pu, S. (2019). CPSX: Using AI-Machine Learning for Mapping Human-Human Interaction and Measurement of CPS Teamwork Skills. *IEEE Xplore*. https://ieeexplore.ieee.org/abstract/document/9032906

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. (2017). Supervised Learning of Universal Sentence Representations from Natural Langue Inference Data. *ACL Anthology*, *D17* (1070), 670-680. https://aclanthology.org/D17-1070/

Fernández, A. (2019). DE CADA 100 MEXICANOS, 15 PADECEN DEPRESIÓN. *Boletín UNAM-DGCS*, 455. https://www.dgcs.unam.mx/boletin/bdboletin/2019_455.html

Hakdağli, Ö., Özcan, C., & Oğul, I. (2018). Stream text data analysis on twitter using apache spark streaming. *IEEE Xplore*. https://ieeexplore.ieee.org/document/8404540.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition.* Berlin: Springer.

Hogenboom, F., Frasincar, F., Kaymak, U., & De Jong, F. (2011). An overview of event extraction from text. *CEUR Workshop Proceedings*, *779*, 48-57. http://ceur-ws.org/Vol-779/

Hooda, M., Saxena, A., Madhulika, D., & Yadav, B. (2017). A Study and Comparison of Prediction Algorithms for Depression Detection among Millennials: A Machine Learning Approach. *IEEE Xplore*. https://ieeexplore.ieee.org/document/8455078

Hosseinifard, B., Hassan, M., & Rostami, R. (2013). Classifying depression patients and normal subjects using machine learning techniques and nonlinear features from EEG signal. *Computer Methods and Programs in Biomedicine*, *109* (3), 339-345. https://www.sciencedirect.com/science/article/abs/pii/S0169260712002507

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of Tricks for Efficient Text Classification. *ACL Anthology*, *E17* (2068), 427-431. https://aclanthology.org/E17-2068/

Kulasinghe, S., Jayasinghe, A., Rathnayaka, R., Karunarathne, P., Silva, S., & Jayakodi, A. (2019). AI Based Depression and Suicide Prevention System. *IEEE Xplore*. https://ieeexplore.ieee.org/document/9103411

Larsen, M., Boonstra, T., Batterham, P., O'Dea, B., Paris, C., & Christensen, H. (2015). We Feel: Mapping Emotion on Twitter. *IEEE Xplore*. https://ieeexplore.ieee.org/document/7042256

McClellan, C., Ali, M., Mutter, R., Kroutil, L., & Landwehr, J. (2016). Using social media to monitor mental health discussions - evidence from Twitter. *Journal of the American Medical Informatics Association*, 24 (3), 496-502.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel V., Thirion, B., Grisel, O., ... Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *JMLR*, *12* (85), 2825-2830. https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

Pratama, T., & Purwarianti, A. (2017). Topic classification and clustering on Indonesian complaint tweets for bandung government using supervised and unsupervised learning. *IEEE Xplore*. https://ieeexplore.ieee.org/document/8090981

Renuka, J. (2016). Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures. *Exsilio Solutions*. https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

Roman, V. (2019). Algoritmos Naive Bayes: Fundamentos e Implementación. *Medium.* https://medium.com/datos-y-ciencia/algoritmos-naive-bayes-fudamentos-e-implementaci%C3%B3n-4bcb24b307f

Roman, V. (2019). Aprendizaje Supervizado: Introducción a la Clasificación y Principales Algoritmos. *Medium*. https://medium.com/datos-y-ciencia/aprendizaje-supervisado-introducci%C3%B3n-a-la-clasificaci%C3%B3n-y-principales-algoritmos-dadee99c9407

Tekiner, F., Tsuruoka, Y., Tsujii, J., & Ananiadou, S. (2009). Highly scalable Text Mining - parallel tagging application. *IEEE Xplore*. https://ieeexplore.ieee.org/abstract/document/5379432/references#references

Tellez, E., Moctezuma, D., Miranda, S., & Graff, M. (2018). An automated text categorization framework based on hyperparameter optimization. *Knowledge-Based Systems*, *149*, 110-123. https://www.sciencedirect.com/science/article/abs/pii/S0950705118301217

Xiaomeng, Z., Chong, Z., & Zipei, Y. (2020). Emotional Tendency Analysis of Twitter Texts Based on CNNs Model. *IEEE Xplore*. https://ieeexplore.ieee.org/document/9150239

ANEXOS

ANEXO I: Repositorio del código de Python utilizado en este proyecto

Liga:

https://github.com/mlCorpusDepresion/mlPythonDepresionCode

Hipervínculo:

mlCorpusDepresion/mlPythonDepresionCode: This project shows the code used to predict possible cases of depression in Twitter users, a tagged corpus was used. The steps to follow were: preprocessing, cleaning, modeling, training, testing and results. (github.com)

