# UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

## FACULTAD 3


# MÉTODO PARA EL ANÁLISIS DE SENTIMIENTOS EN LA RED DE MENSAJERÍA INSTANTÁNEA DE LA UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE
**INGENIERO EN CIENCIAS INFORMÁTICAS**

**AUTOR:** LETICIA SARDUY MARTÍNEZ
**TUTORES:** ING. VLADIMIR MILIÁN NUÑEZ
LIC. RAYNEL BATISTA TELLEZ

**LA HABANA, SEPTIEMBRE DE 2020**

# Agradecimientos

Al culminar la presente investigación deseo expresar mi agradecimiento a todas aquellas personas que han hecho posible el desarrollo de la misma, mis más sinceros sentimientos de gratitud hacia:

Mis tutores Vladimir Milián Núñes y Raynel Batista Téllez por el asesoramiento y profesionalidad en la materialización de esta investigación.

Mis profesores que he tenido en el transcurso de mi carrera por permitirme aprender de ellos y me ayudaron de una forma u otra en mi formación como profesional.

Mis compañeros y amigos por los buenos y malos momentos que compartimos en estos cinco años de etapa estudiantil.

Mis padres por apoyarme incondicionalmente y animarme a continuar hasta terminar, por guiarme en mi camino, comprensión brindado a lo largo de mi vida.

Mis abuelos que los tengo en mi corazón, aunque no estén físicamente, por hacerme una mejor persona con sus sabios consejos, por su amor y cariño.

# Dedicatoria

*Con todo mi amor y cariño a toda mi familia por todo lo que representan para mí, en especial a mis padres.*

# Declaración Jurada de Autoría

Declaro por este medio que yo **Leticia Sarduy Martínez**, con carné de identidad **97031812574**, soy el autor principal del trabajo de diploma **Método para el análisis de sentimientos en la red de mensajería instantánea de la Universidad de las Ciencias Informáticas**, y que autorizo a la **Universidad de las Ciencias Informáticas** a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, firmo la presente declaración jurada de autoría en La Habana a los ____ días del mes de _____ del año _____.

**Leticia Sarduy Martínez**

**Ing. Vladimir Milián Nuñez**

**Lic. Raynel Batista Tellez**

# Datos de Contacto

Tutor: Ing. Vladimir Milián Núñez, Ingeniero en Ciencias Informáticas, de la Universidad de las Ciencias Informáticas, 2008. Máster en Ciencias en la Universidad de las Ciencias Informáticas, 2018, Cuba. Profesor Asistente de las disciplinas Física y Matemática Aplicada de la Universidad de las Ciencias Informáticas. Miembro del grupo de Investigación de Inteligencia Artificial y Reconocimiento de Patrones desde el 2008. Líder de proyecto y líder técnico (Arquitecto de software principal) de varios proyectos del centro de desarrollo de Telemática de la UCI en el periodo entre 2008 – 2011. Miembro de la Asociación Cubana de Reconocimiento de Patrones (ACRP) desde el 2013 y presidente en funciones de la delegación de base de la UCI. Editor Asociado de la Revista Cubana de Ciencias Informáticas. Miembro del comité organizador y del comité científico de varios eventos. Autor y co-autor de varias publicaciones en revistas y memorias de eventos. Profesor de un curso de postgrado. Tutor de 24 Tesis de pre-grado para Ingeniería. Correo electrónico: vmilian@uci.cu Researchgate: Vladimir Milián Núñez Twitter: @vmiliann

Tutor: Lic. Raynel Batista Téllez, Sociólogo, editor, Coordinador General de Ediciones Futuro, sello editorial de la Universidad de las Ciencias Informáticas. Profesor Auxiliar. Ha impulsado proyectos interdisciplinares que vinculan las ciencias sociales y el cálculo computacional, así como las humanidades digitales, en colaboración con actores de América Latina, Europa y Asia. Ha dirigido varios proyectos en temas como sociocibernética, interacción hombre-computadora y social media. Posee varias publicaciones, participaciones en eventos, proyectos, distinciones y premios. Ha asesorado trabajos de diploma, tesis de maestría y doctorado. Es miembro de la Junta Nacional Editorial del Ministerio de Cultura, Organización Internacional de Sociología, Sociedad Cubana de Derecho e Informática, Asociación Cubana de Comunicadores Sociales, Organización Internacional de Editores, Junta Editorial de la Revista Cubana de Ciencias Informáticas. Ha mostrado interés por los nuevos entornos de lectura, las redes académicas de interacción social y la ciencia de los datos. Sus principales resultados se enmarcan en la sociología de la innovación y la antropología digital, con especial interés en la curación de contenidos digitales. Correo electrónico: rainer@uci.cu Researchgate: Raynel Batista Téllez Twitter: @RBatell

# Resumen

El análisis de sentimientos es un área de investigación que estudia el tratamiento computacional de opiniones, sentimientos y subjetividad en textos. En este contexto, una opinión es una valoración positiva o negativa hacia un producto, servicio, organización, persona que se expresa en un determinado documento. Con el crecimiento explosivo de las redes sociales las personas y las organizaciones utilizan cada vez más las opiniones públicas en estos medios para tomar decisiones. Sin embargo, encontrar y monitorear sitios de opinión y extraer la información contenida en ellos sigue siendo una tarea formidable debido a la proliferación de diversos sitios ricos en opiniones.

Un ejemplo de ello es la red institucional de mensajería instantánea de la Universidad de las Ciencias Informáticas que genera un gran volumen de texto que imposibilita identificar opiniones y sentimientos en los mensajes por el lector humano. El presente trabajo es el resultado de una investigación acerca del desarrollo de un método que permite determinar la polaridad del sentimiento en los mensajes generados en la red.

En la investigación se realizó un estudio sobre los fundamentos teóricos del Análisis de Sentimientos y su relación con la Minería de Datos, el Procesamiento del Lenguaje Natural y el Aprendizaje Automático. Además se exponen los métodos más importantes para el desarrollo de estas soluciones implementando un clasificador de sentimientos basado en algoritmos de aprendizaje supervisados. Como resultado final se obtuvo un método que facilitó identificar los sentimientos generados por los mensajes de forma automática.

**Palabras clave:** análisis de sentimientos, aprendizaje automatizado supervisado, procesamiento del lenguaje natural, clasificación de documentos.

# Abstract

Sentiment analysis is an area of research that studies the computational treatment of opinions, feelings and subjectivity in texts. In this context, an opinion is a positive or negative assessment of a product, service, organization, person that is expressed in a certain document. With the explosive growth of social networks, people and organizations increasingly use public opinions in these media to make decisions. However, finding and monitoring opinion sites and extracting the information contained in them remains a formidable task due to the proliferation of various opinion-rich sites.

An example of this is the institutional instant messaging network of the University of Computer Sciences which generates a large volume of text that makes it impossible for the human reader to identify opinions and feelings in the messages. The present work is the result of an investigation about the development of a method that allows determining the polarity of the feeling in the messages generated in the network.

In the research, a study was carried out on the theoretical foundations of Sentiment Analysis and its relationship with Data Mining, Natural Language Processing and Machine Learning. In addition, the most important methods for the development of these solutions are exposed, implementing a sentiment classifier based on supervised learning algorithms. As a final result, a method was obtained that made it easier to identify the feelings generated by the messages automatically.

**Keywords:** sentiment analysis, supervised machine learning, natural language processing, document classification.

# ÍNDICE

# ÍNDICE DE TABLAS

# ÍNDICE DE FIGURAS

# Introducción

Las personas a la hora de tomar una decisión siempre tienen en cuenta, además de su propia experiencia, lo que otras personas puedan aportar a su selección, por lo que saber lo que piensan los demás es una parte importante en la vida de los seres humanos. El pedir ayuda, consejo u opinión no es más que un recurso que permite al ser humano ampliar su conocimiento sobre determinado tema con el objetivo de minimizar el riesgo que supone el tomar una mala decisión (Sobrino Sande, 2018).

Con la llegada del Internet y de la Web 2.0 [1]se hace un cambio en la forma en que las personas buscan opiniones sin dejar de un lado las formas tradicionales (el sistema boca-a-boca[2], el conocimiento y experiencia de familiares y amigos y las publicaciones en papel) se han convertido en fuente generadora la cantidad sentimientos, anhelos y frustraciones expresada sobre cualquier tema y en cualquier momento.

Una fuente fundamental dentro de este contexto son las redes de comunicación instantánea, donde la habilidad de extraer información de estos es una práctica que ya están adoptando muchas organizaciones a nivel mundial. Se ha probado que los cambios en el sentimiento de las redes sociales se corresponden a cambios, hechos o eventos que ocurren en las comunidades y que afectan a uno o varios de sus miembros (Díaz Lazo et al., 2011).

Los sistemas de mensajería instantánea permiten la comunicación en tiempo real entre dos o más personas, basada en mensajes de texto, a través de dispositivos conectados a una red, lo cual ha facilitado la comunicación entre las personas sin importar la distancia que exista entre ellas. Sin embargo, la mensajería instantánea es un arma de doble filo y podría ser mal utilizada para el intercambio de información ilegítima, influir en el comportamiento y emociones de las personas o incentivar/cometer actos delictivos. Es por ello, que determinar el tono emocional que hay detrás de una serie de palabras, es decir, intentar entender las actitudes, opiniones y emociones expresadas en las conversaciones entre los usuarios de los sistemas de mensajería instantánea con sus contactos ha sido objeto de atención por parte de la comunidad de investigadores y administradores de este tipo de sistemas (men, 2020).

Es muy interesante tener en cuenta que los usuarios de las redes de mensajería instantánea, suelen escribir a sus contactos cuando presentan o enfrentan algún problema o situación desfavorable, es decir, para quejarse al respecto de un servicio, producto, evento o persona como forma de desahogo, y por lo general, tienden a compartirlo con más de un contacto, y estos a su vez con sus contactos. Por el contrario, muy rara vez se utiliza para compartir un criterio positivo, ya que se tiende a tener la idea de

[1]Concepto atribuido a Tim O'Reilly y nombrado en la conferencia sobre la Web 2.0 de O'Reilly Media en 2004. El termino establece una primera época de Internet rn donde la comunicación era unidireccional, ya que los usuarios eran objetos pasivos que recibirían la información publicada por los administradores de los sitios web sin posibilidad de interinar. En la Web aparecen los blogs, foros debate, redes sociales, sitios de recomendación y las wikis dando lugar a una comunicación bidireccional en donde todos los actores de la red aportan contenido por igual.

[2]Sistema usado para transmitir opiniones, juicios y expresar virtudes p defectos de productos o servicio

que eso es lo que se debe hacer. Esto significa, que este tipo de red es una fuente que permite medir el estado de opinión de la comunidad universitaria.

En los sistemas de mensajería instantánea de redes institucionales, es extremadamente útil la monitorización de las conversaciones de los usuarios ya que permite obtener una idea de la opinión pública general sobre ciertos temas, y cobra mayor importancia aún, sobre todo en redes institucionales de centros educativos. En este caso se encuentra la Universidad de las Ciencias Informáticas (UCI), donde existe una red de mensajería instantánea (comúnmente conocida como Jabber).

Poder conocer en todo momento qué es lo que piensan los usuarios ofrece una ventaja competitiva a la hora de poder mejorar los aspectos menos populares sobre algunos de tus productos o servicios, conocer si las palabras o acciones de un usuario son bien recibidas o no por sus contactos es un instrumento de incuestionable utilidad. Es por esto, que para determinar o descubrir cuál es el tono emocional de las conversaciones en la red de mensajería de la universidad, es necesario aplicar técnicas de procesamiento de lenguaje natural. Según Mejova (2012) el análisis de sentimiento tiene por objetivo extraer opiniones y emociones de textos, clasificando las emociones expresadas en estos textos a lo largo de un espectro de polaridad *positiva – neutral – negativa*.

El volumen de datos que diariamente es generado en la red de mensajería instantánea de la UCI, son analizados con herramientas que no distinguen entre la información relevante y no relevante. En este proceso se invierte demasiado tiempo en el análisis de dicha información, lo que trae consigo la perdida de la oportunidad de prevenir actividades ilegitimas en función de los reglamentos internos de la institución. Además, dichas herramientas limitan la obtención e identificación de las opiniones para determinar el tono emocional (polaridad) de las conversaciones entre los usuarios. Por otra parte, la carencia de personal para atender esta tarea imposibilita que no se pueda analizar toda la información, el carácter subjetivo (falta de precisión) del análisis realizado, hace que los resultados conseguidos sean inexactos e incompletos, y se vea afectada la fidelidad y el alcance de los mismos. A partir del análisis anterior es posible afirmar que los resultados obtenidos son ineficaces a pesar de los altos costos en tiempo y recursos destinados.

Una vez conocida la importancia que tiene esta información de opiniones y sentimientos generados a cada instante en el Jabber se hace necesario poder trabajar con ella sin perder datos e información relevantes. Procesar tal cantidad de datos se requieren de nuevas tecnologías capaces de analizar y clasificar de manera automática la polaridad del sentimiento descrita por los usuarios en sus opiniones en la red y obtener una medida del sentir general, surge así el siguiente **problema a resolver**: el modo en que se analizan los datos en la mensajería instantánea en la UCI limita la identificación oportuna y rápida del sentimiento general que las personas expresan en la red a partir de la detección de la polaridad en los mensajes de textos.

Teniendo como **objeto de estudio**: el análisis de sentimiento en redes de mensajería instantánea, específicamente en el **campo de acción**: detección de polaridad en los mensajes de textos de la red de mensajería instantánea de la UCI (Jabber).

Para dar solución al problema planteado se define como **objetivo general**: desarrollar un método que permita detectar el tono emocional en las conversaciones de la red de mensajería instantánea de la UCI (Jabber), con el objetivo de detectar estados de opiniones y facilitar la toma de decisiones.

Por lo descrito anteriormente la **idea a defender** es: si se desarrolla un método que determine la

polaridad en los mensajes publicados en la red de mensajería instantánea de la UCI contribuiría a la detección proactiva de sentimientos y opiniones en el sistema.

A partir del objetivo general, se derivan los siguientes **objetivos específicos**:

- Analizar los principales métodos existentes para la detección de polaridad y los algoritmos de aprendizaje supervisado usados para la clasificación de textos en base al sentimiento.
- Crear, determinar y etiquetar la polaridad de forma manual un corpus de entrenamiento de mensajes extraídos de la red de mensajería instantánea de la UCI.
- Implementar un método para la clasificación automática supervisada de los mensajes de la red de mensajería Jabber .
- Evaluar el comportamiento del método implementado a partir de su utilización en un entorno real.

Las **tareas de investigación** que se deben cumplir son:

- Análisis de los principales conceptos, trabajos y herramientas relacionados con el análisis de sentimientos, específicamente en la detección de polaridad en opiniones basadas en textos.
- Diseño de un método para la detección de polaridad en los textos en la red de mensajería instantánea de la UCI.
- Implementación de un método basado en un algoritmo de clasificación automática supervisada para la detección de polaridad en las opiniones de los usuarios del Jabber.
- Validar la propuesta mediante diferentes técnicas encontradas para medir el comportamiento de los algoritmos seleccionados.
- Comprobar el método diseñado mediante ejemplos de mensajes reales.

Para dar cumplimiento a las tareas investigativas se emplearon los siguientes **métodos científicos** de investigación:

**Métodos teóricos**

- **Histórico-Lógico:** se utilizó para estudiar la trayectoria y el desarrollo histórico de las herramientas que se utilizan en el análisis de polaridad del sentimiento y poder comprender la lógica de sus aportes, así como las tendencias actuales.
- **Analítico-Sintético:** permitió analizar individualmente los principales conceptos relacionados con el área de estudio a tratar, posibilitando un análisis profundo de cada uno, para luego llevar a cabo el estudio de las relaciones que se establecen entre ellos.
- **Análisis documental:** se utilizó para el apoyo de la investigación, donde el objetivo es la revisión de artículos, libros, tesis, publicaciones, documentos, que ayude al desarrollo del tema propuesto.

**Métodos empíricos**

- **Observación científica:** se usó para identificar los mejores algoritmos empleados para el aprendizaje automático y poder diseñar el método propuesto para la detección de polaridad en textos.
- **Experimentación:** se empleó para verificar la correspondencia entre los resultados obtenidos con la aplicación del algoritmo propuesto y los resultados de la aplicación de técnicas empleadas en la detección de polaridad en el texto.

El presente documento está estructurado en tres capítulos cuyos contenidos son:

**Capítulo 1 Fundamentación teórica sobre el análisis de sentimientos:** Se definen los conceptos más relevantes relacionados al objeto de estudio. Se detallan las tareas para el análisis de sentimientos. Se realiza un estudio sobre el concepto Opinión. Al finalizar el capítulo se analizan las herramientas que se emplean en el análisis de sentimiento.

**Capítulo 2 Propuesta de solución:** Se describe el método de solución que se propone. El método escogido estará basado en algoritmos de aprendizaje supervisado. Se detallan todos los pasos necesarios para la creación de los modelos aplicando diferentes técnicas y al final se escoge la mejor base.

**Capítulo 3 Validación del método implementado:** Se presenta la validación del método desarrollado mediante varios criterios. Se realiza un análisis de los resultados obtenidos mediante un caso de estudio propuesto que permite comprobar la utilidad del método propuesto.

# Capítulo 1

# Fundamentación teórica sobre el análisis de sentimientos

En este capítulo se describen los conceptos más relevantes relacionados al objeto de estudio: el Procesamiento del Lenguaje Natural, Análisis de Sentimientos, y la Opinión, así como el Proceso de Descubrimiento del Conocimiento con las fases que lo componen. Se detallan las tareas para el análisis de sentimientos y se aborda la definición formal del concepto Opinión. Al finalizar el capítulo se analizan las herramientas que se emplean en el análisis de sentimiento.

## 1.1. Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural (PLN) es una disciplina del área de investigación de la Inteligencia Artificial (IA), que se encarga de analizar e interpretar el lenguaje natural empleado por las personas para comunicarse de diferentes maneras: hablado o a través de textos.

Existen varias definiciones sobre PLN, como: “El concepto de PLN hace referencia a las técnicas de tratamiento del lenguaje y su aplicación en diversas áreas por medio de métodos computacionales. Paralelamente se hace uso de otros términos próximos, como lingüística computacional o ingeniería lingüística”, tributada en Sosa (1997).

“El procesamiento del lenguaje natural es un conjunto de técnicas computacionales teóricamente motivadas para analizar y representar naturalmente textos de origen natural en uno o más niveles de análisis lingüísticos para lograr el propósito de procesar el lenguaje humano por una serie de tareas o aplicaciones”, expresada en Liddy (2001).

Resumiendo, el PLN permite la comunicación entre las personas y las computadoras mediante el tratamiento automático del lenguaje natural a través de técnicas y construcción de herramientas dentro del área de la inteligencia artificial, la computación y la lingüística.

### 1.1.1. Niveles del PLN

El procesamiento en la comprensión del lenguaje algunos investigadores como Feldman (1999); Liddy (2001) siguen las etapas tradicionales de un análisis lingüístico: fonología, morfología, léxico, sintaxis, semántica, pragmática; moviéndose gradualmente del texto a las intenciones detrás de él. Para comprender el procesamiento del lenguaje natural, es importante poder distinguir entre estos, ya que no

todos los sistemas PNL utilizan todos los niveles. Son las funciones a desempeñar por el sistema las que determinan que niveles de análisis deben ser desarrollados. Sobrino Sande (2018) destaca que existen cuatro componentes principales o niveles de análisis, ver Figura 1.1:

- **Nivel de análisis morfológico:** es el componente donde se examina las palabras para extraer raíces, rasgos flexivos, sufijos, prefijos y otros elementos. Su objetivo es entender cómo se construyen las palabras a partir de unidades de significado más pequeñas denominadas morfemas.
- **Nivel de análisis sintáctico:** analiza la estructura de las oraciones en base al modelo gramatical empleado con el objetivo de conocer como se unen las palabras para crear oraciones.
- **Nivel de análisis semántico:** proporciona sentido a las oraciones y les otorga un significado, resolviendo además las ambigüedades léxicas y estructurales que pudieran aparecer.
- **Nivel de análisis pragmático:** se encarga del análisis de los textos más allá del de una oración aislada, teniendo en consideración aquellas inmediatamente anteriores, la relación existente entre ellas y el contexto en el que se producen.