ANEXO II: Códigos individuales de Python utilizados en este proyecto

Código de Python utilizado para el análisis por tweet utilizando microTC (SEP y SAP). Sin limpieza previa.

```
from sklearn.model_selection import train_test_split
import numpy as np
import tarfile
from microtc.utils import tweet_iterator

tar = tarfile.open("usuarios_depresion.json.tar.gz")
tar.extractall()
tar.close()
data = [i for i in tweet_iterator('usuarios_depresion.json')]

ktu,klases,tweets,users = [],[],[],[]
for i in data:
    for j in i['tweets']:
        ktu.append([j['text'],int(i['klass']),i['user']])

for i in ktu:
    tweets.append(i[0])
    klases.append(i[1])
    users.append(i[2])

import pandas as pd
dataE = pd.DataFrame({'Textos':tweets, 'Klases':klases,'Usuarios':users})

# Train Test
XE = dataE
yE = dataE['Klases']
XE_data_train, XE_data_test, yE_train, yE_test = train_test_split(XE, yE,
test_size=0.3)
print('se creo el train y el test')

print(XE.groupby(['Klases']).count())
print(XE_data_train.groupby(['Klases']).count())
print(XE_data_test.groupby(['Klases']).count())

yE_train = list(yE_train)
yE_test = list(yE_test)

XE_train = list(XE_data_train['Textos'])
XE_test = list(XE_data_test['Textos'])

# microTC - micro Text Classification
# Params editados
```

```
from microtc.textmodel import TextModel

textmodelEM = TextModel(docs=None,
text='text',num_option='delete',usr_option='delete',
    url_option='delete',emo_option='group',hashtag_option='delete',
    ent_option='none',lc=True,del_dup=True,del_punc=True,del_diac=True,
    token_list=[[2, 1], -1,3,4],token_min_filter=0,token_max_filter=1,
    select_ent=False,select_suff=False, select_conn=False,
    weighting='tfidf')
textmodelEM.fit(list(XE_train))

from sklearn.svm import LinearSVC
svcEM = LinearSVC()
svcEM.fit(textmodelEM.transform(XE_train), yE_train)

y_predEM = svcEM.predict(textmodelEM.transform(XE_test))
print('Termino microTC params')

# Default
textmodelEMD = TextModel()
textmodelEMD.fit(list(XE_train))

svcEMD = LinearSVC()
svcEMD.fit(textmodelEMD.transform(XE_train), yE_train)

y_predEMD = svcEMD.predict(textmodelEMD.transform(XE_test))
print('Termino microTC default')

from sklearn.metrics import precision_score, recall_score, f1_score,
accuracy_score

def metricas(modelo,y_test,predict):
    print(modelo)
print('Precision:',round(precision_score(y_test,predict,average=None)[0],3),
    'Macro:',round(precision_score(y_test, predict, average='macro'),3),
    'Micro:',round(precision_score(y_test, predict, average='micro'),3))
    print('Recall:',round(recall_score(y_test,predict,average=None)[0],3),
    'Macro:',round(recall_score(y_test, predict, average='macro'),3),
    'Micro:',round(recall_score(y_test, predict, average='micro'),3))
    print('F1:       ',round(f1_score(y_test, predict, average=None)[0],3),
    'Macro:',round(f1_score(y_test, predict, average='macro'),3),
    'Micro:',round(f1_score(y_test, predict, average='micro'),3))
    print('Accuracy: ',round(accuracy_score(y_test, predict),3))

print(metricas('\nmicroTC_Params',yE_test,y_predEM))
print(metricas('\nmicroTC_Default',yE_test,y_predEMD))

from sklearn.metrics import confusion_matrix
print('\nmicroTC\n',confusion_matrix(yE_test, y_predEM))
print('\nmicroTC_Default\n',confusion_matrix(yE_test, y_predEMD))
```