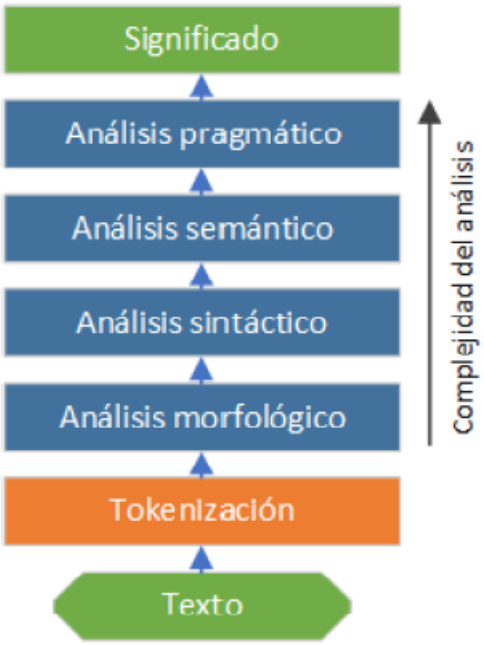



Figura 1.1: Niveles de análisis en aplicaciones de PLN, tomado de Sobrino Sande (2018)

El texto de entrada debe generalmente ser preprocesado, este proceso se puede dividir en dos etapas: clasificación de documentos y segmentación de texto. La selección de documentos es el proceso de convertir un conjunto de archivos digitales en documentos de texto bien definidos, mientras que por otra parte la segmentación de texto es el proceso de convertir un corpus bien definido en sus palabras y oraciones componentes. La segmentación de palabras divide la secuencia de caracteres en un texto al ubicar los límites de las palabras, los puntos donde termina una palabra y comienza otra. Para fines de lingüística computacional, las palabras así identificadas se denominan con frecuencia tokens, y la segmentación de palabras también se conoce como tokenización (Indurkhya y Damerau, 2010).

La Tokenización es una de las operaciones más básicas que pueden aplicarse a un texto, y consiste en dividir una secuencia de caracteres en palabras, signos de puntuación, números y otros elementos. Con esta técnica, no se puede conseguir mucha información sobre la estructura sintáctica, y mucho menos puede conocerse sobre la semántica de esa secuencia, pero se puede realizar una búsqueda de patrones, que es una forma alternativa de detectar nombres u otro tipo de palabras (Saporiti y Tibaldo, 2017).

### 1.1.2. Aplicaciones del PLN

El procesamiento del lenguaje natural es una disciplina que cuenta con un alto potencial y múltiples posibilidades prácticas y teóricas para innumerables aplicaciones. Algunas de las más frecuentes que utilizan PNL son las siguientes (Indurkhya y Damerau, 2010):

- **Recuperación de información (IR del inglés *Information Retrieval*):** El dominio de recuperación de información (IR) se puede ver, hasta cierto punto, como un dominio de PNL aplicado con éxito. La velocidad y la escala de la utilización de la Web en todo el mundo han sido posibles gracias a los motores de búsqueda eficaces y de libre acceso. El objetivo de estos sistemas es la búsqueda y obtención de información a partir de grupos de documentos electrónicos (páginas web, documentos hablados, imágenes, música, vídeo) o documentos reales. Los IR tratan con los elementos de representación, almacenamiento, organización y acceso a la información.
- **Pregunta-respuesta (QA del inglés *Question-Answering*):** Desde una perspectiva muy general, el QA puede definirse como un proceso automático capaz de comprender preguntas formuladas en un idioma natural como el inglés y responder exactamente con la información solicitada. Sin embargo, esta definición aparentemente simple se vuelve muy compleja cuando analizamos en detalle cuáles son las características y la funcionalidad que debe tener un sistema de control de calidad “ideal”. Este sistema debería poder determinar la necesidad de información expresada en una pregunta, ubicar la información requerida, extraerla y luego generar una respuesta y presentarla de acuerdo con los requisitos expresados en la pregunta. Además, este sistema ideal debería ser capaz de interpretar preguntas y procesar documentos escritos en lenguajes naturales de dominio ilimitado, lo que permitiría una interacción cómoda y apropiada por parte de los usuarios.
- **Traducción automática de texto:** Considerada una de las aplicaciones paradigmáticas de los sistemas PLN, entre la que se destaca la traducción automática entre múltiples lenguajes naturales. Los sistemas actuales de traducción automática utilizan enfoques basados en mediciones estadísticas y relaciones entre textos a partir de un entrenamiento previo de ciento o miles de textos. Aunque las traducciones no son siempre perfectas, si son aceptables para la traducción de mensajes en las redes sociales, páginas web, etc.(Sobrino Sande, 2018)
- **Análisis de sentimientos:** Surge de la necesidad de encontrar fuentes relevantes, de extraer oraciones relacionadas con opiniones, leerlas, resumirlas y organizarlas en formas utilizables a través de sistemas automatizados de descubrimiento y resumen de opinión, ya que estas tareas son muy complejas y difíciles de ser detectadas por un lector humano. El análisis de sentimientos o minería de opinión surge de esta necesidad y es un problema desafiante para PLN. Debido a su enorme valor para las aplicaciones prácticas, se selecciona esta área para la investigación del presente trabajo de diploma.

## 1.2. Análisis de sentimientos / Minería de opinión

El *análisis de sentimientos* o *minería de opinión* es el estudio computacional de opiniones, sentimientos y emociones expresado en un texto (Liu et al., 2010).

El *análisis de sentimientos*, también llamado *minería de opinión*, es el campo de estudio que analiza las opiniones, sentimientos, evaluaciones, valoraciones, actitudes de las personas, y emociones hacia entidades como productos, servicios, organizaciones, individuos, problemas, eventos, temas y sus atributos. Debido a la popularización de este tipo de estudios han surgido nombres y tareas ligeramente diferentes para hacer referencia a este campo de investigación, por ejemplo: *análisis de sentimientos, minería de opinión, extracción de opinión, minería de sentimientos, análisis de la subjetividad, análisis de afecto, análisis de emociones, etc*. Sin embargo, ahora están todos bajo el paraguas del análisis de sentimientos o minería de opinión. Mientras que, en la industria, el término *análisis de sentimientos* es más de uso común, pero en la academia tanto el *análisis de sentimientos* como la *minería de opinión* son empleados con frecuencia, básicamente representan el mismo campo de estudio (Liu, 2012).

Las dos expresiones *análisis de sentimientos* o *minería de opinión* son intercambiables, expresan un significado mutuo. Sin embargo, algunos investigadores declaran que tienen conceptos ligeramente diferentes. La *minería de opinión* extrae y analiza la opinión de las personas sobre una entidad, mientras que el *análisis de sentimientos* identifica el sentimiento expresado en un texto y luego lo analiza. Por lo tanto, el objetivo del *análisis de sentimientos* es encontrar opiniones, identificar los sentimientos que expresan y luego clasificar su polaridad, además puede considerarse un proceso de clasificación como se muestra en la Figura 1.2 (Medhat et al., 2014).

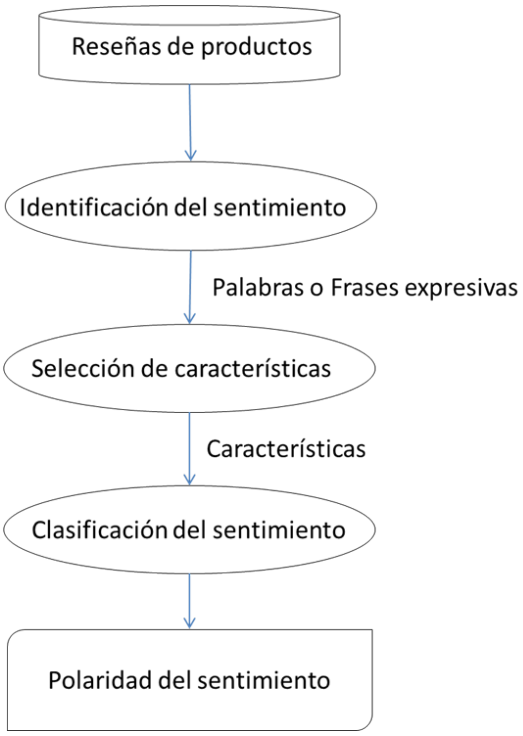



Figura 1.2: Proceso de análisis de opiniones sobre revisiones de productos, traducido de Medhat et al. (2014)

En la investigación de Sobrino Sande (2018) se enuncian diversos trabajos sobre los verdaderos creadores de los métodos que más se utilizan a la hora de capturar el sentimiento global de textos y documentos. (Pang et al., 2002) es el primer trabajo conocido que hace uso de algoritmos de aprendizaje automático para la clasificación de textos; en este caso, críticas de películas extraídas de sitio web [3] especializado en esta temática. Otro precursor es P.D Turney que en su estudio (Turney, 2002) muestra un sistema que es capaz de clasificar opiniones de usuarios sobre diversos productos y servicios como automóviles, viajes o películas, mediante un análisis gramatical de las oraciones y una serie de consultas en el motor de búsquedas AltaVista[4]. Cabe también destacar (Taboada et al., 2011) que hace uso de diccionarios de palabras etiquetadas con su polaridad para la clasificación de textos y (Pak y Paroubek, 2010) que usa algoritmos de aprendizaje automático para clasificar mensajes de Twitter en base a su sentimiento y además presenta un sistema para crear automáticamente un corpus con el que entrena dichos algoritmos.

El número de trabajos es tan extenso que existen estudios cuyo objetivo principal es analizar el estado del arte mostrando las publicaciones más importantes sobre el tema. Dos recursos notables sobre este tipo de estudios podrían ser (Medhat et al., 2014) y (Kumar y Teeja, 2012). Además de estos trabajos de investigación, también es necesario destacar el estudio de divulgación de Pan y Lee (Pang et al., 2008) donde se muestran diversas técnicas y procedimientos para la construcción de sistemas de análisis de sentimientos, así como el libro de (Liu, 2012) en que se desarrollan en detalle todos los conceptos relacionados con el análisis de sentimientos y cuyos planteamientos siguen vigentes hasta la fecha de hoy.

En los últimos años el campo de análisis de sentimientos se ha extendido hacia el análisis en otros idiomas pocos conocidos como el portugués (Pereira, 2020) y el árabe (Al-Ayyoub et al., 2019) y a otras áreas de investigaciones como, el impacto emocional de las imágenes presentado por (Ortis et al., 2020), acercándose a los trabajos de tratamiento de las emociones, la actitud y la opinión desde el formato audiovisual, conocidos como análisis de sentimiento multimedia (MAS por sus siglas en inglés *Multimodal Sentiment Analysis*) investigadas por (Chakraborty et al., 2020; Kaur y Kautish, 2019; Li et al., 2019; Yue et al., 2019).

### 1.2.1. Conceptos asociados a la opinión

Con el objetivo de exponer los elementos del análisis de sentimientos, se relacionarán a continuación conceptos asociado a la investigación expresados en Liu et al. (2010).

**Emociones:** Las emociones son los sentimientos y pensamientos subjetivos.

Las emociones se han estudiado en muchos campos, por ejemplo: psicología, filosofía, sociología, biología, etc. Sin embargo, todavía no hay un conjunto de emociones básicas acordadas entre los investigadores. Basado en Parrott (2001), las personas tienen 6 tipos de emociones primarias, es decir: amor, alegría, sorpresa, ira, tristeza y miedo, que pueden ser subdividido en muchas emociones secundarias y terciarias. Cada emoción también puede tener diferentes intensidades.

[3] Se hace referencia a la base de datos de películas "rec.arts.movies". Actualmente se conoce como IMDb (por sus siglas en ingés *Internet Movie Database*) una base de datos en línea de información relacionada con películas, programas de televisión, videos caseros, videojuegos y contenido de transmisión en línea, reseñas críticas y de fans. Se puede acceder y descargar los archivos del conjunto de datos desde https://datasets.imdbws.com/

[4] https://search.yahoo.com/?fr=altavista

Las fortalezas de las opiniones están estrechamente relacionadas con las intensidades de ciertas emociones, por ejemplo: alegría e ira. Sin embargo, los conceptos de emociones y opiniones no son equivalentes, aunque tienen una gran intersección.

Cuando se discuten sentimientos subjetivos de emociones u opiniones, es útil distinguir dos diferentes nociones: estados mentales (o sentimientos) de las personas y expresiones del lenguaje utilizadas para describir los estados mentales.

Del mismo modo, también hay una gran cantidad (aparentemente ilimitada) de expresiones de opinión que describen sentimientos positivos o negativos. El análisis de sentimientos o la minería de opinión esencialmente trata de inferir los sentimientos de las personas en función de sus expresiones lingüísticas.

Ahora describimos el objetivo del análisis de sentimientos o la minería de opinión, que no solo pretende inferir opiniones/sentimientos positivos o negativos del texto, sino también descubrir las otras piezas de información que son importantes para las aplicaciones prácticas de las opiniones (Liu et al., 2010).

**Objeto:** un *objeto* ***O*** es una entidad que puede ser un producto, persona, evento, organización o tema. Se asocia con un par, ***O: (T, A)***, donde **T** es una jerarquía de *componentes (o partes), subcomponentes*, y así sucesivamente, y ***A*** es un *conjunto de atributos de O*. Cada componente tiene su propio conjunto de subcomponentes y atributos.

El uso de características para un objeto es bastante común en el dominio del producto, ya que las personas a menudo usan el término características del producto. Sin embargo, cuando los objetos son eventos y temas, el término característica puede no sonar natural. De hecho, en algunos otros dominios, los investigadores también usan el término tema o aspecto para referirse a la característica.

Se usará el término característica junto con el término objeto. Deje que un *documento con opiniones* sea ***d***, que puede ser una revisión del producto, una publicación en el foro o un blog que evalúa un conjunto de objetos, en el caso más general, ***d*** consiste en una secuencia de oraciones $(d = S_1, S_2, ..., S_m)$.

***Pasaje de opinión sobre una característica:*** un *pasaje de opinión* sobre una característica ***f*** de un objeto ***O*** evaluado en ***d*** es un grupo de oraciones consecutivas en ***d*** que expresa una opinión positiva o negativa sobre ***f***.

Es posible que una secuencia de oraciones (al menos una) en un documento con opiniones expresen juntas una opinión sobre un objeto o una característica del objeto. También es posible que una sola oración exprese opiniones en más de una función.

***Característica explícita e implícita:*** si una característica ***f*** o cualquiera de sus sinónimos aparecen en una oración ***S***, ***f*** se llama una *característica explícita* en ***S***. Si ni ***f*** ni ninguno de sus sinónimos aparecen en ***S*** pero ***f*** está implícito, entonces ***f*** se llama una característica implícita en ***S***.

***Titular de opinión:*** el *titular* de una opinión es la persona u organización que expresa la opinión. Los titulares de opiniones también se denominan *fuentes de opinión* (Wiebe et al., 2005). En el caso de reseñas de productos y blogs de opinión, los titulares suelen ser los autores de las publicaciones.

***Opinión:*** Una opinión sobre una característica ***f*** es una visión, actitud, emoción o actitud positiva o negativa, o valoración en ***f*** de un titular de opinión.

Una opinión puede ser **directa** o **comparativa**. Las **opiniones directas** se realizan sobre una sola entidad, mientras que la **opinión comparativa** expresa una relación de similitudes o diferencias entre

dos o más objetos, y/o preferencias de objeto del titular de la opinión en función de algunos de las características compartidas de los objetos. Una *opinión comparativa* generalmente se expresa usando el comparativo o forma superlativa de un adjetivo o adverbio, aunque no siempre.

***Orientación de opinión:*** la orientación de una opinión sobre una característica ***f*** indica si la opinión es *positiva, negativa o neutral*. La orientación de opinión también se conoce como *orientación de sentimiento, polaridad de opinión u orientación semántica*.

Ahora se define un modelo de un objeto de una opinión, el cual se representara mediante una quíntuple $(O_j, f_{jk}, P_{ijkl}, h_i, t_l)$, donde $O_j$ es un objeto, $f_{jk}$ es un característica del objeto $O_j$, $PO_{ijkl}$ es la orientación o polaridad de la opinión sobre la característica $f_{jk}$ del objeto $O_j$, $h_i$ es el titular de la opinión y $t_l$ es el momento en que $h_i$ expresa la opinión. La opinión orientación $P_{ijkl}$ puede ser positivo, negativo o neutro (o se mide en base a una escala más granular para expresar diferentes puntos fuertes de opiniones (Wilson et al., 2004)). Para la característica $f_{jk}$ sobre la cual el titular de la opinión comenta, elige una palabra o frase del conjunto de sinónimos correspondiente $W_{jk}$, o una palabra o frase del conjunto de indicadores de característica correspondiente $I_{jk}$ para describir la característica, y luego expresar un valor positivo, negativo o opinión neutral sobre la función.

**Subjetividad de la oración:** una *oración objetiva* expresa cierta información objetiva sobre el objeto o entidad, mientras que una *oración subjetiva* expresa algunos sentimientos o creencias personales. Las expresiones subjetivas vienen en muchas formas, por ejemplo: opiniones, alegaciones, deseos, creencias, sospechas y especulaciones (Riloff et al., 2006; Wiebe et al., 2000). Por lo tanto, una oración subjetiva puede no contener una opinión. Del mismo modo, también debemos tener en cuenta que no todas las oraciones objetivas contienen una opinión.

**Opinión explícita e implícita:** una *opinión explícita* sobre la característica ***f*** es una opinión explícitamente expresado en ***f*** en una oración subjetiva. Una *opinión implícita* sobre la característica ***f*** es una opinión sobre ***f*** implícita en una oración objetiva.

**Oración con opinión:** una *oración con opinión* es una oración que expresa explícita o opiniones implícitas positivas o negativas. Puede ser una oración subjetiva u objetiva.

Como podemos ver, los conceptos de oraciones subjetivas y oraciones de opinión no son los mismos, aunque las oraciones de opinión son a menudo un subconjunto de oraciones subjetivas. Los enfoques para identificarlos son similares.

### 1.2.2. Tareas del análisis de opiniones

El ***objetivo*** del análisis de sentimientos: es dado un documento ***d***, descubrir todos los quíntuples de opinión $(O_j, f_{jk}, P_{ijkl}, h_i, t_l)$, en ***d***. Para completar la quintupla de opinión se realizan las siguientes tareas que se derivan de los cinco componentes de la quíntupla (Liu, 2012):

- **Tarea 1 (extracción y categorización de entidades):** extraer todas las expresiones de entidades en el documento ***d*** y categorizar o agrupar expresiones de entidades sinónimas en grupos de entidades o categorías. Cada grupo de expresión de entidad indica una entidad única $O_j$.
- **Tarea 2 (extracción y categorización de aspectos):** extraer todas las expresiones de aspecto de las entidades y categorizar estas expresiones de aspecto en grupos. Cada grupo de expresiones de aspecto de la entidad $O_j$ representa un aspecto único $f_{jk}$.

- **Tarea 3 (extracción y categorización del titular de la opinión):** extraer los titulares de opiniones de texto o datos estructurados y categorizarlos. La tarea es análoga a las dos tareas anteriores.
- **Tarea 4 (extracción de tiempo y estandarización):** extraer los tiempos en que se dan las opiniones y estandarizar diferentes formatos de tiempo. La tarea es también análoga a las tareas anteriores.
- **Tarea 5 (clasificación de sentimiento de aspecto):** por cada par de entidad $O_j$ y aspecto $f_{jk}$ se debe determinar la valoración $PO_{ijkl}$ emitida por el autor de la opinión. Esta valoración puede ser positivo, negativo y neutral o una calificación numérica de sentimiento.
- **Tarea 6 (generación de quintuples de opinión):** producir todos los quintuples de opinión $(O_j, f_{jk}, PO_{ijkl}, h_i, t_l)$ expresados en el documento ***d*** basado en los resultados de las tareas anteriores. Esta tarea es aparentemente muy simple, pero de hecho es muy difícil en muchos casos.

### 1.2.3. Niveles del Análisis de sentimientos

En general, el análisis de sentimientos en un documento ha sido investigado principalmente en tres niveles de acuerdo a la granularidad, profundidad y detalles requeridos (Liu, 2012):

**Nivel de documento:** la tarea en este nivel es clasificar si un documento de opinión completo expresa un sentimiento positivo o negativo (Pang et al., 2002; Turney, 2002). Por ejemplo, dada una revisión del producto, el sistema determina si la revisión expresa una opinión general positiva o negativa sobre el producto. Esta tarea se conoce comúnmente como clasificación de sentimientos a nivel de documento. Este nivel de análisis supone que cada documento expresa opiniones sobre una sola entidad (por ejemplo, un solo producto). Por lo tanto, no es aplicable a documentos que evalúan o comparan múltiples entidades.

**Nivel de oración o frase:** la tarea en este nivel va a las oraciones y determina si cada oración expresó una opinión positiva, negativa o neutral. La clasificación neutral generalmente significa que no hay opinión. Este nivel de análisis está estrechamente relacionado con la clasificación de subjetividad (Wiebe et al., 1999), que distingue oraciones que expresan información y las oraciones que expresan opiniones. Sin embargo, debemos tener en cuenta que la subjetividad no es equivalente al sentimiento, ya que muchas oraciones objetivas pueden implicar opiniones.

**Nivel de entidad y aspecto:** tanto el nivel de documento como el análisis de nivel de oración no descubren qué es exactamente lo que a la gente le gustó y no le gustó. El nivel de aspecto realiza un análisis más detallado. En lugar de mirar las construcciones del lenguaje (documentos, párrafos, oraciones, cláusulas o frases), el nivel de aspecto mira directamente la opinión misma. Se basa en la idea de que una opinión consiste en un sentimiento (positivo o negativo) y un objetivo (opinión).