Código de Python utilizado para el análisis por tweet utilizando microTC (SAP) y fasttext con SVM. Utilizando limpieza previa.

```
import fasttext
from InferSent.models import InferSent
import torch
import torchvision
import nltk
nltk.download('punkt')
from sklearn.model_selection import train_test_split
import numpy as np
import math
import tarfile
from microtc.utils import tweet_iterator

tar = tarfile.open("usuarios_depresion.json.tar.gz")
tar.extractall()
tar.close()
data = [i for i in tweet_iterator('usuarios_depresion.json')]

ktu,klases,tweets,users = [],[],[],[]
for i in data:
    for j in i['tweets']:
        ktu.append([j['text'],int(i['klass']),i['user']])

from random import sample

for i in ktu:
    tweets.append(i[0])
    klases.append(i[1])
    users.append(i[2])

import re as reg
import unicodedata
import nltk
nltk.download('stopwords')
import re, string
from nltk.corpus import stopwords
lsw = stopwords.words('spanish')

text2 = tweets.copy()

def QuitarAcentos(cadena):
    formaNFKD = unicodedata.normalize('NFKD', cadena)
    return u"".join([c for c in formaNFKD if not unicodedata.combining(c)])
def RP(text):
    return re.sub('[%s]' % re.escape(string.punctuation), '', text)
def RD(digito):
    return ''.join(i for i in digito if not i.isdigit())
#def deEmojify(inputString):
#    return inputString.encode('ascii', 'ignore').decode('ascii')
```

```
def reducirString(palabra):
    pattern=reg.compile(r"(.)\1{1,}",reg.DOTALL)
    string=pattern.sub(r"\1",palabra)
    return string

susp2,susp3,susp5,TextoL,TextoLimpio = [],[],[],[],[]
QR,stop1,stem1,Tt,KC,Us = [],[],[],[],[],[]

for i in text2:
    susp2.append(QuitarAcentos(i).lower())
for i in susp2:
    susp5.append(RP(i))
for i in susp5:
    TextoL.append(RD(i))
for i in TextoL:
    aux = i.split()
    aux2 = []
    for j in aux:
        aux2.append(reducirString(j))
    QR.append(' '.join(aux2))

for i in QR:
    aux = []
    h = i.split()
    for r in h:
        if r not in lsw: #and r not in lsw2):
            aux.append(r)
    stop1.append(' '.join(aux))

for i in range(len(stop1)):
    if stop1[i] == '':
        continue
    else:
        Tt.append(stop1[i])
        KC.append(klases[i])
        Us.append(users[i])

print('Limpieza realizada')

import pandas as pd
dataE = pd.DataFrame({'Textos':Tt, 'Klases':KC,'Usuarios':Us})

# FastText
ft = fasttext.load_model('crawl-300d-2M-subword.bin')
print('Se cargo el modelo de fasttext')

embeddingsEF = [ ft.get_sentence_vector(v) for v  in list(dataE['Textos'])]
print('Se crearon los vectores de fasttext')

dataE = dataE.assign(fft=embeddingsEF)

# Train Test
```

```
XE = dataE
yE = dataE['Klases']
XE_data_train, XE_data_test, yE_train, yE_test = train_test_split(XE, yE,
test_size=0.3)
print('se creo el train y el test')

print(XE.groupby(['Klases']).count())
print(XE_data_train.groupby(['Klases']).count())
print(XE_data_test.groupby(['Klases']).count())

yE_train = list(yE_train)
yE_test = list(yE_test)

XEF_train = list(XE_data_train['fft'])
XEF_test = list(XE_data_test['fft'])

# Support Vector Machine (SVM)
from sklearn.svm import LinearSVC
svcEF = LinearSVC(random_state=0)
svcEF.fit(XEF_train, yE_train)
# In[34]:
y_predEF0 = svcEF.predict(XEF_test)
print('Termino SVM')

# microTC - micro Text Classification
X_trainEM = list(XE_data_train['Textos'])
X_testEM = list(XE_data_test['Textos'])

from microtc.textmodel import TextModel
textmodelEM = TextModel()
textmodelEM.fit(list(X_trainEM))

from sklearn.svm import LinearSVC
svcEM = LinearSVC()
svcEM.fit(textmodelEM.transform(X_trainEM), yE_train)

y_predEM = svcEM.predict(textmodelEM.transform(X_testEM))
print('Termino microTC')

from sklearn.metrics import precision_score, recall_score, f1_score,
accuracy_score

def metricas(modelo,y_test,predict):
    print(modelo)
print('Precision:',round(precision_score(y_test,predict,average=None)[0],3),
    'Macro:',round(precision_score(y_test, predict, average='macro'),3),
    'Micro:',round(precision_score(y_test, predict, average='micro'),3))
    print('Recall:',round(recall_score(y_test,predict,average=None)[0],3),
    'Macro:',round(recall_score(y_test, predict, average='macro'),3),
    'Micro:',round(recall_score(y_test, predict, average='micro'),3))
    print('F1:       ',round(f1_score(y_test, predict, average=None)[0],3),
    'Macro:',round(f1_score(y_test, predict, average='macro'),3),
```

```
    'Micro:',round(f1_score(y_test, predict, average='micro'),3))
    print('Accuracy: ',round(accuracy_score(y_test, predict),3))

print(metricas('\nFastText_SVM',yE_test,y_predEF0))
print(metricas('\nmicroTC',yE_test,y_predEM))

from sklearn.metrics import confusion_matrix

print('\nFastText_SVM\n',confusion_matrix(yE_test, y_predEF0))
print('\nmicroTC\n',confusion_matrix(yE_test, y_predEM))
```