### 1.2.4. Dificultades de clasificación en el análisis de sentimientos

El análisis de sentimiento es una tarea más complicada que la clasificación de texto tradicional debido a varios factores, como son (Aisopos et al., 2012):

- *Escasez*, debido a que los textos comúnmente usados son cortos.

- *Vocabulario no estándar*, los textos al ser creados en redes sociales presentan *alto porcentaje de jergas y contracciones gramaticales* propias del individuo.
- *Ruido* debido a la presencia de errores gramáticas, incomprensibilidad de contenido, etc.
- *Multi–idioma*, textos en el idioma materno de quien escribe con ciertas palabras o frases en un idioma foráneo.
- Presencia de *sarcasmo*, entre otros.

### 1.2.5. Clasificación de la polaridad

La clasificación de la polaridad de un mensaje dado se considera la tarea central del análisis de sentimientos. Un texto se clasifica como positivo o negativo si su mensaje general expresa un sentimiento positivo o negativo. En ausencia de cualquier polaridad detectada, un mensaje es neutral. Otra alternativa, a la tarea de clasificación binaria tradicional con valores de clase positivos y negativos como posibles, es una tarea de clasificación multi-etiqueta que incluye también el valor neutral como posible valor de polaridad para denotar aquellos casos en los cuales el texto no transmite ninguna opinión. Tal marco de evaluación generalmente está relacionado con la tarea de clasificación de la subjetividad. La clasificación del grado de subjetividad de un mensaje puede convertirse en una tarea independiente, aunque generalmente se considera algo dependiente de la detección de polaridad, es decir, solo las expresiones subjetivas se pueden polarizar (Basile et al., 2018).

Identificar la polaridad de los sentimientos puede parecer trivial para la mente humana. Sin embargo, la automatización de este proceso no es así de simple. Una serie de estudios han sido desarrollados para encontrar soluciones a este problema. El método más sencillo para la clasificación del sentimiento es armar listas de palabras que se dividen en dos grupos, el primer grupo es una lista de palabras y sinónimos con polaridad positiva (por ejemplo: buena, óptimo, excelente) y el segundo es una lista de palabras y de sinónimos con polaridad negativa (por ejemplo: mala, ruin, desagradable).

A partir de estos dos grupos, se busca en el texto la ocurrencia de las palabras que aparecen en la lista, y el texto se clasifica según el número de palabras positivas y negativas localizadas (Kim y Hovy, 2004). A partir de esta propuesta inicial, se han desarrollado diversos enfoques para la detección de la polaridad de las opiniones, así como diferentes nomenclaturas para describir estos enfoques. En Poirier et al. (2011) describen dos enfoques para clasificar las opiniones: el enfoque lingüístico y el estadístico. El primer enfoque consiste en aplicar técnicas de aprendizaje automático y el segundo consiste en calcular estadísticos a partir de los términos presentes en las opiniones. En general, los sistemas que aplican estas técnicas clasifican los comentarios textuales en dos clases (positivos y negativos). Según el autor los dos enfoques pueden ser combinados (Amores Fernández, 2016).

#### 1.2.5.1. Clasificación de la polaridad general de un documento

El análisis a nivel de documento es probablemente al que se le dedica un mayor porcentaje de los estudios que cada año se publican sobre esta área de investigación y su objetivo principal consiste en clasificar un documento en base al sentimiento que en él se expresa. Esta tarea también se conoce como clasificación de sentimientos en documentos. Los documentos son considerados las unidades básicas de información y éstos pueden ser: opiniones de blog, en tiendas online, sitios web especializado o

mensajes en redes sociales. La opinión generalmente toma un valor entre tres posibles: sentimiento positivo, negativo o neutro, aunque, existen otras escalas, como son las: numéricas, continuas o discretas (Sobrino Sande, 2018).

Tomando como referencia la definición formal de opinión descrita en secciones anteriores, se representa la clasificación de sentimiento a nivel de documento a través de la siguiente quíntupla:

$$(\_, GENERAL, s, \_, \_) \tag{1.1}$$

Dado un documento ***d***, este nivel de análisis trata de determinar el sentimiento ***s*** del aspecto *GENERAL* de la entidad ***e***. La entidad ***e***, el autor de la opinión ***h*** y el momento en que es emitida ***t*** son conocidos o irrelevantes. El valor ***s*** puede ser una categoría entre varias posibles (por ejemplo, positivo, negativo o neutro) o bien un valor numérico (por ejemplo, un valor entre el 1 y 5). Para poder asegurar que este proceso de clasificación puede ser llevado a la práctica, es necesario asumir que el documento que se quiere clasificar expresa una opinión sobre una única entidad ***e*** y dicha opinión pertenece a una única persona ***h***.

### 1.2.6. Técnicas de clasificación de sentimientos en documentos

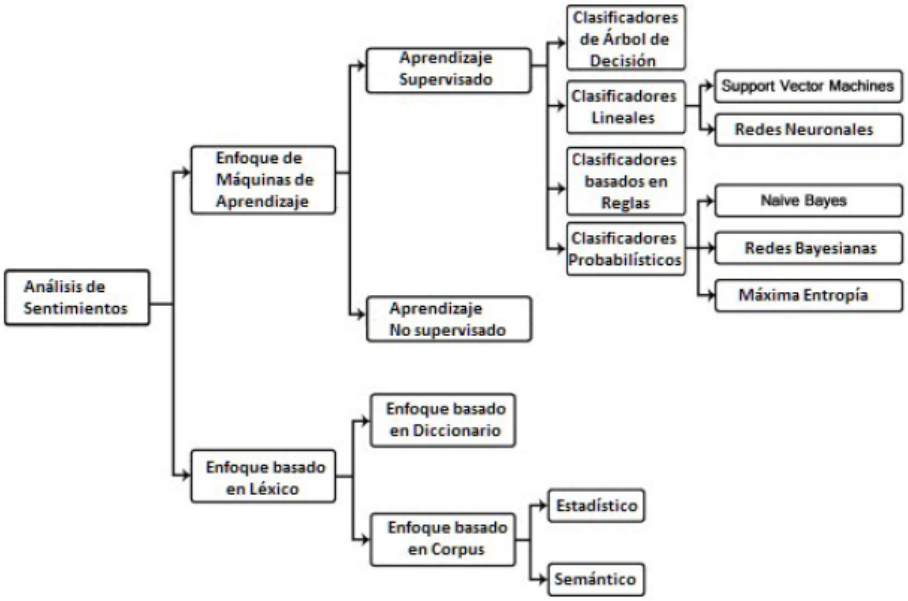



Figura 1.3: Técnicas de clasificación de sentimientos, traducido de (Medhat et al., 2014)

Un aspecto importante que influye en el aprendizaje es el grado de supervisión. La clasificación de sentimiento de un documento se puede realizar mediante diversas técnicas y métodos, que se dividen en un enfoque de aprendizaje automático, un enfoque basado en léxico y un enfoque híbrido (Maynard y Funk, 2011). El enfoque basado en el léxico se basa en un léxico de sentimientos, una colección de términos de sentimientos conocidos y recopilados. Se divide en un enfoque basado en diccionario y un enfoque basado en corpus que utilizan métodos estadísticos o semánticos para encontrar la polaridad de los sentimientos. El enfoque híbrido mediante léxicos combina ambos enfoques y es muy común con los léxicos de sentimiento que juegan un papel clave en la mayoría de los métodos (Medhat et al., 2014).

Los métodos de clasificación de texto que utilizan los algoritmos de aprendizaje automático (ML por sus siglas en inglés *Machine Learning*) se dividen en métodos de aprendizaje supervisados y no supervisados.

Los métodos supervisados necesitan de un grupo de documentos de ejemplos previamente etiquetados para generar un modelo que será usado posteriormente para clasificar nuevos textos y que en este contexto es conocido como *"corpus"*. Este tipo de aprendizaje tiene la ventaja de que no es necesario que al modelo se le muestren todos los ejemplos existentes, es decir puede clasificar un ejemplo sin haberlo visto nunca. La desventaja es que a pesar de lo anterior, si es necesario contar con una gran cantidad de ejemplos para el entrenamiento.

Mientras que los métodos no supervisados tratan de inferir la polaridad del sentimiento global de un documento a partir de la orientación semántica de las palabras o frases que lo conforman, a través de dos enfoques uno basado en diccionarios y otro en relaciones lingüísticas (Sobrino Sande, 2018). Este tipo de aprendizaje tiene la ventaja de que no es necesaria la presencia de un modelo para el aprendizaje o de un conjunto de entrenamiento.

Cabe destacar que existe otro método de aprendizaje automático que utiliza una combinación de aprendizaje supervisado y no supervisado. A este se le llama método de aprendizaje híbrido La idea aquí es aprender con los datos asociados a su clase y asociar una clase a los datos que no contienen asociada una clase.

## 1.3. Proceso de Descubrimiento del Conocimiento (KDD)

El proceso de extracción del conocimiento en los datos (KDD por sus siglas del inglés *Knowledge Discovery in Databases*) se puede definir como el *proceso no trivial* de identificar modelos y/o patrones válidos, previamente desconocidos, potencialmente útiles y humanamente comprensibles, a partir de grandes conjuntos de datos, con el objetivo de predecir tendencias y comportamientos de los datos (Fayyad et al., 1996b). El término *proceso* implica que KDD involucra muchos pasos y el término *no trivial* implica que no se tratan de cálculos sencillos.

El proceso KDD es interactivo e iterativo, involucra numerosos pasos con muchas decisiones que deben ser tomadas por el usuario (Fayyad et al., 1996a,b):

- **Selección:** consiste en la creación o elección de un conjunto de datos objetivo, sobre los que el descubrimiento de conocimiento será realizado. Una consideración importante en esta etapa es que los datos pueden proceder de diferentes fuentes y, por tanto, deben ser combinados.
- **Preprocesamiento:** tiene por objetivo asegurar la calidad de los datos a analizar con el fin de obtener datos consistentes, ya que de ello depende, en gran medida, la calidad del conocimiento a descubrir. Las operaciones básicas incluyen el filtrado de valores atípicos (datos que no se ajustan al comportamiento general de los datos), la eliminación del ruido, estrategias para el manejo de los campos de datos vacíos o la normalización de los datos.
- **Transformación:** consiste en modificar la estructura de los datos usando reducción de dimensionalidad o métodos de transformación, con el objetivo de facilitar el análisis. Los datos son transformados en formas apropiadas para la minería de datos y/o se seleccionan los atributos más útiles

capaces de representar los datos dependiendo de las metas propuestas.

- **Minería de Datos:** es un proceso donde un conjunto de métodos, técnicas y herramientas son aplicados con el fin de realizar una búsqueda y extracción de patrones de interés. Esta etapa se puede abordar, en función del problema a resolver, desde dos puntos de vistas distintos:
  - **predictivo**, en el que se intenta obtener conocimiento para clasificación o predicción.
  - **descriptivo**, cuyo objetivo fundamental es descubrir conocimiento de interés dentro de los datos, intentando obtener información que describa el modelo que existe detrás de los datos.
- **Interpretación/Evaluación:** consiste en la interpretación y evaluación de los patrones encontrados. Se identifican los patrones realmente interesantes basados en alguna medida de interés.

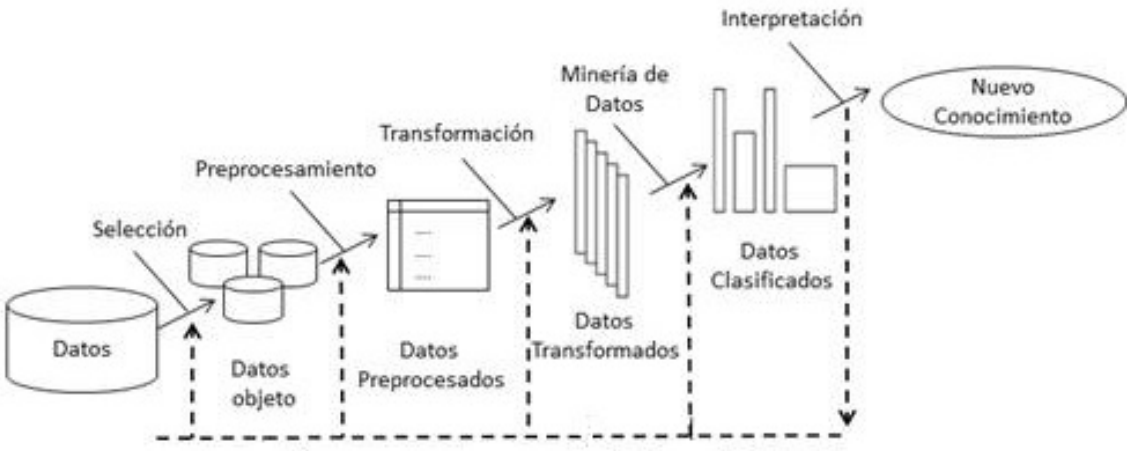



Figura 1.4: Una descripción general de los pasos que componen el proceso KDD, tomado de (Hernández Orallo et al., 2004)

.

El proceso KDD debe ser precedido por el desarrollo de un entendimiento del dominio de aplicación (área específica donde se aplicará) y la correcta identificación de las metas desde el punto de vista del usuario final (Azevedo y Santos, 2008; Fayyad et al., 1996a). Una vez finalizado el proceso KDD, el conocimiento descubierto puede ser usado directamente, incorporado a otros sistemas para futuras acciones o simplemente documentado y reportado a las partes interesadas (Fayyad et al., 1996b). KDD puede involucrar importantes iteraciones y contener bucles entre pares de etapas (Ver Figura 1.4). La mayor parte del trabajo en el proceso KDD está enfocado en las etapas de preprocesamiento y minería, aunque el resto de las etapas son igualmente importantes para la exitosa aplicación de KDD en la práctica (Fayyad et al., 1996b).

## 1.4. Soluciones actuales

#### Herramientas para el análisis de sentimientos

En la actualidad, existen diversas herramientas de análisis de sentimiento publicadas en la Web. Algunas pueden encontrarse en forma de librería o código fuente, para ser estudiadas o utilizadas en algún proyecto. Las implementaciones que se pueden hallar en este formato, en general no son lo

suficientemente potentes; un ejemplo es **sentiment**, una librería para usar en node.js, que básicamente tiene un diccionario de palabras, en el cual, para cada palabra, tiene un puntaje que indica si la palabra es positiva, negativa o neutral. Para procesar el texto, sentiment usa ese diccionario para reconocer todas las palabras sueltas que pueda y sumar el peso, para definir la opinión expresada en la frase (Saporiti y Tibaldo, 2017).

Existen varias herramientas que permiten realizar minería de opinión, entre ellas se encuentra **ITelligent**[5] que es un sistema de minería de opinión para inteligencia comercial; **OPAL**[6] plugin de Drupal que analiza comentarios realizados por los usuarios y detecta si el resultado es positivo, negativo o neutral; **IIC-Lynguoque**[7] es un conjunto de herramientas que ayudan a extraer la opinión positiva, negativa y neutra de un texto; **netOpinion**[8] permite conocer opiniones de productos o servicios en foros y redes sociales; **WebOpinion**[9] gestiona mes a mes la evolución de la imagen de un usuario en la red y **Sentitex**[10] que consiste en un conjunto de aplicaciones para el análisis de sentimientos en textos, entre otras. Desafortunadamente la mayoría de estas herramientas son propietarias y es necesario pagar la licencia para su utilización, ya que fueron concebidas con fines comerciales. Adicionalmente, cada una de ellas responde a objetivos específicos de la minería de opinión (Amores Fernández, 2016).

**Herramientas para la detección de polaridad**

**PosNeg Opinion** permite detectar de manera no supervisada la polaridad de las opiniones. PosNeg Opinion fue desarrollada completamente en JAVA, por lo que es multiplataforma. Necesita como entrada un fichero XML con todas las opiniones a analizar y como salida muestra cuántas fueron positivas y cuántas negativas. A petición del usuario también retorna el porcentaje de las opiniones negativas y positivas, así como una lista con las opiniones negativas y otra con las opiniones positivas. Adicionalmente, la aplicación destaca cuáles opiniones fueron las de mayor puntuación en cada caso (positivas/negativas) (Amores Fernández, 2016).

**MOAS-Les** es un sistema de clasificación de documentos escritos en español. MOAS-Les está caracterizado principalmente por la estructura sintáctica de las oraciones, así como el empleo de diccionarios semánticos, tratamiento de los intensificadores y oraciones adversativas. Es una herramienta para determinar la polaridad de documentos fue implementada con Java en su versión 7, empleando como herramientas de apoyo el Freeling 3.1. El proceso de clasificación es completamente no supervisado. El algoritmo de clasificación en que se apoyan es una ligera modificación del lexicón multilingüe de polaridades semánticas, llamado ML-SentiCon con mejoras basados en el SentiWordNet 3.0, para resolver el cálculo de la orientación de opiniones emplean técnicas de análisis de dependencias y el uso de intensificadores y atenuadores (López-Palma y Estévez-Crespo, 2016).

**AOpinion** está diseñada para ayudar a los desarrolladores de los Centros de Desarrollo de la Universidad de las Ciencias Informáticas a clasificar de forma automática los comentarios de los usuarios que utilizan dichos software. AOpinion, permite que el usuario suba un archivo con extensión .txt que contenga los comentarios de usuarios para ser clasificados en positivo, negativo y neutro, además de mostrar por categorías los comentarios positivos y negativos (Lazara Barrios Domínguez, 2016).

[5] http://www.itelligent.es
[6] http://gi2mo.org/app/opal/opal v1.3.2.zip
[7] http://innova.iic.uam.es/lynguo/
[8] http://netopinion.itelligent.es/
[9] http://www.webopinion.es
[10] http://www.sentitext.com

## 1.5. Conclusiones del capítulo

A partir del análisis de la información presentada en este capítulo puede arribarse a las siguientes conclusiones:

1. El estudio de las principales fuentes bibliográficas permitió identificar los aspectos fundamentales que caracterizan al análisis de sentimientos, permitiendo detectar que las herramientas actuales dedicadas a la detección de la polaridad del sentimiento no satisfacen al problema de investigación.
2. El análisis de sentimientos es la técnica utilizada para dar solución al problema planteado ya que permite determinar la polaridad del sentimiento en las conversaciones de los usuarios de la red.
3. Los pasos descrito por KDD se adaptan al proceso análisis de sentimientos en los mensajes extraídos del Jabber, permitiendo hacer un procesamiento del lenguaje natural para poder realizar una minería sobre los datos.
4. Los algoritmos de aprendizaje automático supervisado permiten resolver el problema planteado debido a su efectividad y éxito en el campo del análisis de sentimientos.

# Capítulo 2

# Propuesta de solución

En este capitulo se presenta un método para resolver el problema de clasificación de textos por su sentimiento a nivel de documento. El método escogido estará basado en algoritmos de aprendizaje supervisado. Para ello se ha creado un corpus cuyos ejemplos han sido etiquetados previamente según a la categoría del sentimiento al que pertenecen. Se detallan todos los pasos necesarios para la creación de los modelos aplicando diferentes técnicas y al final se escoge la mejor base.

## 2.1. Tecnologías empleadas

### 2.1.1. Lenguaje de programación

**Python v.3.7**[11] es un lenguaje de programación interpretado, orientado a objetos de alto nivel y con semántica dinámica. Su sintaxis hace énfasis en la legibilidad del código, lo que facilita su depuración y, por tanto, favorece la productividad. Ofrece la potencia y la flexibilidad de los lenguajes compilados con una curva de aprendizaje suave (Luca, 2020). Python ha desarrollado una gran y activa comunidad científica de computación y análisis de datos. Ha pasado de ser un lenguaje informático científico a ser uno de los más importantes para la ciencia, el aprendizaje automático y el desarrollo de software en la academia y la industria (McKinney, 2012).

### 2.1.2. Plataforma y marco de desarrollo

**Anaconda v.2019.07**[12] es una distribución de código libre y abierto de los lenguajes Python y R, que facilita el proceso de implementar soluciones relacionadas con la manipulación de datos. Cuenta con una plataforma escalable, segura y lista para colaborar e implementar proyectos de ciencia de datos. Anaconda presenta un repositorio de última generación para todo lo relacionado con ella. Esta plataforma presenta un administrador de paquetes, administrador de entorno y distribución de Python gratuito, fáciles de instalar. Tiene un servicio de administración de paquetes que facilita la búsqueda, el acceso, el almacenamiento y el intercambio de cuadernos y entornos públicos, así como paquetes Conda y PyPI (Ana, 2020).