Código de Python utilizado para el análisis por tweet utilizando microTC (SEP) y fasttext con SVM. Utilizando limpieza previa.

```
import fasttext
from InferSent.models import InferSent
import nltk
nltk.download('punkt')
from sklearn.model_selection import train_test_split
import numpy as np
import math
import tarfile
from microtc.utils import tweet_iterator

tar = tarfile.open("usuarios_depresion.json.tar.gz")
tar.extractall()
tar.close()
data = [i for i in tweet_iterator('usuarios_depresion.json')]

ktu,klases,tweets,users = [],[],[],[]
for i in data:
    for j in i['tweets']:
        ktu.append([j['text'],int(i['klass']),i['user']])

from random import sample

for i in ktu:
    tweets.append(i[0])
    klases.append(i[1])
    users.append(i[2])

import re as reg
import unicodedata
import nltk
nltk.download('stopwords')
import re, string
from nltk.corpus import stopwords
lsw = stopwords.words('spanish')

```

```
text2 = tweets.copy()

def QuitarAcentos(cadena):
    formaNFKD = unicodedata.normalize('NFKD', cadena)
    return u"".join([c for c in formaNFKD if not  unicodedata.combining(c)])

def RP(text):
    return re.sub('[%s]' % re.escape(string.punctuation), '', text)
def RD(digito):
    return ''.join(i for i in digito if not i.isdigit())
def reducirString(palabra):
    pattern=reg.compile(r"(.)\1{1,}",reg.DOTALL)
    string=pattern.sub(r"\1",palabra)
    return string

susp2,susp3,susp5,TextoL,TextoLimpio = [],[],[],[],[]
QR,stop1,stem1,Tt,KC,Us = [],[],[],[],[],[]

for i in text2:
    susp2.append(QuitarAcentos(i).lower())
for i in susp2:
    susp5.append(RP(i))
for i in susp5:
    TextoL.append(RD(i))
for i in TextoL:
    aux = i.split()
    aux2 = []
    for j in aux:
        aux2.append(reducirString(j))
    QR.append(' '.join(aux2))

for i in QR:
    aux = []
    h = i.split()
    for r in h:
        if r not in lsw: #and r not in lsw2):
            aux.append(r)
    stop1.append(' '.join(aux))

for i in range(len(stop1)):
    if stop1[i] == '':
        continue
    else:
        Tt.append(stop1[i])
        KC.append(klases[i])
        Us.append(users[i])

print('Limpieza realizada')

import pandas as pd
dataE = pd.DataFrame({'Textos':Tt, 'Klases':KC,'Usuarios':Us})

```

```
# FastText
ft = fasttext.load_model('crawl-300d-2M-subword.bin')
print('Se cargo el modelo de fasttext')

embeddingsEF = [ ft.get_sentence_vector(v) for v  in list(dataE['Textos'])]
print('Se crearon los vectores de fasttext')

dataE = dataE.assign(fft=embeddingsEF)

# Train Test
XE = dataE
yE = dataE['Klases']
XE_data_train, XE_data_test, yE_train, yE_test = train_test_split(XE, yE,
test_size=0.3)
print('se creo el train y el test')

print(XE.groupby(['Klases']).count())
print(XE_data_train.groupby(['Klases']).count())
print(XE_data_test.groupby(['Klases']).count())

yE_train = list(yE_train)
yE_test = list(yE_test)

XEF_train = list(XE_data_train['fft'])
XEF_test = list(XE_data_test['fft'])

# Support Vector Machine (SVM)
from sklearn.svm import LinearSVC
svcEF = LinearSVC(random_state=0)
svcEF.fit(XEF_train, yE_train)

y_predEF0 = svcEF.predict(XEF_test)
print('Termino SVM')

# microTC - micro Text Classification
X_trainEM = list(XE_data_train['Textos'])
X_testEM = list(XE_data_test['Textos'])

from microtc.textmodel import TextModel
textmodelEM = TextModel(docs=None,
text='text',num_option='delete',usr_option='delete',
    url_option='delete',emo_option='group',hashtag_option='delete',
    ent_option='none',lc=True,del_dup=True,del_punc=True,del_diac=True,
    token_list=[[2, 1], -1,3,4],token_min_filter=0,token_max_filter=1,
    select_ent=False,select_suff=False, select_conn=False,
    weighting='tfidf')
textmodelEM.fit(list(X_trainEM))

from sklearn.svm import LinearSVC
svcEM = LinearSVC()
svcEM.fit(textmodelEM.transform(X_trainEM), yE_train)

```

```
y_predEM = svcEM.predict(textmodelEM.transform(X_testEM))
print('Termino microTC')

from sklearn.metrics import precision_score, recall_score, f1_score,
accuracy_score

def metricas(modelo,y_test,predict):
    print(modelo)
print('Precision:',round(precision_score(y_test,predict,average=None)[0],3),
    'Macro:',round(precision_score(y_test, predict, average='macro'),3),
    'Micro:',round(precision_score(y_test, predict, average='micro'),3))
    print('Recall:',round(recall_score(y_test,predict,average=None)[0],3),
    'Macro:',round(recall_score(y_test, predict, average='macro'),3),
    'Micro:',round(recall_score(y_test, predict, average='micro'),3))
    print('F1:       ',round(f1_score(y_test, predict, average=None)[0],3),
    'Macro:',round(f1_score(y_test, predict, average='macro'),3),
    'Micro:',round(f1_score(y_test, predict, average='micro'),3))
    print('Accuracy: ',round(accuracy_score(y_test, predict),3))

print(metricas('\nFastText_SVM',yE_test,y_predEF0))
print(metricas('\nmicroTC',yE_test,y_predEM))

from sklearn.metrics import confusion_matrix

print('\nFastText_SVM\n',confusion_matrix(yE_test, y_predEF0))
print('\nmicroTC\n',confusion_matrix(yE_test, y_predEM))
```