Algunas de las librerías y módulos más relevantes que esta incluye son:

[11] https://www.python.org/
[12] https://www.anaconda.com/

**NLTK v.3.4.4**[13] es una plataforma líder para crear programas Python que funcionen con datos de lenguaje humano. Proporciona interfaces fáciles de usar para más de 50 corpus y recursos léxicos, junto con un conjunto de bibliotecas de procesamiento de texto para clasificación, tokenización, derivación, etiquetado, análisis y razonamiento semántico, envoltorios para bibliotecas de PNL de nivel industrial, y un foro de discusión activo. NLTK está disponible para Windows, Mac OS X y Linux. Lo mejor de todo es que NLTK es un proyecto gratuito, de código abierto e impulsado por la comunidad. NLTK ha sido llamado *"una herramienta maravillosa para enseñar y trabajar en lingüística computacional usando Python"* y *"una biblioteca increíble para jugar con el lenguaje natural"* (NLT, 2020) .

**Pandas v0.24.2**[14] proporciona estructuras y funciones de datos de alto nivel diseñadas para trabajar con datos estructurados o tabulares de manera rápida, fácil y expresiva. Ha ayudado a que Python sea un entorno de análisis de datos potente y productivo. Proporciona una sofisticada funcionalidad de indexación para facilitar la remodelación y realizar agregaciones y seleccionar subconjuntos de datos (McKinney, 2012).

**NumPy v1.16.4**[15] es el paquete fundamental para la computación científica con Python. Contiene: un potente objeto de matriz N-dimensional, herramientas para integrar código C / C++ y Fortan, álgebra lineal útil, transformada de Foyrier y la capacidad de generar números aleatorios. También puede ser utilizado como un eficiente contenedor multidimensional de datos genéricos. Se pueden definir tipos de datos arbitrarios. Esto permite a NumPy integrarse de forma transparente y rápida con una amplia variedad de bases de datos

**Scikit-learn v0.22.1**[16] es un módulo de Python que integra una amplia gama de algoritmos de aprendizaje automático de última generación para problemas supervisados y no supervisados de mediana escala. Este paquete se centra en llevar el aprendizaje automático a los no especialistas mediante un lenguaje de alto nivel de uso general. Utiliza una interfaz coherente y orientada a tareas, lo que permite una fácil comparación de métodos para una aplicación determinada. Dado que se basa en el ecosistema científico de Python, se puede integrar fácilmente en aplicaciones fuera del rango tradicional de análisis de datos estadísticos.Construido sobre NumPy, SciPy y MatPlotLib (Pedregosa et al., 2011).

**MatPlotLib v.3.10**[17] es una biblioteca para la generación de gráficos a partir de datos contenidos en listas o arrays en el lenguaje de programación Python y su extensión matemática NumPy. Proporciona una API, pylab, diseñada para recordar al MATLAB. Produce figuras de calidad de publicación en una variedad de formato impreso y entornos interactivos en todas las plataformas (Mat, 2020).

**Re v.2020.4**[18] es un módulo de Python integrado en *regex*[19], que proporciona facilidades en el trabajo con expresiones regulares. Una expresión regular especifica un conjunto de cadenas que coinciden con ella; las funciones de este módulo permiten verificar si una cadena en particular coincide con una expresión regular dada. Siempre que sigamos sus reglas, podremos realizar búsquedas simples y avanzadas, esto hace que sus opciones más útiles e importantes de cualquier lenguaje.

[13] https://www.nltk.org/
[14] https://pandas.pydata.org/
[15] https://numpy.org/
[16] https://scikit-learn.org/stable/index.html
[17] https://matplotlib.org/
[18] https://docs.python.org/3/library/re.html
[19] https://docs.python.org/3/howto/regex.html

### 2.1.3. Entorno de desarrollo

Un entorno de desarrollo integrado en inglés *Integrated Development Environment (IDE)*, es un software que proporciona servicios integrales para facilitarle al programador el desarrollo del software.

**Pycharm v2019.2.3**[20] es un IDE que proporciona el completamiento inteligente de códigos, inspecciones de códigos, resaltando los errores sobre la marcha y soluciones rápidas, junto con refactorizaciones automáticas de códigos y capacidades de navegación avanzadas. Se integra con IPython Notebook, tiene una consola interactiva de Python y es compatible con Anaconda, así como con múltiples paquetes científicos, incluidos MatPlotLib y NumPy. Además de Python, PyCharm admite JavaScript, CoffeeScript, TypeScript, Cython, SQL, HTML/CSS, AngularJS, Node.js y más. También posee una gran colección de herramientas lista para usar: un depurador integrado y un corredor de prueba; Python Profiler; una terminal incorporada; e integración con sistemas de control de versiones y herramientas de base de datos incorporadas (PyC, 2020).

### 2.1.4. Sistema de control de versiones

**GIT** [21] es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar todo, desde proyectos pequeños a muy grandes, con velocidad y eficiencia. Git es fácil de aprender y ocupa poco espacio con un rendimiento increíblemente rápido. GitHub es un servicio de alojamiento de repositorios de Git, pero agrega muchas de sus propias características. GitHub proporciona una interfaz gráfica basada en web. También proporciona control de acceso y varias funciones de colaboración, como wikis y herramientas básicas de gestión de tareas para cada proyecto.

La funcionalidad insignia de GitHub es la “bifurcación”: copiar un repositorio de la cuenta de un usuario a otro. Esto le permite tomar un proyecto para el que no tiene acceso de escritura y modificarlo en su propia cuenta. Además de sus repositorios públicos de código abierto, GitHub también vende repositorios privados e instancias locales de su software para empresas (Finley, 2012).

## 2.2. Presentación de la solución

Para la realización del clasificador de texto basado en un sistema de aprendizaje automático se sigue el procedimiento que se describe a continuación, ver Figura 2.1:

1. Preparar los datos del corpus para entrenar el modelo.
2. Limpieza y normalización de la información con el objetivo de reducir o eliminar aquellos datos que puedan influir de manera negativa en el resultado final.
3. Aplicar el método de clasificación creado a partir de los datos procesados.
4. Detección e interpretación de la polaridad del sentimiento.

[20] https://www.jetbrains.com/es-es/pycharm/
[21] https://github.com

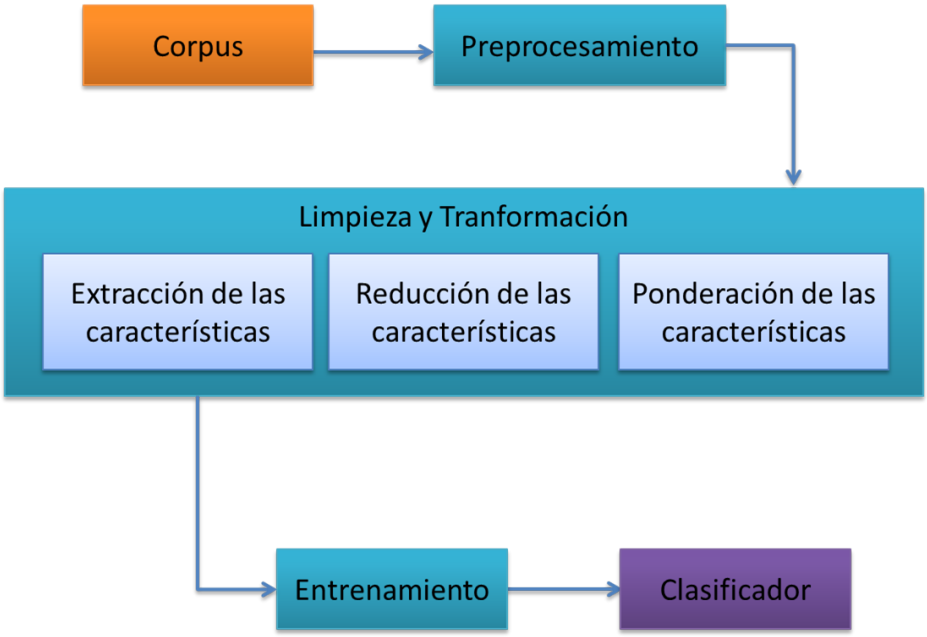



Figura 2.1: Fases de entrenamiento para el algoritmo de aprendizaje supervisado, elaboración propia.

## 2.3. Corpus

Para realizar un aprendizaje mediante métodos supervisados es necesario contar con un conjunto de pruebas representativas y previamente etiquetadas para poder entrenar el algoritmo de aprendizaje automático, es decir, un corpus o data. La mayoría de los trabajos realizados sobre la clasificación de mensajes se han hecho sobre Twitter con corpus que se encuentran en inglés y los que se realizan en español toman como referencia los corpus pertenencientes al Taller de Análisis de Sentimientos (TASS[22]) de la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN[23]).

Para desarrollar el objeto de estudio de la investigación, se decide crear un corpus con mensajes seleccionados y etiquetados de acuerdo con las características del Jabber, proceso que fue realizado por una sola persona. Para obtener mejores resultados se hace necesario disponer de un equipo de clasificación para disminuir el tiempo y el costo de la tarea.

El total de mensajes del corpus creado se encuentran clasificados en tres categorías de sentimientos: Positivo, Negativo y Neutro, como se muestran a continuación:

Aunque existen otras categorías que se derivan de positivo, negativo y neutro, entre la que se puede encontrar la categórica de sentimiento, Sin sentimiento (NONE); para el estudio esta categoría sería igual al de un mensaje clasificado como Neutro. En Sobrino Sande (2018) se puede ver la diferencia entre un mensaje Neutro y uno Sin sentimiento.

[22] http://tass.sepln.org/
[23] http://www.sepln.org/

Tabla 2.1: Ejemplos de mensajes

| | **Mensajes Positivos** | **Mesajes Negativos** | **Mensajes Neutros** |
|---|---|---|---|
| | *"ya casi esty salien"* | *"nada de eso"* | *"hola como estas"* |
| | *"ahh q ricoooooo"* | *"y toy aburrida y sin ganas de nadaaaa"* | *"q bola"* |
| | ... | ... | ... |
| | *"si ya lo vi claro"* | *"jajajaj el jabber no me gusta"* | *"dime veo"* |
| Total | 2339 | 1900 | 2606 |

## 2.4. Preprocesamiento

Este proceso es fundamental cuando se refiere a los mensajes extraídos del Jabber ya que es muy común encontrar mensajes con abreviaturas para referirnos a diversos contenidos, repeticiones de caracteres, faltas de ortografías, uso de la jerga, mezcla de letras mayúsculas y minúsculas.

En este trabajo seleccionaremos algunas de las reglas utilizadas por Sobrino Sande (2018):

- **Normalización de mayúsculas y minúsculas:** los algoritmos de aprendizaje automático tratan con diferentes significado "uci" y "UCI", aunque tenga igual significado para las personas.
- **Tratamiento de la duplicidad de caracteres:** se reduce a dos caracteres toda secuencia que tenga más de dos caracteres iguales, debido a que en el idioma español existen: cc, ll y rr.
- **Eliminación de tildes:** para no perder una relación semántica, ya que los algoritmos toman como palabras distintas "presentación" y "presentacion".
- **Eliminación de números y de caracteres especiales:** por lo general, los números y caracteres especiales no infieren en la polaridad del mensaje.
- **Eliminación de retornos de carro:** no es más que las eliminaciones de los saltos de líneas en los mensajes.
- **Eliminación de risas:** las múltiples y distintas formas de representación de la risa, "jjj", "jajaj", "jejej", "ggg", "jojo", "jajaajjaaa".
- **Normalización de la jerga:** el lenguaje informal utilizado frecuentemente en estos mensajes instantáneos, debido a escribir en un menor tiempo posible, ejemplo *k* en lugar de *que*, *tb* en lugar de *también*, entre otros.
- **Eliminación de enlaces**: al igual que los números y caracteres especiales los enlaces no suelen aportar sentimiento al mensaje.

## 2.5. Limpieza y transformación de los datos

En todo método que se haga uso de algoritmos de aprendizaje automático es necesario tratar previamente los datos con los que serán entrenados. El objetivo de esta fase es limpiar y normalizar la información para evitar que determinados datos puedan influir de manera negativa en el resultado final (Sobrino Sande, 2018).

### 2.5.1. Tokenización

Una vez completado el proceso de normalización de los mensajes del corpus, la siguiente etapa es la denominación *tokenización*. En esta fase los textos se dividen en unidades más pequeñas llamadas *tokens* y que normalmente se corresponden con las palabras de cada texto. Este proceso puede ser tan sencillo como separar los términos de las frases por los espacio en blanco y los caracteres de puntuación, considerar que la agrupación de determinados símbolos puede contener algún tipo de información que sea útil al proceso de clasificación. Este podría ser el caso de los emoticonos (del ingles *emoticon*), secuencias de caracteres de puntuación que suelen ser un indicador de la polaridad del sentimiento de las palabras a las que acompañan (Sobrino Sande, 2018).

### 2.5.2. Extracción de las características

A partir de los *tokens* obtenidos en el paso anterior, se definirá la manera de representar con ellos los mensajes de los que proceden, creando así las llamadas características. Lo habitual en la tarea de clasificación de textos es hacer uso del modelo de bolsa de palabras (Bow del inglés *Bag or Words*) en donde cada mensaje se representa mediante sus *tokens* sin tener en cuanta ningún orden concreto entre ellos. Esta bolsa de palabras puede contener *unigramas*, es decir, *tokens* independientes, *bigramas*, formados por la concatenación de dos *tokens* preservando el orden original de que estos tenían dentro del mensaje del que proceden, *trigramas*, etc (Sobrino Sande, 2018). En este estudio, las características serán *unigramas* o *tokens* individuales.

### 2.5.3. Reducción de las características

Esta etapa es opcional y su objetivo es disminuir el número de características del corpus mediante la eliminación de determinados *tokens* o de su conversión buscando una misma manera de representarlos. Existen tres técnicas habituales para llevar a cabo esta tarea: eliminación de *stopwords*, lemantización y *stemming* (Sobrino Sande, 2018).

- **Eliminación de stopwords:** es cuando existe un conjunto de palabras que, aunque son necesarias para construir oraciones con sentido, carecen de información que ayuden a determinar la polaridad de los textos en los que se encuentran. En español estas palabras son las preposiciones, los pronombres, las conjunciones y las distintas formas del verbo haber, entre otras. Mediante esta técnica, todos los términos pertenecientes a la lista de *stopwords* serán eliminadas del modelo antes del entrenamiento de los algoritmos.
- **Lemantización:** este es un proceso de normalización morfológica que transforma cada palabra en un lema mediante el uso de diccionarios y de un proceso de análisis morfológico. A modo de ejemplo, la lemantización convertiría la palabra “guapas” a su lema “guapo”. Por tanto, muchas características tomarían la misma forma, reduciendo así su variabilidad.
- **Stemming:** se trata de otro método de normalización morfológica pero más agresivo que la lemantización. En este caso, una palabra se transforma a su raíz por medio de la supresión de sus sufijos e inflexiones. Siguiendo el ejemplo anterior, la palabra “guapas” se convertiría a su raíz, “guap”.

### 2.5.4. Ponderación de las características

Las características extraídas en las etapas anteriores pueden ser consideradas todas de igual importancia y otorgarles distintos pesos en función de algún tipo de criterio. Aunque existen múltiples métodos de ponderación, hay cuatro modelos muy populares en la clasificación de textos y que tienen su origen en el campo de la Recuperación de la Información. Dichos pesos determinan la relevancia de cada característica dentro del mensaje al que pertenecen y, por lo tanto, influyen a la hora de clasificar los textos por parte de los algoritmos de aprendizaje supervisado (Sobrino Sande, 2018):

- **Ponderación binaria:** (BTO del inglés *Binary Term Ocurrences*) dada una lista con todas las características de todos los mensajes del corpus de entrenamiento, para cada mensaje se indicará con un valor 1 aquellas características que formen parte del mismo, y con un 0 en caso contrario.
- **Frecuencia absoluta:** (TO del inglés *Term Ocurrences*) en este caso, cada característica tendrá un peso igual al número de veces que aparece en un mensaje dado.
- **Frecuencia relativa:** (TF del inglés *Term Frequency*) este modelo de ponderación es igual al anterior, pero el valor a cada característica se le aplica un proceso de normalización Euclidea[24] que tiene en cuenta el número de características del mensaje al que pertenecen y sus frecuencias absolutas.
- **Esquema TF-IDF:** (del inglés *Term Frequency-Inverse Document Frequency*) este método otorga una mayor importancia a aquellas características que aparecen un mayor número de veces en el corpus, pero en pocos mensajes del mismo. Estos términos son los que suelen ayudar a identificar con mayor facilidad las distintas clases existentes. De esta forma, se evitan los problemas que implica el uso de la frecuencia absoluta o relativa en donde las características más repetidas son las que tiene mayor importancia independientemente del tipo de mensajes en el que aparezca. TF-IDF[25] es una medida muy utilizada en la tarea de clasificación de textos y en el campo de la Recuperación de la Información.

## 2.6. Entrenamiento

En el aprendizaje supervisado, los algoritmos trabajan con datos “etiquetados” (*labeled data*), intentado encontrar una función que, dadas las variables de entrada (*input data*), les asigne la etiqueta de salida adecuada. El algoritmo se entrena con un “histórico” de datos y así “aprende” a asignar la etiqueta de salida adecuada a un nuevo valor, es decir, predice el valor de salida (Simeone, 2018).

Después de realizadas las etapas anteriores y la ponderación de las características en función de la importancia que se les quiera dar, se pasa al entrenamiento de los clasificadores.

### 2.6.1. Algoritmos de clasificación

Los algoritmos de aprendizaje supervisado se pueden dividir principalmente en dos grandes categorías: regresión y clasificación. Los primeros permiten predecir un atributo de valor continuo asociado

[24] https://es.wikipedia.org/wiki/Norma_vectorial
[25] https://es.wikipedia.org/wiki/Tf-idf

a un objeto, por ejemplo, respuesta a precios de medicamentos y precios de acciones. En cambio, los de clasificación se utilizan para identificar a qué categoría pertenece un objeto, por ejemplo, detección de spam y reconocimiento de imágenes (sci, 2020). En esta investigación se seleccionaron cuatro algoritmos de clasificación: Máquina de Vectores de Soporte, Naive Bayes, K vecinos más cercanos y Árboles de Decisión. Aunque los de regresión se han usado en múltiples ocasiones en el análisis de sentimientos.

#### 2.6.1.1. Máquinas de Vectores de Soporte

Las Máquinas de Vectores de Soporte (SVM del inglés *Support Vector Machines*) son un grupo de algoritmos de aprendizaje, que tienen muchas cualidades deseables que lo convierten en uno de los más populares algoritmos. No solo tiene una base teórica sólida, sino que también tiene una clasificación de las más precisa que la mayoría de los otros algoritmos en muchas aplicaciones, especialmente aquellas aplicaciones que involucran datos dimensionales muy altos. Por ejemplo, varios investigadores han demostrado que SVM es quizás el algoritmo más preciso para la clasificación de texto. También es ampliamente utilizado en clasificación de páginas web y aplicaciones bioinformáticas (Liu, 2007).

El principio fundamental del SVM es determinar separadores lineales en el espacio de búsqueda que separen mejor las diferentes clases (Medhat et al., 2014). En la Figura 2.2, hay 2 clases (-1), (1) y un *hiperplano*, denominado vector de soporte, que proporciona la mejor separación posible en base a su clase. De esta forma, el vector determina la frontera que sirve para clasificar un nuevo elemento, por lo que dependiendo a qué parte del espacio pertenezca, se le asignará una clase u otra.

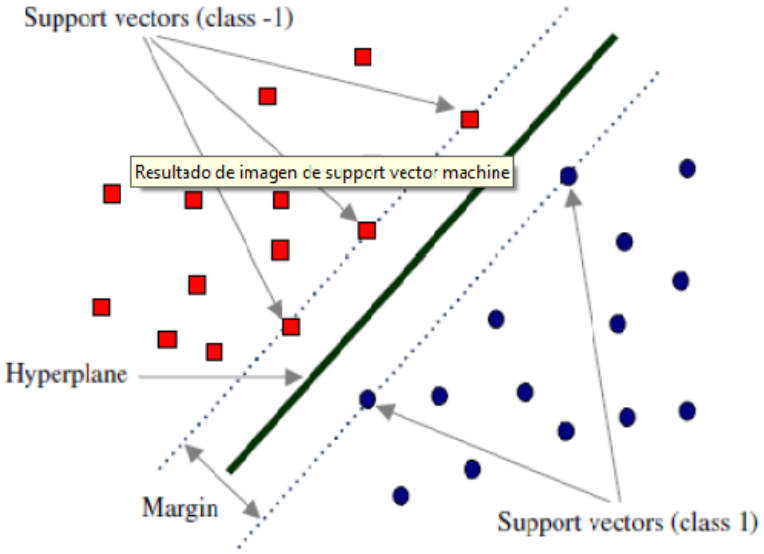



Figura 2.2: Máquina de Vectores de Soporte, tomado de Sobrino Sande (2018)

Este tipo de algoritmos cuenta con una serie de parámetros que permiten ajustar su configuración interna y así optimizar los resultados durante el proceso de clasificación. Uno de estos parámetros es el *kernel* y se utiliza cuando no es posible separar las muestras mediante una línea recta, plano o hiperplano de N dimensiones, permitiendo tal separación mediante otro tipo de funciones matemáticas. Otro de estos parámetros es *regularización* (también conocido como “C”) que permite crear un margen blan-

do de manera que se consientan ciertos errores en la clasificación y este evite el sobreentrenamiento (del inglés *overfitting*). Para terminar, el parámetro *gamma* determina la distancia máxima a partir de la cuál una muestra pierde su influencia en la configuración del vector de soporte, y *margin*, que es la separación entre el vector y las muestras de cada clase más cercanas al mismo.

#### 2.6.1.2. Clasificador de Naive Bayes

El clasificador Naïve ve Bayes es el clasificador más simple y más utilizado. El modelo de clasificación de Naive Bayes calcula la probabilidad posterior de una clase, en función de la distribución de las palabras en el documento. El modelo funciona con la extracción de características BOW que ignora la posición de la palabra en el documento. Utiliza el teorema de Bayes para predecir la probabilidad de que un conjunto de características dado pertenezca a una etiqueta en particular.

$$P(etiqueta|caracteristica) = \frac{P(etiqueta) * P(caracteristica|etiqueta)}{P(caracteristica)} \tag{2.1}$$

*P(etiqueta)* es la probabilidad de que una característica aleatoria establezca la etiqueta. *P(características | etiqueta)* es la probabilidad previa de que un conjunto de características determinado se clasifique como una etiqueta. *P(características)* es la probabilidad previa de que se produzca un conjunto de características dado (Medhat et al., 2014).

Los algoritmos Naive Bayes suelen recibir el apelativo de “ingenuos” debido a que en sus cálculos las características seleccionadas para representar a los ejemplos de entrenamiento son estadísticamente independientes y contribuyen por igual en el proceso de clasificación. Dicho de otro modo en el caso concreto de la clasificación de textos, se considera que las palabras de un mismo mensaje no mantiene ningún tipo de relación entre si y es diferente la posición que tiene dentro del texto al que pertenecen (Sobrino Sande, 2018), como se puede ver en la Figura 2.3.

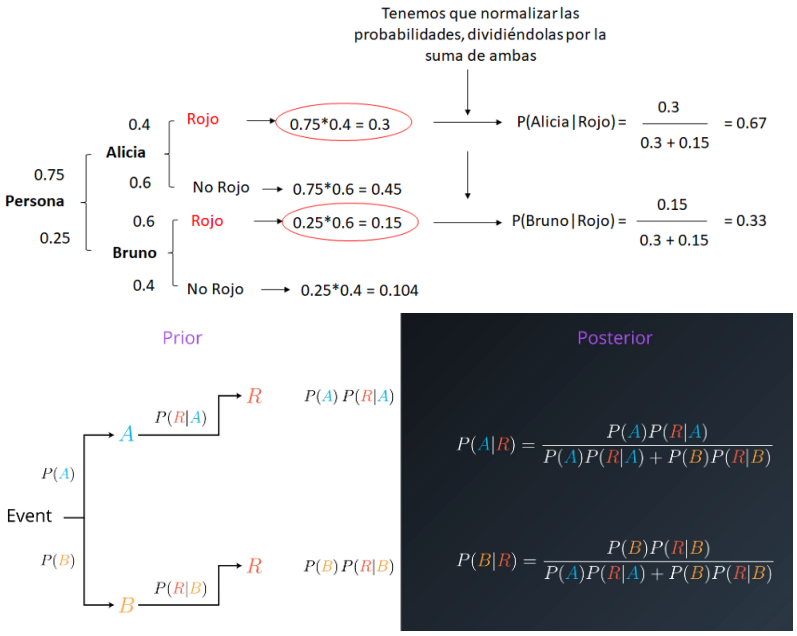



Figura 2.3: Naive Bayes, tomado de Roman (2019)

#### 2.6.1.3. Árboles de decisión

El aprendizaje a través de árboles de decisión es una de las técnicas más utilizadas para la clasificación de textos, su precisión de clasificación es competitiva y muy eficiente con otros métodos de aprendizaje automático. El clasificador de árbol de decisión proporciona una descomposición jerárquica del espacio de datos de entrenamiento en el que se usa una condición en el valor del atributo para dividir los datos. La condición o predicado es la presencia o ausencia de una o más palabras. La división del espacio de datos se realiza de forma recursiva hasta que los nodos hoja contienen ciertos números mínimos de registros que se utilizan con el fin de clasificarlos (Medhat et al., 2014).

Su estructura es la de un grafo dirigido en forma de árbol compuesto por un conjunto de reglas extraídas a partir de las características de los datos de entrenamiento y que se aplican de manera sucesiva a la hora de predecir a que clase pertenece un nuevo ejemplo (Sobrino Sande, 2018).

El árbol está conformado por varios tipos de nodos: nodos de decisión que especifican alguna decisión de un atributo, nodos hoja que indica la clase correspondiente y el nodo raíz que es por donde se empieza el recorrido. En cada nodo que posee el árbol se escoge un atributo de los datos que discrimina de mejor manera el conjunto, dividiéndolo así en subconjuntos pertenecientes a una clase u otra, de esta forma, una vez entrenado los datos se procede a clasificar los nuevos datos a partir de las decisiones que tenga que ir tomando en cada nodo, llegando así a determinar a qué clase debe pertenecer (Oliva, 2014), como se muestra en la Figura 2.4:

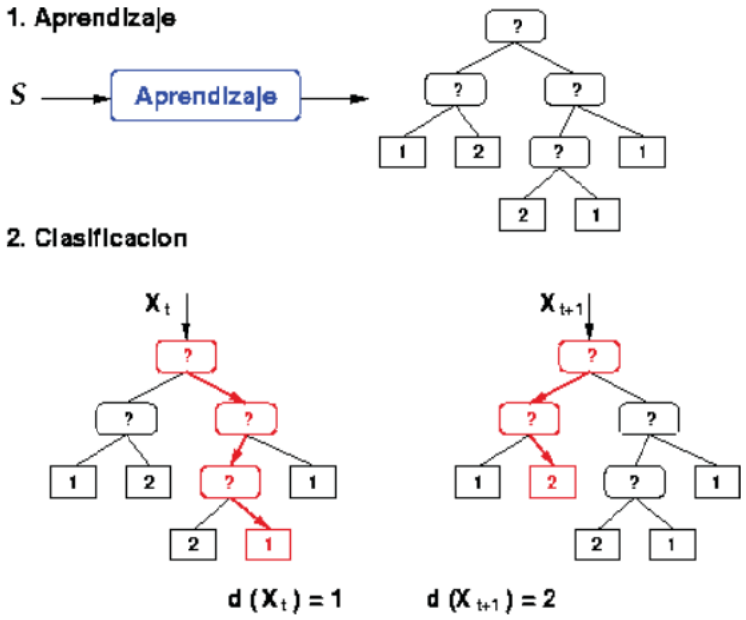



Figura 2.4: Árbol de decisión con un esquema del algoritmo C4.5, tomado de Oliva (2014)

#### 2.6.1.4. K vecinos más cercanos

Todos los métodos de aprendizaje anteriores aprenden de algún tipo de modelos de datos de entrenamiento, a estos métodos de aprendizaje se le llaman métodos de aprendizaje ansiosos porque aprenden de modelos de datos antes de la prueba. En contraste, *k-vecino* (k-NN del inglés *k-Nearest Neighbord*)

es un método de aprendizaje vago en el sentido de que no crea ningún modelo a partir de los datos de entrenamiento. El aprendizaje solo ocurre cuando un ejemplo de prueba necesita ser clasificado. La idea de k-NN es extremadamente simple y, sin embargo, bastante eficaz en muchas aplicaciones, por ejemplo, clasificación de texto.

El componente clave de un algoritmo k-NN es la función distancia/similitud, que se elige en función de las aplicaciones y la naturaleza de los datos. El número de vecinos *k* más cercanos generalmente se determina mediante el uso de un conjunto de validación o mediante la validación cruzada de los datos de entrenamiento. Es decir, se prueba un rango de valores k y se selecciona el valor k que proporciona la mejor precisión en el conjunto de validación, ver Figura 2.5, (Liu, 2007).

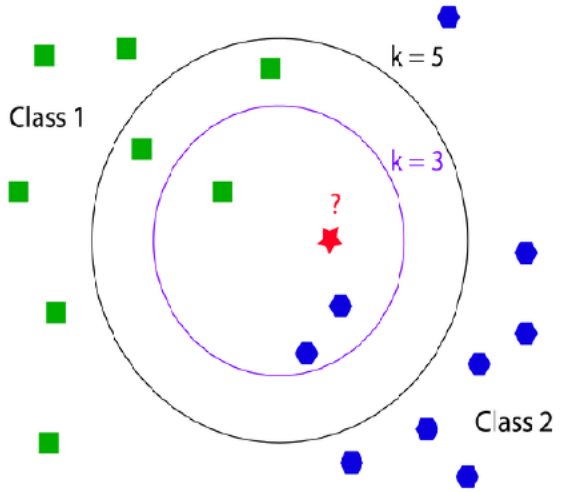



Figura 2.5: Importancia de elegir el k correcto, tomado de Sobrino Sande (2018)

## 2.7. Clasificador línea de base

Para la realización del clasificador base se empleó una combinación de varias técnicas para mejorar su rendimiento. Estas combinaciones están formadas por los siguientes elementos:

- **Algoritmos de aprendizaje supervisado:** aplicando SVM con *kernel lineal*, Naive Bayes, Árboles de Decisión implementado con el algoritmo CART y KNN con *k=30* .
- **Preprocesamiento del corpus:** utilización de todas las reglas descritas anteriormente midiendo por separado el rendimiento de los modelos con la normalización de los elementos de los mensajes y su eliminación.
- **Reducción de las características:** mediante las combinaciones de reducción del espacio vectorial: sin ninguna técnica, solo aplicando la eliminación de los *stopwords*, solo mediante *stemming* y con ambas a la vez.
- **Ponderación de las características:** para obtener los modelos con cada una de las cuatro técnicas presentadas: ponderación binaria, frecuencia absoluta, frecuencia relativa y TF-IDF.

## 2.8. Propuesta de la solución

Partiendo de los resultados obtenidos en la sección anterior se crea un método con todas las técnicas que ofrecen valores de mejora positivos y después de varios experimentos se ha concluido con un valor de C=0.5 para el algoritmo de clasificación y una reducción de características mediante la técnica *k highest scores* con $k = 500$ y la función de selección $X^2$ (chi cuadrado).

La forma más simple que se puede usar para evaluar el rendimiento de un algoritmo de aprendizaje automático es emplear diferentes conjuntos de datos de entrenamiento y prueba. Es decir, tomar el conjunto de datos original y dividirlo en dos partes. Entrenar el algoritmo en la primera parte, hacer las predicciones con la segunda parte y evaluar las predicciones contra los resultados esperados (eva, 2020; res, 2020).

El tamaño de la división puede depender de la cantidad y los detalles del conjunto de datos, aunque es común usar el 67 % de los datos para entrenamiento y el 33 % restante para pruebas. Como el conjunto de datos utilizados no es tan grande se usó una división de 80 y 20, como se puede ver en la Figura 2.6, permitiendo ver un equilibrio entre los mensajes clasificados como neutro, positivo y negativo como se muestra en la Figura 2.8.

Esta técnica de división de algoritmos es muy rápida, pero entre sus desventajas se encuentra tener una gran variación. Lo que significa que las diferencias en el conjunto de datos de entrenamiento y prueba pueden resultar con diferencias significativas en la estimación de la precisión. Para evitarlo se especifica una semilla aleatoria[26], ver Figura 2.7, para asegurar de obtener los mismos números aleatorios cada vez que se ejecuta el código, consultar en (Tra, 2020).

```
train, test = DatasetHelper.generate_train_test_subsets(data, size=0.80)
```

Figura 2.6: Fragmento de código para la división del corpus para test y train

```
codes_train, codes_test, labels_train, labels_test = train_test_split(codes,
    labels, train_size=size,random_state=42)
```

Figura 2.7: Fragmento de código de asignación de semilla aleatoria

A partir de lo planteado se obtiene un 72.56 % de exactitud, ver los resultados en la Tabla 2.2.

## 2.9. Mejora del método

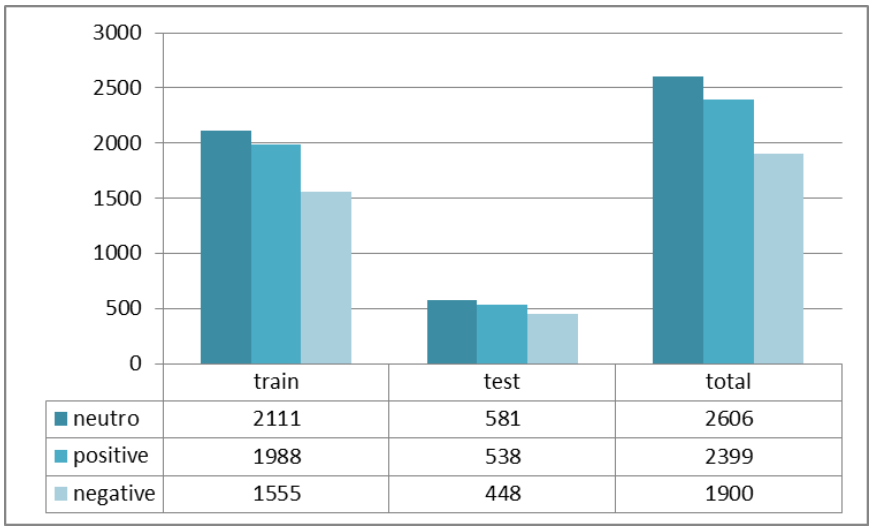
Partiendo del método descrito en la sección anterior, se tratará de mejorar los resultados añadiéndole dos características. Estas características se confeccionaron con varios métodos extraídos de distintos trabajos de investigación:

- **Símbolos de sentimiento:** los usuarios de la red de mensajería instantánea acostumbran añadir emoticonos o emojis a sus mensajes y éstos son un indicador inequívoco en la polaridad del sentimiento de sus palabras (Wang y Castanon, 2015). Para comprobar si estos símbolos ayudan al

[26]https://scikit-learn.org/stable/glossary.html#term-random-state

proceso de clasificación de textos, se incluirá como característica el número de símbolos positivos, negativos y neutros que contienen cada mensaje. Para ello, se han creado listas distintas de símbolos, una para cada tipo de sentimiento, a partir de emoticonos y otros emojis utilizados por Sobrino Sande (2018), aunque estos se pueden extraer de Internet[27].

- **Lexicón de palabras con sentimiento:** un recurso clásico y muy usado en la clasificación de textos son los diccionarios de palabras etiquetados con su polaridad, normalmente positiva o negativa. Así, para cada texto se buscará el número de palabras positivas y negativas que contienen y esta información se añadirá como característica asociada a cada mensaje. Para el modelo, se ha confeccionado un lexicón mediante la unión de otros ya existentes: iSOL[28](Molina-González et al., 2013) y Spanish Sentiment Lexicon[29](Perez-Rosas et al., 2012).




| | train | test | total |
|---|---|---|---|
| neutro | 2111 | 581 | 2606 |
| positive | 1988 | 538 | 2399 |
| negative | 1555 | 448 | 1900 |

Figura 2.8: Representación del total de datos empleados, creación propia

Tabla 2.2: Reporte de clasificación

| | precision | recall | f1_score | support |
|---|---|---|---|---|
| negative | 0.7484 | 0.6585 | 0.7134 | 448 |
| neutro | 0.7254 | 0.7728 | 0.7483 | 581 |
| positive | 0.6907 | 0.7305 | 0.7100 | 538 |
| accuracy | | | 0.7256 | 1567 |
| macro avg | 0.7315 | 0.7206 | 0.7239 | 1567 |
| weigthed avg | 0.7286 | 0.7256 | 0.7252 | 1567 |

0.7255902999361837

[27] Extraer en Emojis: https://unicode.org/emoji/charts/full-emoji-list.html
Emoticonos: https://en.wikipedia.org/wiki/Emotion_classification#Lists_of_emotions
[28] http://timm.ujaen.es/recursos/isol/
[29] http://web.eecs.umich.edu/ mihalcea/downloads.html#SPANISH_SENT_LEXICONS

Tabla 2.3: Reporte de clasificación de las nuevas características

| | Solo con lexicón de palabras | | | | Solo con simbolos de sentimientos | | | |
|---|---|---|---|---|---|---|---|---|
| | precision | recall | f1_score | support | precision | recall | f1_score | support |
| negative | 0.7748 | 0.6987 | 0.7374 | 448 | 0.7819 | 0.6562 | 0.7136 | 448 |
| neutro | 0.7321 | 0.7762 | 0.7536 | 581 | 0.7258 | 0.7745 | 0.7493 | 581 |
| positive | 0.7166 | 0.7286 | 0.7226 | 138 | 0.6918 | 0.7342 | 0.7124 | 538 |
| accuracy | | | 0.7377 | 1657 | | | 0.7269 | 1567 |
| macro avg | 0.7412 | 0.7312 | 0.7370 | 1657 | 0.7332 | 0.7217 | 0.7251 | 1567 |
| weigthed avg | 0.7390 | 0.7377 | 0.7375 | 1657 | 0.7302 | 0.7269 | 0.7264 | 1567 |
| | 0.7377153797064454 | | | | 0.7268666241225271 | | | |

Tabla 2.4: Reporte de clasificación del modelo

| | Clasificador base | | | | Union de las caracteristicas | | | |
|---|---|---|---|---|---|---|---|---|
| | precision | recall | f1_score | support | precision | recall | f1_score | support |
| negative | 0.7484 | 0.6585 | 0.7134 | 448 | 0.7753 | 0.7009 | 0.7362 | 448 |
| neutro | 0.7254 | 0.7728 | 0.7483 | 581 | 0.7330 | 0.7797 | 0.7556 | 581 |
| positive | 0.6907 | 0.7305 | 0.7100 | 538 | 0.7188 | 0.7268 | 0.7227 | 538 |
| accuracy | | | 0.7256 | 1567 | | | 0.7390 | 1567 |
| macro avg | 0.7315 | 0.7206 | 0.7239 | 1567 | 0.7424 | 0.7358 | 0.7382 | 1567 |
| weigthed avg | 0.7286 | 0.7256 | 0.7252 | 1567 | 0.7402 | 0.7390 | 0.7388 | 1567 |
| | 0.7255902999361837 | | | | 0.7389917038927888 | | | |

Como se puede apreciar, en la Tabla 2.3, estas nuevas características mejora los resultados del clasificador. Los diccionarios de palabras catalogadas por sentimiento ayudan de manera importante en la tarea de clasificación de textos, en este caso el valor de rendimiento aumentó de 0.7252 a 0.7375 con respecto al valor-f1 y de un 0.7256 a 0.7377 con respecto a la exactitud. Mientras que aplicando la característica usando solo los símbolos de sentimientos no tienen una gran mejora, solo de un 0.7252 a 0.7269 para el valor-f1 y de 0.7256 a 0.7269 para la exactitud. Pero aplicando ambas características se obtiene una gran mejora del clasificador, como se muestra en la Tabla 2.4.

El porcentaje de exactitud del método final es de un 73.90 %. Pudiese parecer que el porcentaje no es lo suficiente alto para afirmar que el sistema posee un buen rendimiento, pero no es así. En el análisis de sentimientos se considera que un buen sistema presenta un buen nivel de precisión cuando alcanza un valor del 70 % de acierto, ver en Ogneva (2010); Roebuck (2012).

## 2.10. Conclusiones del capítulo

En este capítulo se propuso un método para la detección de polaridad de sentimientos en los mensajes extraídos de la red de mensajería de la UCI. Luego de presentada la propuesta se obtienen las conclusiones siguientes:

1. El KDD se utiliza en la propuesta de solución para identificar los patrones válidos dentro de la fuente de datos y eliminar o corregir los datos incorrectos.
2. La implementación de la solución diseñada se obtuvo a partir de los algoritmos para la clasificación de polaridad en documentos según los mensajes extraídos del Jabber.
3. La obtención de la solución para la detección de la polaridad de sentimientos en los mensajes extraídos del Jabber, posibilitó incorporar un nuevo instrumento para el análisis de sentimientos en las opiniones y comentarios.
4. La polaridad obtenida a partir de los mensajes facilitó el estudio y comprensión del comportamiento de los comentarios y opiniones generados por los usuarios en las conversaciones de la red.

# Capítulo 3

# Validación del método implementado

En este capítulo se describe la validación del método desarrollado mediante varios criterios para determinar que la solución propuesta responde satisfactoriamente a los objetivos que se trazaron. Se mide la precisión del algoritmo de clasificación a partir de las métricas de evaluación que comprueban la eficiencia con que se ejecutan y la calidad de las soluciones que éstos producen. Al concluir se realiza un análisis de los resultados con datos reales.