Código de Python utilizado para el análisis por usuarios utilizando microTC (SEP) y Tfldf. Utilizando limpieza previa.

```
import tarfile
tar = tarfile.open("usuarios_depresion.json.tar.gz")
tar.extractall()
tar.close()

from microtc.utils import tweet_iterator
data = [i for i in tweet_iterator('usuarios_depresion.json')]

# Extraer los tweets
data[0]['tweets'][4099]['text']

# Extraer lo que se requiere, tres listas de dimension 317 en donde todos
los tweets de cada usuario se juntaron para hacer solo un tweet unico por
usuario.
klases,users,t,tweets = [],[],[],[]
for i in data:
    t = []
    klases.append(int(i['klass']))
    users.append(i['user'])
```

```
    for j in i['tweets']:
        t.append(j['text'])
    tweets.append(' '.join(t))

print('Data Original')
print(len(klases))
print(len(tweets))
print(len(users))


# Limpieza
import re as reg
import unicodedata
import nltk
import re, string

textos = tweets.copy()

def QuitarAcentos(cadena):
    formaNFKD = unicodedata.normalize('NFKD', cadena)
    return u"".join([c for c in formaNFKD if not unicodedata.combining(c)])
def RP(text):
    return re.sub('[%s]' % re.escape(string.punctuation), '', text)
def RD(digito):
    return ''.join(i for i in digito if not i.isdigit())
def reducirString(palabra):
    pattern=reg.compile(r"(.)\1{1,}",reg.DOTALL)
    string=pattern.sub(r"\1",palabra)
    return string

l1,l2,l3,l4,l5 = [],[],[],[],[]

for i in textos:
    l1.append(QuitarAcentos(i).lower())
for i in l1:
    aux = []
    h = i.split()
    for r in h:
        if (r.startswith('http') or r.startswith('rt') or
r.startswith('"http')):
            continue
        aux.append(r)
    l2.append(' '.join(aux))
for i in l2:
    l3.append(RP(i))
for i in l3:
    l4.append(RD(i))
for i in l4:
    aux = i.split()
    aux2 = []
    for j in aux:
        aux2.append(reducirString(j))
```

```
    l5.append(' '.join(aux2))

nltk.download('stopwords')
from nltk.corpus import stopwords
lsw = stopwords.words('spanish')

TweetsLimpios = []

for i in l5:
    aux = []
    h = i.split()
    for r in h:
        if r not in lsw:
            aux.append(r)
    TweetsLimpios.append(' '.join(aux))

print('Data Trabajada')
print(len(users))
print(len(klases))
print(len(TweetsLimpios))

import numpy as np
print('Limpieza realizada')
print(np.unique(klases, return_counts=True))

import pandas as pd
dataE = pd.DataFrame({'Textos':TweetsLimpios,
'Klases':klases,'Usuarios':users})

# Particion 70/30
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(dataE, dataE['Klases'],
test_size=0.30)
print('Particion realizada')

print(dataE.groupby(['Klases']).count())
print(X_train.groupby(['Klases']).count())
print(X_test.groupby(['Klases']).count())

y_train = list(y_train)
y_test = list(y_test)

# TfIdf
XTM_train = list(X_train['Textos'])
XTM_test = list(X_test['Textos'])

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB

#TfidfVectorizer - entrenamiento
vec = TfidfVectorizer()
```

```
X = vec.fit_transform(XTM_train)
mnb = MultinomialNB()
mnb.fit(X, y_train)

#Tfidf - prueba
XTFIDF = vec.transform(XTM_test)
mnb_ = mnb.predict(XTFIDF)
print('TfIdf realizado')


# microTC - TextModel
# TextModel - entrenamiento
from microtc.textmodel import TextModel
from sklearn.svm import LinearSVC

textmodel = TextModel(docs=None,
text='text',num_option='delete',usr_option='delete',
    url_option='delete',emo_option='group',hashtag_option='delete',
   ent_option='none',lc=True,del_dup=True,del_punc=True,del_diac=True,
    token_list=[[2, 1], -1,3,4],token_min_filter=0,token_max_filter=1,
    select_ent=False,select_suff=False, select_conn=False,
    weighting='tfidf').fit(XTM_train)

lsvc = LinearSVC().fit(textmodel.transform(XTM_train), y_train)

#TextModel - prueba
lsvc_ = lsvc.predict(textmodel.transform(XTM_test))
print('microTC realizado')


# Metricas
from sklearn.metrics import  f1_score, precision_score, recall_score,
accuracy_score

def metricas(modelo,y_test,predict):
    print(modelo)
print('Precision:',round(precision_score(y_test,predict,average=None)[0],3),
    'Macro:',round(precision_score(y_test, predict, average='macro'),3),
    'Micro:',round(precision_score(y_test, predict, average='micro'),3))
    print('Recall:',round(recall_score(y_test,predict,average=None)[0],3),
    'Macro:',round(recall_score(y_test, predict, average='macro'),3),
    'Micro:',round(recall_score(y_test, predict, average='micro'),3))
    print('F1:       ',round(f1_score(y_test, predict, average=None)[0],3),
    'Macro:',round(f1_score(y_test, predict, average='macro'),3),
    'Micro:',round(f1_score(y_test, predict, average='micro'),3))
    print('Accuracy: ',round(accuracy_score(y_test, predict),3))

print(metricas('\ntfidf',y_test,mnb_))
   print(metricas('\nmicroTC',y_test,lsvc_))

   # Confusion Matrix
   from sklearn.metrics import confusion_matrix
```

```
print('\ntfidf\n',confusion_matrix(y_test, mnb_))
print('\nmicroTC\n',confusion_matrix(y_test, lsvc_))
```