## 3.1. Métricas de clasificación y medidas de evaluación de los resultados

Para evaluar la precisión de predicción de los algoritmos y la efectividad de clasificación, se emplean diferentes medidas capaces de brindar informaciones objetivas referentes al desempeño de la clasificación a partir de los datos proporcionados.

Sobrino Sande (2018) explica que para entender los cuatro estados de un ejemplo a clasificar, se debe tomar una clase A y un algoritmo que determina si dicho ejemplo pertenece o no a esa clase:

- **True Positives:** (Verdaderos Positivos o TP) son los ejemplos que han sido marcados de manera correcta como perteneciente a la clase A.
- **False Positives:** (Falsos Positivos o FP) serán los ejemplos marcados como de clase A, pero en realidad no pertenecen a ella, es decir, han sido clasificados de manera incorrecta.
- **True Negatives:** (Verdaderos Negatives o TN) en este caso, los ejemplos no son de la clase A y han sido clasificados correctamente.
- **False Negative:** (Falsos Negativos o FN) en este grupo estarán los ejemplos marcados como no pertenecientes a la clase A, pero en realidad sí lo son y, por tanto, no se han clasificado correctamente.

Teniendo en cuenta los estados anteriores, representados en la matriz de confusión, ver Tabla 3.1, se definen las siguientes medidas que serán usadas para evaluar los métodos de clasificación:

- **Exactitud:** Esta métrica es la número uno por ser la más sencilla e intuitiva, que simplemente indica la relación del número de elementos clasificados correctamente en comparación con el número total de observaciones.

Tabla 3.1: Matriz de Confusión

| | Clasificado como clase A | Clasificado como clase B |
|---|---|---|
| Real A | Verdaderos Positivos (TP) | Falso Negativo (FN) |
| Real B | Falso Positivo (FP) | Verdadero Negativo (VN) |

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.1}$$

Si se tiene una alta exactitud como resultado, no significa que el método sea el mejor, hay que tener en cuenta que esta métrica es ideal solo cuando los datos están balanceados, es decir que el número de elementos de cada clase sea aproximadamente el mismo y el corpus esté balanceado (Sobrino Sande, 2018). En caso contrario, es necesario hacer uso de otro tipos de medidas como la precisión, la exhaustividad y el valor-F que ofrecerán valores distintos para la clase A y para la B.

- **Precisión:** es la razón entre el número de documentos clasificados correctamente como pertenecientes a la clase A y el número total de documentos de que han sido clasificados por el modelo como clase A.

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

La precisión es una medida de cuántas predicciones positivas fueron observaciones positivas reales. La alta precisión se relaciona con la baja tasa de falsos positivos.

- **Exhaustividad/Sensibilidad:** es la relación entre los documentos clasificados correctamente como pertenecientes a la Clase A y la suma de todos los documentos de clase A.

$$Recall = \frac{TP}{TP + FP} \tag{3.3}$$

La cobertura es la proporción de los elementos positivos reales identificados acertadamente. También se puede ver como la capacidad que tiene el modelo de construir de manera correcta las clases. Cuanto más cercano a 1, mejor estarán definidas las distintas clases existentes ya que su valor aumenta a medida que disminuya el número de falsos negativos.

- **Valor-F:** representa la media aritmética entre las medidas de precisión y sensibilidad y suele utilizare como referencia para comprar el rendimiento entre varios modelos. La fórmula del valor-F combina las dos medidas anteriores de manera ponderada a través de un parámetro $\beta$ lo que permite otorgar una mayor importancia a una que otra:

$$F_\beta = (1 + \beta^2)\frac{Precision * Recall}{(\beta^2) + Recall} \tag{3.4}$$

Es frecuente que la precisión y la exhaustividad tengan el mismo peso en la fórmula, es decir, con un valor $\beta = 1$. Esta configuración se le conoce como Valor-$F_1$ o $F_1$-score (Sobrino Sande, 2018). La mejor puntuación $F_1$ es igual a 1 y la peor a 0.

En caso de que solo se tenga dos clases de respuestas, se puede categorizar como un problema de clasificación binaria, pero en el caso de que se tenga más de dos clases, como el sistema presentado, se debe calcular cada una de las métricas anteriormente explicadas por cada clases y combinarlas entre ellas para poder obtener una media global. Para ello, existen tres posibilidades:

- **Macro-averaging:** en este caso se calculan las medidas de cada clase y a continuación, calcular la media aritmética:

$$Macro - Precision = \frac{\sum_{i=1}^{n} Precision_i}{n} \tag{3.5}$$

$$Macro - Recall = \frac{\sum_{i=1}^{n} Recall_i}{n} \tag{3.6}$$

  El problema es que estas medidas no tienen en cuenta la posible desigualdad entre el número de ejemplos de cada clase por lo que sus resultados pueden no ser fiables en ese tipo de escenarios.

- **Micro-averaging:** en este caso se tienen en cuenta el número de elementos de cada clase ya que hacen uso de todos los resultados para el cálculo de los indicadores:

$$Micro - Precision = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FP_i} \tag{3.7}$$

$$Micro - Recall = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FN_i} \tag{3.8}$$

  Las medidas *micro-averaging* sí tienen en cuenta la desigualdad entre el número de ejemplos de cada tipo, pero en caso de que el sistema cuente con más de dos clases distintas, se tiene que $\sum_{i=1}^{n} FP_i = \sum_{i=1}^{n} FN_i$ por lo que la precisión, exhaustividad y el valor-F1 toman siempre los mismos valores.

- **Weigthed-averaging:** para resolver el problema de tener un corpus desequilibrado y con más de dos clases distintas, la librería Scikit-Learn cuenta con un grupo de medidas adicionales basadas en las fórmulas *macro-averaging*. En ellas se ponderan cada componente de la fórmula en base al peso que cada clase tiene dentro del sistema global:

$$Weigthed - Precision = \frac{\sum_{i=1}^{n} Precision_i * P_i}{n} \tag{3.9}$$

$$Weigthed - Recall = \frac{\sum_{i=1}^{n} Recall_i * P_i}{n} \tag{3.10}$$

Estas medidas ponderadas serán las que determinan cuál de los métodos tiene el mejor rendimiento.

## 3.2. Entrenamiento y validación del método

El ajuste del algoritmo es un paso final en el proceso de aprendizaje automático aplicado antes de presentar los resultados. Es considerado un paso importante en el proceso de aprendizaje automático para mejorar el rendimiento del algoritmo justo antes de presentar los resultados o preparar un sistema para la producción.

Entrenar modelos de aprendizaje automático requiere dos tipos de parámetros: los que se aprenden de los datos y los del algoritmo. Los primeros pueden ser, por ejemplo, los parámetros de una regresión lineal, los vectores de soporte de un SVM. Los segundos son los parámetros de ajuste, también llamados hiperparámetros, como el número de vecinos en k-vecinos más cercanos (k-nn), la profundidad máxima en un árbol de decisión (ana, 2019a).

En los modelos de aprendizaje automático, los parámetros son las variables que se estiman durante el proceso de entrenamiento con los conjuntos de datos. Por lo que sus valores no los indica manualmente el científico de datos, sino que son obtenidos. Los parámetros son la parte más importante en los modelos de aprendizaje automático. Ya que es la parte de los modelos que se aprende de los datos. Son necesarios para realizar las predicciones. Además de definir la capacidad del modelo para resolver un problema dado. Por lo que estimarlos correctamente es una tarea clave (ana, 2019b).

Los hiperparámetros[30] de un modelo son los valores de las configuraciones utilizadas durante el proceso de entrenamiento. Son valores que generalmente se obtienen de los datos, por lo que suelen ser indicados por el científico de datos. El valor óptimo de un hiperparámetro no se puede conocer a priori para un problema dado. Por lo que se tiene que utilizar valores genéricos, reglas genéricas, los valores que han funcionado anteriormente en problemas similares o buscar la mejor opción mediante prueba y error. Siendo una buena opción buscar los hiperparámetros la validación cruzada (ana, 2019b).

Muchos de los modelos que se implementan en aprendizaje automático requieren que se fijen los valores de los hiperparámetros durante el entrenamiento. En la mayoría de las ocasiones, la selección de un valor u otro no es una tarea trivial. Pudiendo afectar de forma significativa a los resultados. Para la selección de estos valores se puede utilizar algunas de las herramientas disponibles en `scikit-learn`, como `GridSearchCV` o `RandomizedSearchCV`. Ambas clases permiten seleccionar los parámetros más apropiados para un modelo y un conjunto de datos utilizando la técnica de validación cruzada (ana, 2019b).

### 3.2.1. Validación cruzada

La validación cruzada (del inglés *croos-validation*)[31] es una técnica con la que se puede identificar la existencia de diferentes problemas durante el entrenamiento de los modelos, como la aparición de sobreajuste. Permitiendo así obtener modelos más estables.

La validación cruzada divide el número de muestras del corpus en conjuntos de igual tamaño de manera que *k-1* grupos son usados para entrenar el sistema y el grupo sobrante para su evaluación. Este proceso se repite *k* veces y en cada uno de ellos se escogerá un grupo diferente para validar la eficacia del modelo. En cada iteración se calcula el valor-F1 ponderado y se halla la media entre todos, esta media que servirá de referencia de su efectividad (Cro, 2020). La validación cruzada suele tomar los valores 3, 5 o 10 para el parámetro *k*, para este estudio el valor de *k* será 10.

Así es posible identificar si los modelos son inestables o estables, es decir, el resultado depende de los datos utilizados o no. En caso de que los resultados dependan de los datos utilizados, el modelo posiblemente estará siendo sobreajustado. Indicando que el modelo empleado dispone de demasiados

[30] hiperparámetro es un parámetro cuyo valor se establece antes de que comience el proceso de aprendizaje. Por el contrario, los valores de otros parámetros se derivan a través de la formación

[31] https://scikit-learn.org/stable/modules/cross_validation.html

grados de libertad (ana, 2019a).

Los resultados de las medidas de evaluación aplicadas mediante la validación cruzada, se pueden observar en la Figura 3.1, para una mejor comprensión de la misma se tomará el valor F como referencia del análisis, debido a que los resultados de la precisión y exhaustividad no tienen muchas variaciones entre los algoritmos. En la Figura 3.2, se muestran los mejores valores para los algoritmos probados, mostrando que las maquinas de vectores de soporte tiene mejores resultados y el peor ha sido k vecinos más cercanos.

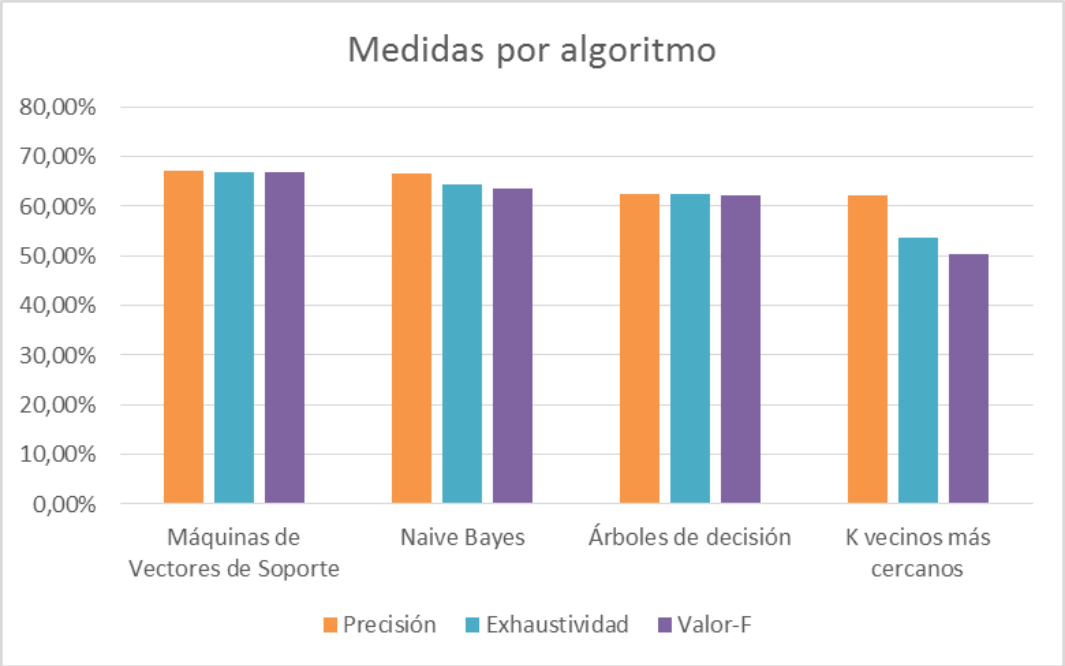



Figura 3.1: Medidas por algoritmo

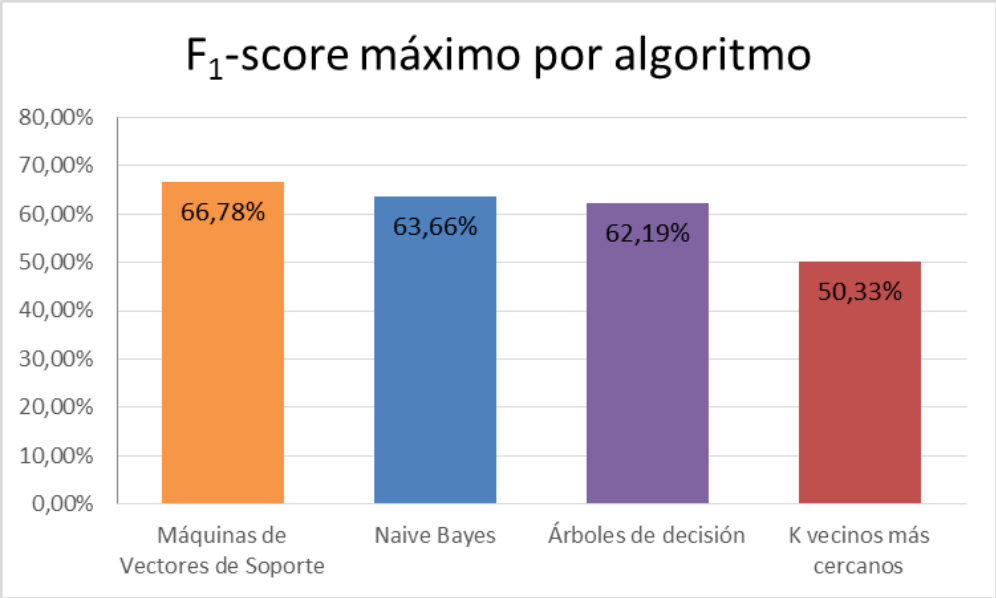



Figura 3.2: Valor $F_1$ máximo por algoritmo

En la Figura 3.3, se puede observar con mayor claridad la superioridad de algunos algoritmos sobre otros. La progresión de cada línea permite apreciar en que grado aumenta o disminuye el rendimiento de cada algoritmo con la regla aplicada. Los test 1-8 usan una ponderación BTO, en el intervalo 9-16 la ponderación es TO, las pruebas 17-24 tienen ponderación TF y entre los números 25-32 la ponderación es TF-IDF. En los 16 primeros test los algoritmos se comportan de manera estable, ya a partir de aquí la maquina de vectores de soporte empieza a aumentar su rendimiento mientras que k vecinos más cercanos empieza a disminuir.

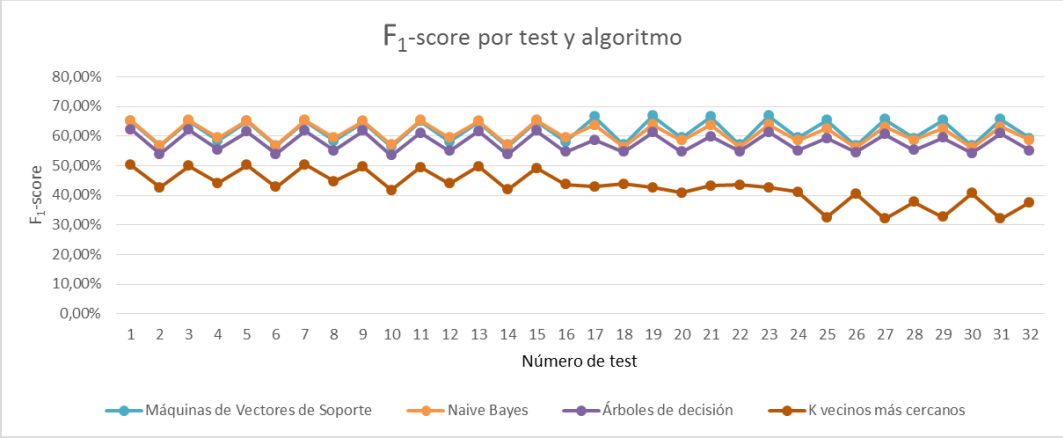



Figura 3.3: Valor $F_1$ por test y algoritmo

Estas variaciones son más fáciles de percibir en la Figura 3.4, donde se muestra el valor medio de la medida $F_1$ ponderada para los diferentes test agrupados por algoritmo y método de ponderación. Aquí Naive Bayes y K vecinos más cercanos tienen los mejores resultados con las ponderaciones BTO y TO. En caso contrario, las maquinas de vectores de soporte tienen mejor resultado para TF y TF-IDF.

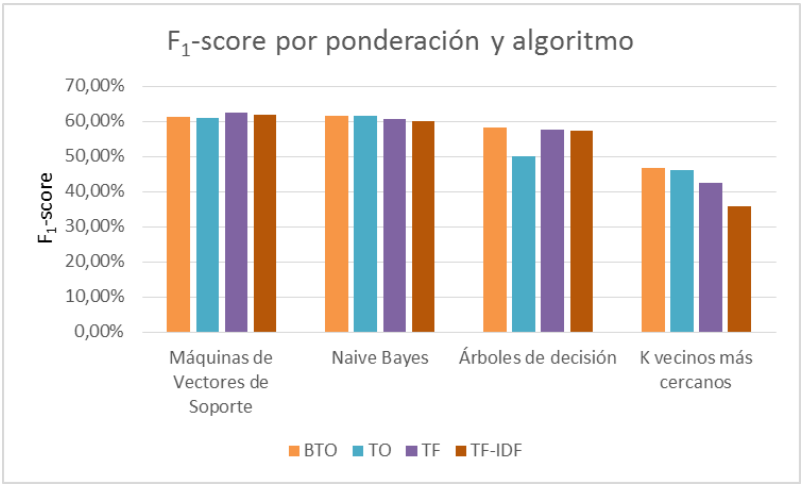



Figura 3.4: Valor $F_1$ por ponderación y algoritmo

En la siguiente Figura 3.5 se muestra los valores alcanzados para cada algoritmo y las cuatros combinaciones de reducción de las características. Todos los algoritmos presentan una mejora al no reducir ningunas de las características, estos resultados dependen del set de prueba.

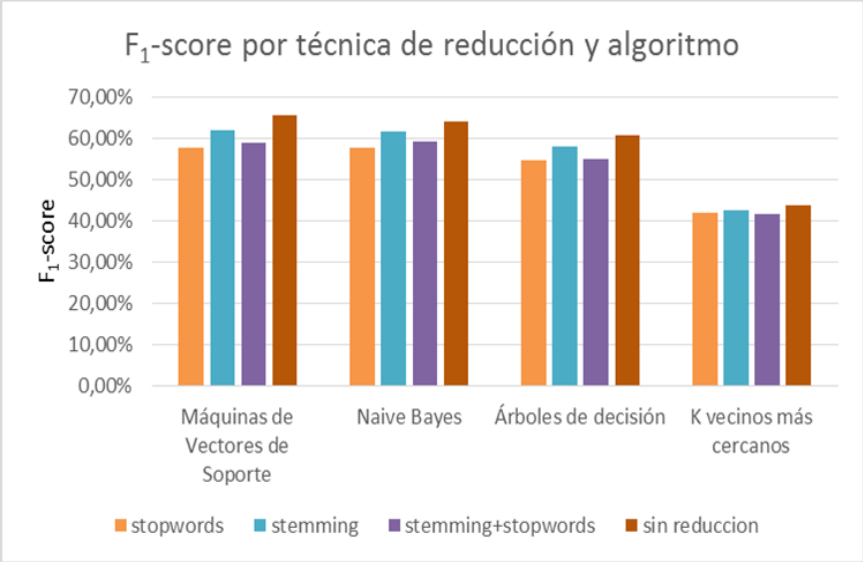



Figura 3.5: Valor $F_1$ por técnica de reducción y algoritmo

### 3.2.2. Selección de parámetros con GridSearchCV

Scikit-learn dispone de varias clases que implementan la metodología de la validación cruzada. En el caso de que se desee utilizar para seleccionar los parámetros de entrenamiento de un modelo una de las opciones es `GridSearchCV`. Siendo uno de los más simples y fáciles de utilizar. El constructor de esta clase se ha de llamar indicándole la instancia de un modelo, los valores a probar y el número de conjuntos en el que se dividen los datos. Esto se realiza mediante los siguientes parámetros (ana, 2019a):

- `estimator:` el método que se ha de evaluar.
- `param_grid:` un diccionario en que se indicar los parámetros a evaluar como clave y el conjunto elementos como valor.
- `cv:` el número de conjuntos en los que se divide los datos para la validación cruzada.

En las Figuras 3.6 y 3.7 se explican los componentes para la realización de la validación cruzada en el método creado con el clasificador base aplicando las métricas de evaluación a través de la clase `GridSearchCV`. Los resultados obtenidos se pueden ver a continuación: ver Figuras 3.8 y 3.9:

- **best_score:** Puntuación media de la validación cruzada del *best_estimator* (gri, 2020), que obtiene un valor de 0.6768, determina el grado de precisión del clasificador, en este caso seria de un 67.68 %.
- **best_parameter:** Configuración de parámetros brinda los mejores resultados en los datos (gri, 2020). Los mejores parámetros obtenidos son, como mejor clasificador el LinearSVC con una ponderación de las características a través del TF-IDF, mediante una reducción de las características y realizando una normalización mediante *stemming* y eliminación de *stopwords*.
- **best_estimator:** Estimador que fue elegido por la búsqueda, es decir, estimador con la puntuación más alta (o la pérdida más pequeña si se específica) en los datos omitidos (gri, 2020). En este

caso con, $(refit = f1\_weighted)$[32], se obtiene como mejor clasificador el $LinearSVC(C = 1{,}0)$ con $TfidfVectorizer(use\_idf = False)$.

```
scoring = {'accuracy': 'accuracy',
           'precision_macro': 'precision_macro',
           'recall_macro': 'recall_macro',
           'f1_macro': 'f1_macro',
           'precision_micro': 'precision_micro',
           'recall_micro': 'recall_micro',
           'f1_micro': 'f1_micro',
           'precision_weighted': 'precision_weighted',
           'recall_weighted': 'recall_weighted',
           'f1_weighted': 'f1_weighted',
           }
pipeline = Pipeline([('vectorizer', None),
                     ('classifier', None)])
tokenizer = TweetTokenizer().tokenize
bow_binary_term_ocurrences = CountVectorizer(binary=True, tokenizer=tokenizer)
bow_absolute_term_ocurrences = CountVectorizer(binary=False, tokenizer=tokenizer)
bow_term_frequency = TfidfVectorizer(use_idf=False, tokenizer=tokenizer)
bow_tfidf = TfidfVectorizer(use_idf=True, tokenizer=tokenizer)
parameters = [{
    'vectorizer': (bow_binary_term_ocurrences,
                   bow_absolute_term_ocurrences,
                   bow_term_frequency,
                   bow_tfidf),
    'vectorizer__preprocessor': (Preprocessor(text_features=Preprocessor.REMOVE).
        preprocess,
                                 Preprocessor(text_features=Preprocessor.REMOVE,
                                     stemming=True).preprocess,
                                 Preprocessor(text_features=Preprocessor.NORMALIZE).
                                     preprocess,
                                 Preprocessor(text_features=Preprocessor.NORMALIZE,
                                     stemming=True).preprocess),
    'vectorizer__stop_words': (None, spanish_stopwords),
    'classifier': (MultinomialNB(), LinearSVC(), DecisionTreeClassifier(),
        KNeighborsClassifier(n_neighbors=30))
}]
```

Figura 3.6: Fragmento de código para la validación cruzada mediante GridSearchCV.

```
if __name__ == '__main__':
    skf = StratifiedKFold(n_splits=10, shuffle=True)
    grid_search = GridSearchCV(pipeline, param_grid=parameters, n_jobs=-1, cv=skf,
        verbose=5, scoring=scoring,
                               refit='f1_weighted', return_train_score=False)
    grid_search.fit(message, label)
    print("best_score:", grid_search.best_score_)
    print("best_parameters:", grid_search.best_params_)
    print("best_estimator:", grid_search.best_estimator_)
    pd.DataFrame(grid_search.cv_results_).to_csv(path_or_buf='baseline.csv',
                                                 quoting=csv.QUOTE_NONNUMERIC)
```

Figura 3.7: Fragmento de código para la validación cruzada mediante GridSearchCV.

[32]refit es un estimador utilizando los mejores parámetros encontrados en todo el conjunto de datos.

```
best_estimator: Pipeline(memory=None,
         steps=[('vectorizer',
                 TfidfVectorizer(analyzer='word', binary=False,
                                 decode_error='strict',
                                 dtype=<class 'numpy.float64'>,
                                 encoding='utf-8', input='content',
                                 lowercase=True, max_df=1.0, max_features=None,
                                 min_df=1, ngram_range=(1, 1), norm='l2',
                                 preprocessor=<bound method Preprocessor.preprocess of Preprocessor([text_features=normalize, stemming=T...
                                 tokenizer=<bound method ToktokTokenizer.tokenize of <nltk.tokenize.toktok.ToktokTokenizer object at 0x000002A520E91588>>,
                                 use_idf=False, vocabulary=None)),
                ('classifier',
                 LinearSVC(C=1.0, class_weight=None, dual=True,
                           fit_intercept=True, intercept_scaling=1,
                           loss='squared_hinge', max_iter=1000,
                           multi_class='ovr', penalty='l2', random_state=None,
                           tol=0.0001, verbose=0))],
         verbose=False)
```

Figura 3.8: Best estimator, creación propia

```
best_score: 0.6682089573733032
best_parameters: {'classifier': LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0), 'vectorizer': TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.float64'>, encoding='utf-8',
                input='content', lowercase=True, max_df=1.0, max_features=None,
                min_df=1, ngram_range=(1, 1), norm='l2',
                preprocessor=<bound method Preprocessor.preprocess of Preprocessor([text_features=normalize, stemming=True])>,
                smooth_idf=True, stop_words=None, strip_accents=None,
                sublinear_tf=False, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=<bound method ToktokTokenizer.tokenize of <nltk.tokenize.toktok.ToktokTokenizer object at 0x000002A5208DB630>>,
                use_idf=False, vocabulary=None), 'vectorizer__preprocessor': <bound method Preprocessor.preprocess of Preprocessor([text_features=normalize, stemming=True])>, 
'vectorizer__stop_words': None}
```

Figura 3.9: Best score and best parameters, creación propia

## 3.3. Aplicación del método en un entorno real

Para probar el método seleccionado se trabajó con un corpus que contiene registro de mensajes extraídos de la red de mensajería instantánea de la UCI almacenado en un archivo de extensión .csv, donde las columnas, que contienen los atributos del mensaje están separadas por comas, y las filas, que contienen un mensaje, por saltos de línea. Cada mensaje tiene la siguiente estructura, ver Figura 3.10:

- Conversación: es un número que identifica una conversación en el dominio.
- Emisor y Receptor: son cadenas de texto únicas que identifican al usuario que emite y recibe el mensaje respectivamente.
- C_emisor y C_receptor: son cadenas de texto que identifican el cliente de mensajería instantánea del que emite y recibe el mensaje respectivamente.
- Tiempo: es el instante de tiempo en milisegundos en que se envía el mensaje.
- Contenido: es una cadena de texto con el contenido del mensaje.

```
107915951,â,¬Ã—Â¬Ã¾Ã—Â¥Ã¦,jabber.uci.cu/b114b945,Â»&Ã§â,¬Ã©â,¬Ã§,jabber.uci.cu/e9de1ff8,1574227292024,sabes si por fin es p mannana
```

Figura 3.10: Estructura de un mensaje.

Para desarrollar correctamente la solución que se propone se hace necesario cargar este registro de mensajes en una estructura que sea fácil y rápida de manipular ya que, una vez que los datos sean correctamente cargados estos pasarán por varias transformaciones. Pandas, facilita el proceso de

análisis de grandes volúmenes de datos de manera eficiente, por tal motivo, se decidió hacer uso de la misma en la presente investigación cargando el conjunto de datos de entrada en un objeto DataFrame de Pandas que permite manejar fácilmente datos tabulares para el análisis de datos, ver Figura 3.11.

```
user_file = 'C:/Users/Lethy/PycharmProjects/SAIA/datasets/jabber_cifrado.csv'
if os.path.exists(user_file):
    prev_data = pd.read_csv(user_file, names=['id_conversation', 'id_sender', '
        id_sender_client', 'id_receiver',
                                                'id_receiver_client', 'time', 'message
                                                    '])
```

Figura 3.11: Fragmento de código de lectura mediante la librería Pandas.

| | id_conversation | id_sender | id_sender_client | id_receiver | id_receiver_client | time | message |
|---|---|---|---|---|---|---|---|
| 0 | 107915951 | €x¬þx¥æ | jabber.uci.cu/b114b945 | »&ç€é€ç | jabber.uci.cu/e9de1ff8 | 1574227292024 | sabes si por fin es p mannana;;;;;;;;;;;;;;; |
| 1 | 107915951 | »&ç€é€ç | jabber.uci.cu/e9de1ff8 | €x¬þx¥æ | jabber.uci.cu/b114b945 | 1574227298401 | eso dicen;;;;;;;;;;;;;;; |
| 2 | 107915950 | ®¬á¬€xæþ | jabber.uci.cu/Pandion | <¤æþ®¬áç®á | jabber.uci.cu/b5f5e31e | 1574227328212 | q van a haceR?;;;;;;;;;;;;;;; |
| 3 | 107915954 | ¬é¬¶¬€ç€ | jabber.uci.cu/df36ffa9 | €äxçßçþ¬ | jabber.uci.cu/4db432fe | 1574227341219 | ?;;;;;;;;;;;;;;; |
| 4 | 107915949 | »¥þç€¬€¥é» | jabber.uci.cu/1a543b23 | €äxçßçþ¬ | jabber.uci.cu/4db432fe | 1574226506930 | muere entonces;;;;;;;;;;;;;;; |
| 5 | 107915946 | ü<x¬þé¥áâæ | jabber.uci.cu/8c4e42e5 | €äxçßçþ¬ | jabber.uci.cu/4db432fe | 1574226508765 | lo tienes al lado ;;;;;;;;;;;;;;; |
| 6 | 107915946 | ü<x¬þé¥áâæ | jabber.uci.cu/8c4e42e5 | €äxçßçþ¬ | jabber.uci.cu/4db432fe | 1574226509701 | ? ;;;;;;;;;;;;;;; |
| 7 | 107915949 | €äxçßçþ¬ | jabber.uci.cu/4db432fe | »¥þç€¬€¥é» | jabber.uci.cu/1a543b23 | 1574226514472 | pfff ;;;;;;;;;;;;;;; |
| 8 | 107915946 | €äxçßçþ¬ | jabber.uci.cu/4db432fe | ü<x¬þé¥áâæ | jabber.uci.cu/8c4e42e5 | 1574226524995 | cerca ;;;;;;;;;;;;;;; |
| 9 | 107915949 | €äxçßçþ¬ | jabber.uci.cu/4db432fe | »¥þç€¬€¥é» | jabber.uci.cu/1a543b23 | 1574226531097 | copiame la apk ;;;;;;;;;;;;;;; |
| 10 | 107915949 | »¥þç€¬€¥é» | jabber.uci.cu/1a543b23 | €äxçßçþ¬ | jabber.uci.cu/4db432fe | 1574226538601 | o no ;;;;;;;;;;;;;;; |

Figura 3.12: Estructura de un mensaje.

Una vez cargado, ver Figura 3.12, el dataset de prueba la característica tiempo es dividida en *Fecha* y *Hora* para un mejor manejo del *Tiempo*, como se muestra en la Figura 3.13.

| | Conversation | Sender | Receiver | Time | Message | Date | Hours |
|---|---|---|---|---|---|---|---|
| 0 | 107915951 | €x¬þx¥æ | »&ç€é€ç | 2019-11-20 00:21:32 | sabes si por fin es p mannana | 2019-11-20 | 00:21:32 |
| 1 | 107915951 | »&ç€é€ç | €x¬þx¥æ | 2019-11-20 00:21:38 | eso dicen | 2019-11-20 | 00:21:38 |
| 2 | 107915950 | ®¬á¬€xæþ | <¤æþ®¬áç®á | 2019-11-20 00:22:08 | q van a haceR? | 2019-11-20 | 00:22:08 |
| 3 | 107915954 | ¬é¬¶¬€ç€ | €äxçßçþ¬ | 2019-11-20 00:22:21 | ? | 2019-11-20 | 00:22:21 |
| 4 | 107915949 | »¥þç€¬€¥é» | €äxçßçþ¬ | 2019-11-20 00:08:26 | muere entonces | 2019-11-20 | 00:08:26 |
| 5 | 107915946 | ü<x¬þé¥áâæ | €äxçßçþ¬ | 2019-11-20 00:08:28 | lo tienes al lado | 2019-11-20 | 00:08:28 |
| 6 | 107915946 | ü<x¬þé¥áâæ | €äxçßçþ¬ | 2019-11-20 00:08:29 | ? | 2019-11-20 | 00:08:29 |
| 7 | 107915949 | €äxçßçþ¬ | »¥þç€¬€¥é» | 2019-11-20 00:08:34 | pfff | 2019-11-20 | 00:08:34 |
| 8 | 107915946 | €äxçßçþ¬ | ü<x¬þé¥áâæ | 2019-11-20 00:08:44 | cerca | 2019-11-20 | 00:08:44 |
| 9 | 107915949 | €äxçßçþ¬ | »¥þç€¬€¥é» | 2019-11-20 00:08:51 | copiame la apk | 2019-11-20 | 00:08:51 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 110677 | 107918076 | x¶€çª¡¥þ | á€¤æþ®¬áç® | 2019-11-24 21:05:24 | manda to loq tengas | 2019-11-24 | 21:05:24 |
| 110678 | 107918077 | <x¤æ®¬ | ¬<€¡¬€ç® | 2019-11-24 21:05:25 | mejia tranquilo vemos eso  pero mañana q ahora... | 2019-11-24 | 21:05:25 |
| 110679 | 107924312 | €€<æ¤ç | æáßç€ç® | 2019-12-11 23:54:25 | pp hay q cerrar el dota pa volver a crear ... | 2019-12-11 | 23:54:25 |
| 110680 | 107924196 | <&ç®µ¡¥¬ | »&¥¤çþç€ | 2019-12-11 21:36:49 | asao de mierda | 2019-12-11 | 21:36:49 |
| 110681 | 107924186 | ä€¬á¬®¬€ | äß¬x€é¬ | 2019-12-11 21:36:56 | habla | 2019-12-11 | 21:36:56 |
| 110682 | 107924186 | äß¬x€é¬ | ä€¬á¬®¬€ | 2019-12-11 21:37:06 | era pa lo del payday | 2019-12-11 | 21:37:06 |
| 110683 | 107924186 | ä€¬á¬®¬€ | äß¬x€é¬ | 2019-12-11 21:37:13 | ah ya | 2019-12-11 | 21:37:13 |
| 110684 | 107924188 | ¬¤¬€þ¬x®ç | &¬&æá¥þ¬ | 2019-12-11 21:37:19 | yo terminé con mi pareja en septiembre y bueno... | 2019-12-11 | 21:37:19 |
| 110685 | 107924186 | ä€¬á¬®¬€ | äß¬x€é¬ | 2019-12-11 21:37:20 | dl mañana ñe metemos po la tarde | 2019-12-11 | 21:37:20 |
| 110686 | 107924220 | €äxçßçþ¬ | á€¤æþ®¬áç® | 2019-12-11 22:02:28 | mia  que bien se el dibujo oscuro | 2019-12-11 | 22:02:28 |

Figura 3.13: Transformación del dataset.

A partir de aquí se seleccionan solo las características importantes para el análisis de sentimientos mediante una agrupación por un usuario determinado. De este proceso se obtiene el id que identifica la conversación, el usuario que recibe el mensaje, el tiempo en que se envía; con fecha y hora y el contenido del mensaje, como se observa en la Figura 3.14. Estas características permiten determinar el sentimiento general de las conversaciones y el sentimiento por cada mensaje de la conversación.

| | Conversation | Receiver | Time | Message | Date | Hours |
|---|---|---|---|---|---|---|
| 0 | 107917764 | x€ä¥¬® | 2019-11-23 21:46:09 | sa ya le pregunte | 2019-11-23 | 21:46:09 |
| 1 | 107922320 | ¬é¬¶¬€ç€ | 2019-12-07 16:52:40 | na | 2019-12-07 | 16:52:40 |
| 2 | 107922328 | ¬é¬¶¬€ç€ | 2019-12-07 17:51:47 | jajaja | 2019-12-07 | 17:51:47 |
| 3 | 107922329 | €€<æ€¤ç | 2019-12-07 17:53:22 | a dl dl | 2019-12-07 | 17:53:22 |
| 4 | 107921610 | ö¶¬ááç€éç€æ€ | 2019-12-05 13:26:30 | teñg0 el teclad0 desc0ñectad0 y me pesa p0ñerl... | 2019-12-05 | 13:26:30 |
| 5 | 107921436 | æáßç€ç® | 2019-12-04 23:20:51 | toy conectao en mbface | 2019-12-04 | 23:20:51 |
| 6 | 107923834 | ü¬x¬þä¬ | 2019-12-10 23:09:23 | https://www.amazon.es/lunaoo-Lapices-Material-... | 2019-12-10 | 23:09:23 |
| 7 | 107916386 | €€<æ€¤ç | 2019-11-20 16:31:54 | 178? | 2019-11-20 | 16:31:54 |
| 8 | 107922113 | æáßç€ç® | 2019-12-06 15:34:38 | sa | 2019-12-06 | 15:34:38 |
| 9 | 107922963 | ö¶¬ááç€éç€æ€ | 2019-12-08 21:23:18 | la otra el que la tiene es el choco dejame dec... | 2019-12-08 | 21:23:18 |
| .. | ... | ... | ... | ... | ... | ... |
| 334 | 107923789 | ü¬x¬þä¬ | 2019-12-10 20:39:35 | hada* | 2019-12-10 | 20:39:35 |
| 335 | 107923834 | ü¬x¬þä¬ | 2019-12-10 23:09:22 | este pack lo tiene tdo | 2019-12-10 | 23:09:22 |
| 336 | 107916386 | €€<æ€¤ç | 2019-11-20 16:28:55 | le aviso a omar o ya esta ahi? | 2019-11-20 | 16:28:55 |
| 337 | 107923227 | €€<æ€¤ç | 2019-12-09 15:20:53 | avisale al bbo | 2019-12-09 | 15:20:53 |
| 338 | 107921134 | æáßç€ç® | 2019-12-04 14:45:20 | deja ver | 2019-12-04 | 14:45:20 |
| 339 | 107918586 | æáßç€ç® | 2019-11-25 17:13:27 | ya estan pay? | 2019-11-25 | 17:13:27 |
| 340 | 107923826 | <xçþ€¥µ¡ç® | 2019-12-10 22:39:35 | a okok | 2019-12-10 | 22:39:35 |
| 341 | 107915946 | ü<x¬þé¥ááæ | 2019-11-20 00:19:26 | jajaja no te lo voy a poder andar | 2019-11-20 | 00:19:26 |
| 342 | 107918023 | ü¬x¬þä¬ | 2019-11-24 19:11:54 | ya si las consigo yo te digo | 2019-11-24 | 19:11:54 |
| 343 | 107917577 | x»x¬€é¬»¬ | 2019-11-22 20:20:46 | acabo de comer  estoy en messenger hablando co... | 2019-11-22 | 20:20:46 |

Figura 3.14: Conversaciones del usuario seleccionado.

| | Words | Word Length | Message Characters |
|---|---|---|---|
| [sa, ya, le, pregunte, ] | | 4 | 39 |
| [na, ] | | 1 | 24 |
| [jajaja, ] | | 1 | 28 |
| [a, dl, dl, ] | | 3 | 29 |
| [teñg0, el, teclad0, desc0ñectad0, y, me, pesa... | | 8 | 69 |
| [toy, conectao, en, mbface, ] | | 4 | 44 |
| [https, www, amazon, es, lunaoo, Lapices, Mate... | | 24 | 214 |
| [178, ] | | 1 | 26 |
| [sa, ] | | 1 | 24 |
| [la, otra, el, que, la, tiene, es, el, choco, ... | | 11 | 74 |
| ... | | ... | ... |
| [hada, ] | | 1 | 27 |
| [este, pack, lo, tiene, tdo, ] | | 5 | 44 |
| [le, aviso, a, omar, o, ya, esta, ahi, ] | | 8 | 52 |
| [avisale, al, bbo, ] | | 3 | 36 |
| [deja, ver, ] | | 2 | 30 |
| [ya, estan, pay, ] | | 3 | 35 |
| [a, okok, ] | | 2 | 28 |
| [jajaja, no, te, lo, voy, a, poder, andar, ] | | 8 | 56 |
| [ya, si, las, consigo, yo, te, digo, ] | | 7 | 50 |
| [acabo, de, comer, estoy, en, messenger, habla... | | 9 | 76 |

Figura 3.15: Limpieza y transformación de los mensajes.