Código de Python utilizado para el análisis por usuarios utilizando microTC (SAP) y Fasttext e Infersent con SVM, Vecinos más cercanos y Tfldf. Utilizando limpieza previa.

```
import tarfile
tar = tarfile.open("usuarios_depresion.json.tar.gz")
tar.extractall()
tar.close()

from microtc.utils import tweet_iterator
data = [i for i in tweet_iterator('usuarios_depresion.json')]

# Extraer los tweets
data[0]['tweets'][4099]['text']

# Extraer lo que se requiere, tres listas de dimension 317 en donde todos
los tweets de cada usuario se juntaron para hacer solo un tweet unico por
usuario.
klases,users,t,tweets = [],[],[],[]
for i in data:
    t = []
    klases.append(int(i['klass']))
    users.append(i['user'])
    for j in i['tweets']:
        t.append(j['text'])
    tweets.append(' '.join(t))

print('Data Original')
print(len(klases))
print(len(tweets))
print(len(users))


# Limpieza
import re as reg
import unicodedata
import nltk
import re, string

textos = tweets.copy()

def QuitarAcentos(cadena):
    formaNFKD = unicodedata.normalize('NFKD', cadena)
    return u"".join([c for c in formaNFKD if not unicodedata.combining(c)])
def RP(text):
```

```
    return re.sub('[%s]' % re.escape(string.punctuation), '', text)
def RD(digito):
    return ''.join(i for i in digito if not i.isdigit())
def reducirString(palabra):
    pattern=reg.compile(r"(.)\1{1,}",reg.DOTALL)
    string=pattern.sub(r"\1",palabra)
    return string

l1,l2,l3,l4,l5 = [],[],[],[],[]

for i in textos:
    l1.append(QuitarAcentos(i).lower())
for i in l1:
    aux = []
    h = i.split()
    for r in h:
        if (r.startswith('http') or r.startswith('rt') or
r.startswith('"http')):
            continue
        aux.append(r)
    l2.append(' '.join(aux))
for i in l2:
    l3.append(RP(i))
for i in l3:
    l4.append(RD(i))
for i in l4:
    aux = i.split()
    aux2 = []
    for j in aux:
        aux2.append(reducirString(j))
    l5.append(' '.join(aux2))

nltk.download('stopwords')
from nltk.corpus import stopwords
lsw = stopwords.words('spanish')

TweetsLimpios = []

for i in l5:
    aux = []
    h = i.split()
    for r in h:
        if r not in lsw:
            aux.append(r)
    TweetsLimpios.append(' '.join(aux))

print('Data Trabajada')
print(len(users))
print(len(klases))
print(len(TweetsLimpios))

import numpy as np
```

```
print('Limpieza realizada')
print(np.unique(klases, return_counts=True))


# Tabla en pandas con la data
import pandas as pd
dataE = pd.DataFrame({'Textos':TweetsLimpios,
'Klases':klases,'Usuarios':users})


# Modelo FastText
import fasttext
ft = fasttext.load_model('crawl-300d-2M-subword.bin')#, encoding="latin1")
print('Se cargo el modelo de fasttext')

embeddingsEF = [ ft.get_sentence_vector(v) for v  in list(dataE['Textos'])]
print('Se crearon los vectores de fasttext')

dataE = dataE.assign(fft=embeddingsEF)

# Modelo InferSent
from InferSent.models import InferSent
import torch

def InferSent_model(model_version=1):
    MODEL_PATH = 'infersent1.pkl'
    params_model = {'bsize': 64, 'word_emb_dim': 300, 'enc_lstm_dim': 2048,
        'pool_type': 'max', 'dpout_model': 0.0, 'version': model_version}
    # If infersent1 -> use GloVe embeddings. If infersent2 -> use InferSent
embeddings.
    if model_version == 1:
        W2V_PATH = 'glove.840B.300d.txt'
    else:
        W2V_PATH = 'crawl-300d-2M.vec'
    print("loading model ...  {}".format(W2V_PATH))
    model = InferSent(params_model)
    model.load_state_dict(torch.load(MODEL_PATH))
    model.set_w2v_path(W2V_PATH)

    model.build_vocab_k_words(K=3000000)
    return model

import nltk
nltk.download('punkt')
model = InferSent_model(model_version=1)
sent_detector = nltk.data.load('tokenizers/punkt/english.pickle')
print('Se cargo el modelo de infersent')

embeddingsEI = [ model.encode(sent_detector.tokenize(v), bsize=128,
    tokenize=False, verbose=True) for v  in list(dataE['Textos'])]
print('Se crearon los vectores de infersent')

```

```
dataE = dataE.assign(infer = [i[0] for i in embeddingsEI] )

# Particion 70/30
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(dataE, dataE['Klases'],
test_size=0.30)
print('Particion realizada')

print(dataE.groupby(['Klases']).count())
print(X_train.groupby(['Klases']).count())
print(X_test.groupby(['Klases']).count())

# Vectores fasttext e infersent
y_train = list(y_train)
y_test = list(y_test)

XF_train = list(X_train['fft'])
XF_test = list(X_test['fft'])

XI_train = list(X_train['infer'])
XI_test = list(X_test['infer'])

# SVM - fasttext/infersent
# Support Vector Machine (SVM)
from sklearn.svm import LinearSVC
# entrenamiento
svcF = LinearSVC(random_state=0)
svcI = LinearSVC(random_state=0)
svcF.fit(XF_train, y_train)
svcI.fit(XI_train, y_train)
# prueba
svcF_ = svcF.predict(XF_test)
svcI_ = svcI.predict(XI_test)
print('SVM de fasttext e infersent realizado')

# K-Vecinos - fasttext/infersent
# K-Vecinos Cercanos
from sklearn.neighbors import KNeighborsClassifier
# entrenamiento
knnF = KNeighborsClassifier(n_neighbors=5)
knnI = KNeighborsClassifier(n_neighbors=5)
knnF.fit(XF_train, y_train)
knnI.fit(XI_train, y_train)
# prueba
knnF_ = knnF.predict(XF_test)
knnI_ = knnI.predict(XI_test)
print('KNN de fasttext e infersent realizado')


# TfIdf
XTM_train = list(X_train['Textos'])
```

```
XTM_test = list(X_test['Textos'])

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB

# TfidfVectorizer - entrenamiento
vec = TfidfVectorizer()
X = vec.fit_transform(XTM_train)
mnb = MultinomialNB()
mnb.fit(X, y_train)

# Tfidf - prueba
XTFIDF = vec.transform(XTM_test)
mnb_ = mnb.predict(XTFIDF)
print('TfIdf realizado')


# microTC - TextModel
# TextModel - entrenamiento
from microtc.textmodel import TextModel
from sklearn.svm import LinearSVC

textmodel = TextModel().fit(XTM_train)
lsvc = LinearSVC().fit(textmodel.transform(XTM_train), y_train)

# TextModel - prueba
lsvc_ = lsvc.predict(textmodel.transform(XTM_test))
print('microTC realizado')


# Metricas
from sklearn.metrics import  f1_score, precision_score, recall_score,
accuracy_score

def metricas(modelo,y_test,predict):
    print(modelo)
print('Precision:',round(precision_score(y_test,predict,average=None)[0],3),
    'Macro:',round(precision_score(y_test, predict, average='macro'),3),
    'Micro:',round(precision_score(y_test, predict, average='micro'),3))
    print('Recall:',round(recall_score(y_test,predict,average=None)[0],3),
    'Macro:',round(recall_score(y_test, predict, average='macro'),3),
    'Micro:',round(recall_score(y_test, predict, average='micro'),3))
    print('F1:       ',round(f1_score(y_test, predict, average=None)[0],3),
    'Macro:',round(f1_score(y_test, predict, average='macro'),3),
    'Micro:',round(f1_score(y_test, predict, average='micro'),3))
    print('Accuracy: ',round(accuracy_score(y_test, predict),3))

print(metricas('\ntfidf',y_test,mnb_))
print(metricas('\nmicroTC',y_test,lsvc_))
print(metricas('\nSVC_FastText',y_test,svcF_))
print(metricas('\nSVC_InferSent',y_test,svcI_))
print(metricas('\nKNN_FastText',y_test,knnF_))
```

```
print(metricas('\nKNN_InferSent',y_test,knnI_))


# Confusion Matrix
from sklearn.metrics import confusion_matrix
   print('\ntfidf\n',confusion_matrix(y_test, mnb_))
   print('\nmicroTC\n',confusion_matrix(y_test, lsvc_))
   print('\nSVC_FastText\n',confusion_matrix(y_test, svcF_))
   print('\nSVC_InferSent\n',confusion_matrix(y_test, svcI_))
   print('\nKNN_FastText\n',confusion_matrix(y_test, knnF_))
   print('\nKNN_InferSent\n',confusion_matrix(y_test, knnI_))
```

Código de Python utilizado para el análisis por usuarios utilizando microTC (SAP y SEP). Sin limpieza previa.

```
import tarfile
tar = tarfile.open("usuarios_depresion.json.tar.gz")
tar.extractall()
tar.close()

from microtc.utils import tweet_iterator
data = [i for i in tweet_iterator('usuarios_depresion.json')]

# Extraer lo que se requiere, tres listas de dimension 317 en donde todos
los tweets de cada usuario se juntaron para hacer solo un tweet unico por
usuario.
klases,users,t,tweets = [],[],[],[]
for i in data:
    t = []
    klases.append(int(i['klass']))
    users.append(i['user'])
    for j in i['tweets']:
        t.append(j['text'])
    tweets.append(' '.join(t))

print('Data Original')
print(len(klases))
print(len(tweets))
print(len(users))

import numpy as np
print(np.unique(klases, return_counts=True))

# Tabla en pandas con la data
import pandas as pd
dataE = pd.DataFrame({'Textos':tweets, 'Klases':klases,'Usuarios':users})

# Particion 70/30
from sklearn.model_selection import train_test_split
```

```
35
36 X_train, X_test, y_train, y_test = train_test_split(dataE, dataE['Klases'],
37 test_size=0.30)
38 print('Particion realizada')
39
40 print(dataE.groupby(['Klases']).count())
41 print(X_train.groupby(['Klases']).count())
42 print(X_test.groupby(['Klases']).count())
43
44 y_train = list(y_train)
45 y_test = list(y_test)
46 XM_train = list(X_train['Textos'])
47 XM_test = list(X_test['Textos'])
48
49 # microTC - TextModel
50 # Params editados
51 # TextModel - entrenamiento
52 from microtc.textmodel import TextModel
53 from sklearn.svm import LinearSVC
54
55 textmodel = TextModel(docs=None,
56 text='text',num_option='delete',usr_option='delete',
57     url_option='delete',emo_option='group',hashtag_option='delete',
58     ent_option='none',lc=True,del_dup=True,del_punc=True,del_diac=True,
59     token_list=[[2, 1], -1,3,4],token_min_filter=0,token_max_filter=1,
60     select_ent=False,select_suff=False, select_conn=False,
61     weighting='tfidf').fit(XM_train)
62
63 lsvc = LinearSVC().fit(textmodel.transform(XM_train), y_train)
64
65 # TextModel - prueba
66 lsvc_ = lsvc.predict(textmodel.transform(XM_test))
67 print('microTC params realizado')
68
69 # Params default
70 # TextModel - entrenamiento
71 textmodelD = TextModel().fit(XM_train)
72
73 lsvcD = LinearSVC().fit(textmodelD.transform(XM_train), y_train)
74
75 # TextModel - prueba
76 lsvcD_ = lsvcD.predict(textmodelD.transform(XM_test))
77 print('microTC default realizado')
78
79
80 # Metricas
81 from sklearn.metrics import  f1_score, precision_score, recall_score,
82 accuracy_score
83
84 def metricas(modelo,y_test,predict):
85     print(modelo)
86 print('Precision:',round(precision_score(y_test,predict,average=None)[0],3),
```

```
    'Macro:',round(precision_score(y_test, predict, average='macro'),3),
    'Micro:',round(precision_score(y_test, predict, average='micro'),3))
    print('Recall:',round(recall_score(y_test,predict,average=None)[0],3),
    'Macro:',round(recall_score(y_test, predict, average='macro'),3),
    'Micro:',round(recall_score(y_test, predict, average='micro'),3))
    print('F1:       ',round(f1_score(y_test, predict, average=None)[0],3),
    'Macro:',round(f1_score(y_test, predict, average='macro'),3),
    'Micro:',round(f1_score(y_test, predict, average='micro'),3))
    print('Accuracy: ',round(accuracy_score(y_test, predict),3))

print(metricas('\nmicroTC_params',y_test,lsvc_))
print(metricas('\nmicroTC_default',y_test,lsvcD_))

# Confusion Matrix
from sklearn.metrics import confusion_matrix
print('\nmicroTC_params\n',confusion_matrix(y_test, lsvc_))
print('\nmicroTC_default\n',confusion_matrix(y_test, lsvcD_))
```