Teniendo en cuenta esta selección de datos, al conjunto de mensajes se le realiza un procesamiento de los datos para aplicar el método seleccionado para la clasificación de la polaridad del sentimiento. Obteniéndose los siguientes elementos, ver en las Figuras 3.15 y 3.16:

- *Words*, mensaje tokenizado por unigramas, es decir en palabras.
- *Word Length*, cantidad de tokens.
- *Message Character*, es la cantidad de caracteres contenidos en el mensaje.
- *ListofCleanWords*, lista de palabras después de eliminar aquellas palabras que son consideradas Stopwords de los comentarios.
- *CleanWordsasText* es la unión de cada una de las palabras como si fuese un texto simple.

| ListofCleanWords | CleanWordsasText |
|---|---|
| [sa, pregunte] | sa pregunte |
| [na] | na |
| [jajaja] | jajaja |
| [dl, dl] | dl dl |
| [g, teclad, desc, ectad, pesa, p, erl] | g teclad desc ectad pesa p erl |
| [toy, conectao, mbface] | toy conectao mbface |
| [https, www, amazon, lunaoo, lapices, material... | https www amazon lunaoo lapices material carbo... |
| [] | |
| [sa] | sa |
| [choco, dejame, decircelo] | choco dejame decircelo |
| ... | ... |
| [hada] | hada |
| [pack, tdo] | pack tdo |
| [aviso, omar, ahi] | aviso omar ahi |
| [avisale, bbo] | avisale bbo |
| [deja, ver] | deja ver |
| [estan, pay] | estan pay |
| [okok] | okok |
| [jajaja, voy, poder, andar] | jajaja voy poder andar |
| [si, consigo, digo] | si consigo digo |
| [acabo, comer, messenger, hablando, arlety] | acabo comer messenger hablando arlety |

Figura 3.16: Limpieza y transformación de los mensajes.

Luego de realizar estos procesos de transformación, los mensajes están listos para ser clasificados. Seguidamente se crean las columnas *Score* (contiene la polaridad del mensaje), *Positive*, *Negative* y *Neutral*, estas tres últimas marcan con 0 y 1 la presencia del sentimiento, además de la característica *docLen* que permitirá calcular la media de palabras por el tamaño del documento, estas nuevas características se muestran en la Figura 3.17.

| Word Length | Message Characters | ListofCleanWords | CleanWordsasText | Score | Positive | Negative | Neutral | docLen |
|---|---|---|---|---|---|---|---|---|
| 1 | 25 | ['pff'] | pff | neutro | 0 | 0 | 1 | 11 |
| 7 | 49 | ['hace', 'falta'] | hace falta | neutro | 0 | 0 | 1 | 17 |
| 7 | 50 | ['omar', 'segundo', 'alla'] | omar segundo alla | neutro | 0 | 0 | 1 | 17 |
| 1 | 24 | ['sa'] | sa | positive | 1 | 0 | 0 | 11 |
| 6 | 51 | ['dl', 'crea', 'abriendo'] | dl crea abriendo | neutro | 0 | 0 | 1 | 16 |
| 7 | 46 | ['van', 'jugar', 'mas'] | van jugar mas | positive | 1 | 0 | 0 | 17 |
| 8 | 52 | ['salgo', 'gente', 'igual'] | salgo gente igual | positive | 1 | 0 | 0 | 18 |
| 3 | 30 | ['ip'] | ip | neutro | 0 | 0 | 1 | 13 |
| 3 | 32 | ['mate'] | mate | negative | 0 | 1 | 0 | 13 |
| 6 | 50 | ['queer', 'ada', 'madrina'] | queer ada madrina | neutro | 0 | 0 | 1 | 16 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 5 | 41 | ['mande', 'creo'] | mande creo | positive | 1 | 0 | 0 | 15 |
| 3 | 37 | ['hijo', 'vamos', 'dota'] | hijo vamos dota | neutro | 0 | 0 | 1 | 13 |
| 3 | 38 | ['jajajaja', 'hh', 'okok'] | jajajaja hh okok | neutro | 0 | 0 | 1 | 13 |
| 4 | 42 | ['dl', 'pley'] | dl pley | neutro | 0 | 0 | 1 | 14 |
| 5 | 50 | ['rick', 'saliendo', 'cuerta'] | rick saliendo cuerta | neutro | 0 | 0 | 1 | 15 |
| 5 | 49 | ['singueta', 'bueno'] | singueta bueno | positive | 1 | 0 | 0 | 15 |
| 4 | 40 | ['faltan', 'ds', 'mas'] | faltan ds mas | negative | 0 | 1 | 0 | 14 |
| 6 | 48 | ['pregunta'] | pregunta | neutro | 0 | 0 | 1 | 17 |
| 6 | 52 | ['llev', 'ahora', 'dejame', 'ba', 'arme'] | llev ahora dejame ba arme | positive | 1 | 0 | 0 | 16 |
| 1 | 30 | ['black'] | black | neutro | 0 | 0 | 1 | 11 |

Figura 3.17: Detección de polaridad en los mensajes.

## 3.4. Visualización de los resultados obtenidos

Existe una amplia variedad de gráficos, los más usados son aquellos que por su simplicidad facilitan al usuario su comprensión. Los distintos gráficos, poseen diferentes enfoques según los tipos de preguntas que se deseen resolver mediante su integración (Meyer y Fisher, 2018).

La utilización de los componentes gráficos brindados por MatPlotLib permite obtener un mejor entendimiento de la solución sobre el análisis de sentimientos realizado en las conversaciones de la red, mediante diferentes visualizaciones que se pueden apreciar a continuación.

### 3.4.1. Análisis del chat del usuario

En la Figura 3.18 se visualiza el comportamiento de la fecha en que escribe el usuario, lo que facilita determinar la frecuencia en que escribe con en relación a la cantidad de mensajes intercambiados. En la Figura 3.19 se muestra una distribución en el tiempo.

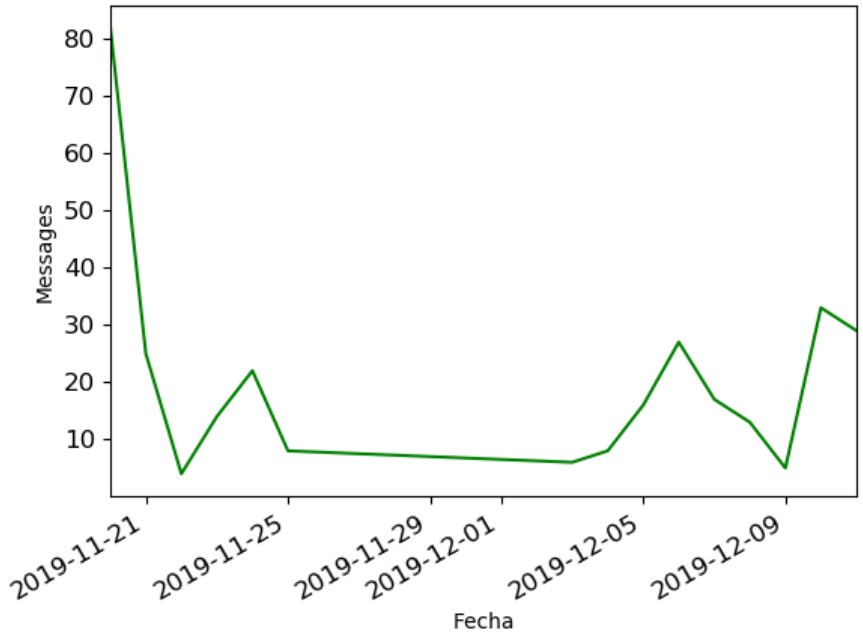



Figura 3.18: Frecuencia de chat del usuario por fecha.

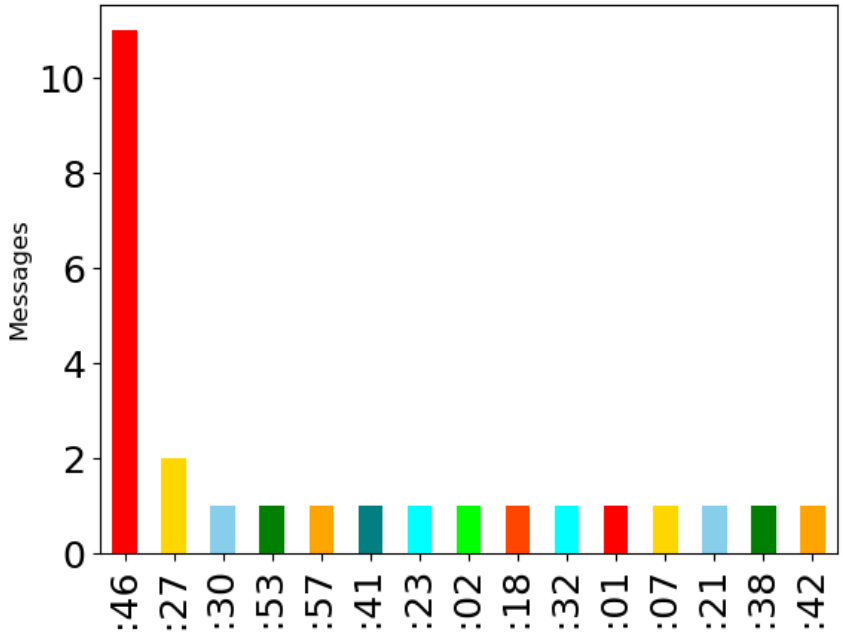



Figura 3.19: Distribución de chat del usuario por fecha.

### 3.4.2. Análisis de los usuarios

En las Figuras 3.20 y 3.21 se muestra una distribución de los mensajes intercambiados por el usuario seleccionado y sus contactos, permitiendo ver cuál es el usuario con quien más se escribe. Como se puede observar es el usuario æáßç€ç©.

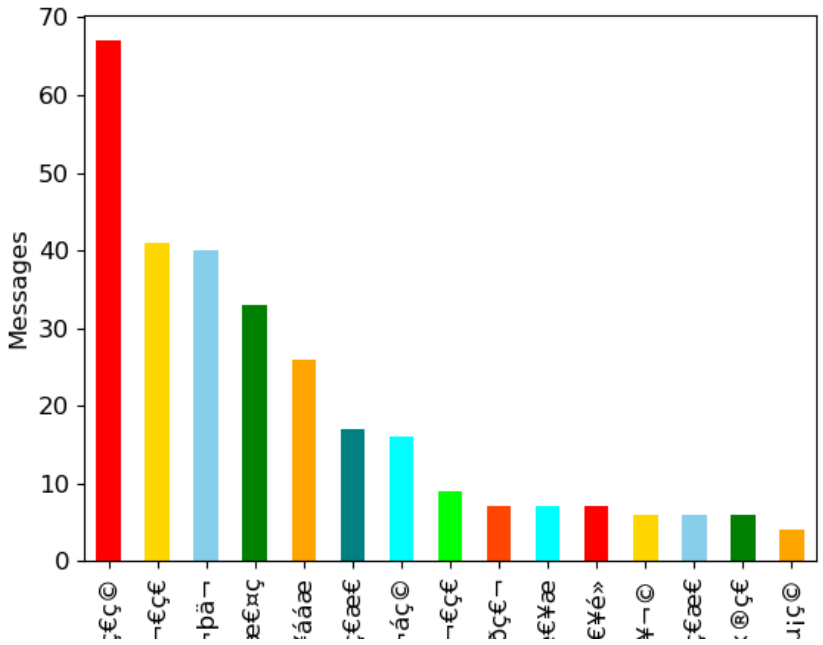



Figura 3.20: Distribución de mensajes por usuarios.

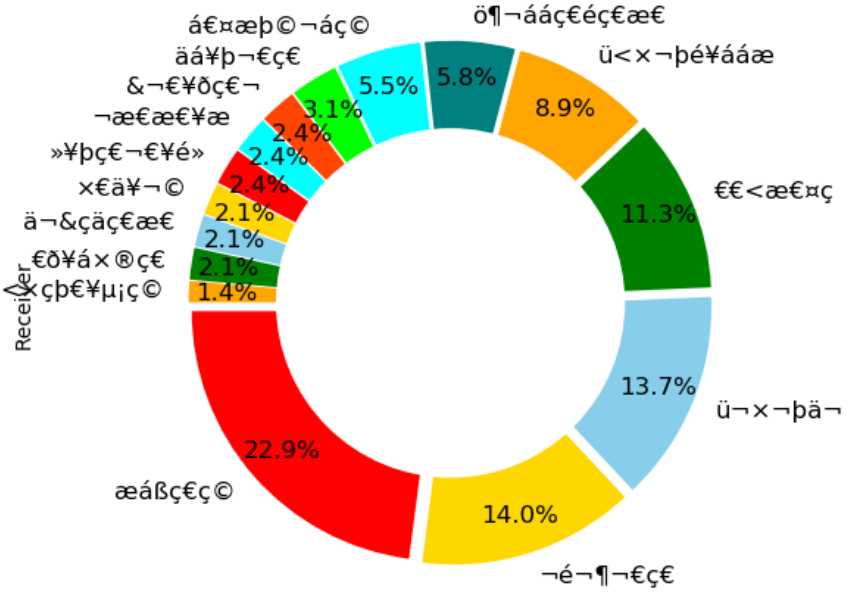



Figura 3.21: Distribución por usuarios.

### 3.4.3. Análisis de los mensajes por usuario

Las siguientes visualizaciones muestran el comportamiento de las palabras escrita por el usuario seleccionado. La Figura 3.22 describe la media del número de palabras por el tamaño del documento, en este caso es la cantidad de mensajes escritos.

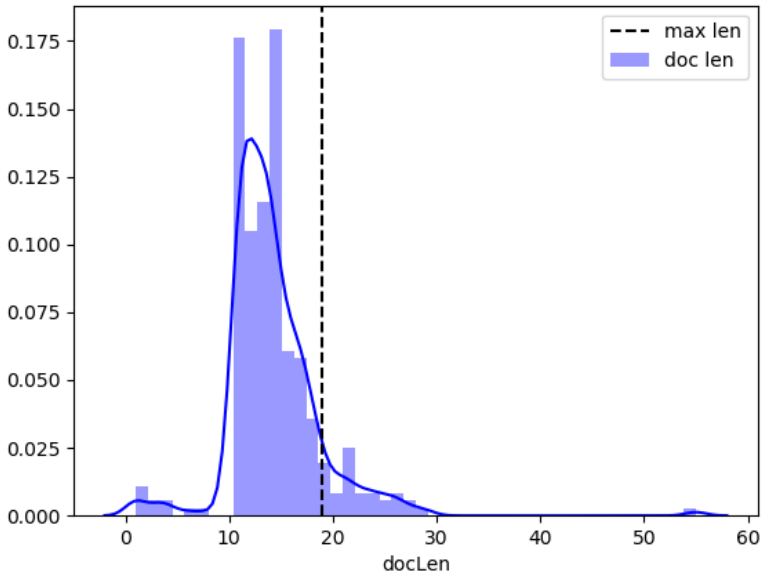



Figura 3.22: Media de palabras escritas.

Un aspecto a tener en cuenta es que la cantidad de palabras escritas no determinan la cantidad de veces con que se escriben los usuarios. El usuario æáßç€ç©, fue seleccionado como el más frecuente, sin embargo no es el que más escribe, como se muestra en la Figura 3.23.

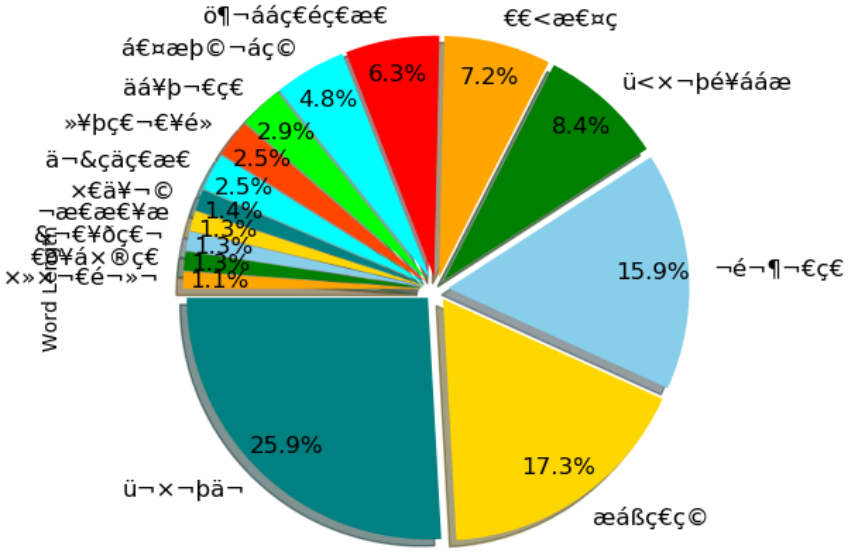



Figura 3.23: Distribución por palabras.

### 3.4.4. Análisis de polaridad del sentimiento

En la Figura 3.24 se muestra el porcentaje de mensajes clasificados en el conjunto de datos por cada clase. Lo cuál permite observar que las clases neutro y positivo son las mayores, clasificándose un total de 168 como neutros, 93 como positivo y 48 como negativos, se puede señalar que la diferencia entre ellos es debido al corpus con que se entrenó el método de clasificación.

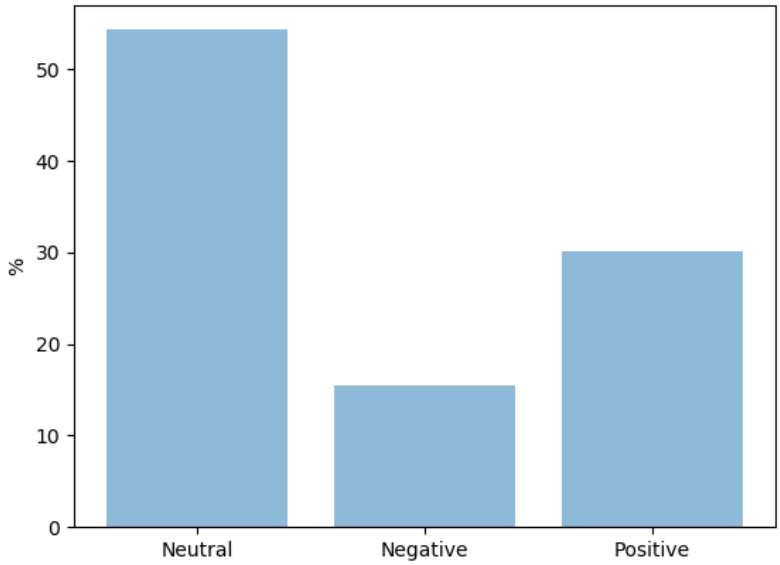



Figura 3.24: % de mensajes clasificados por clases.

A continuación en la Figura 3.25 se muestra la distribución de clasificación del sentimiento por fechas, tomando para el análisis la frecuencia por días.

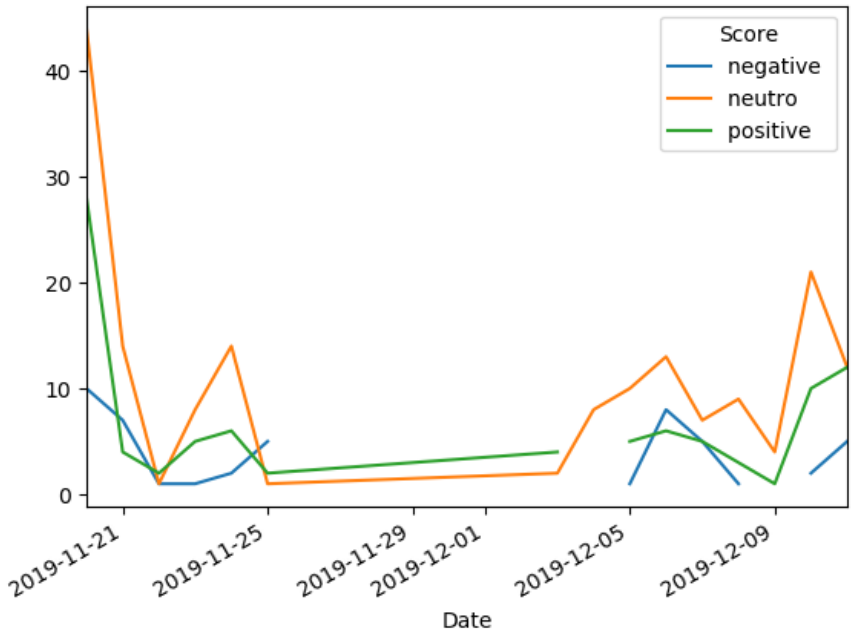



Figura 3.25: Polaridad del sentimiento en el tiempo.

### 3.4.5. Visualización de palabras clave

En la Figura 3.26 se puede observar realmente qué palabras y con qué ocurrencias son repetidas en las conversaciones del usuario, con la finalidad de obtener una idea de las principales palabras.

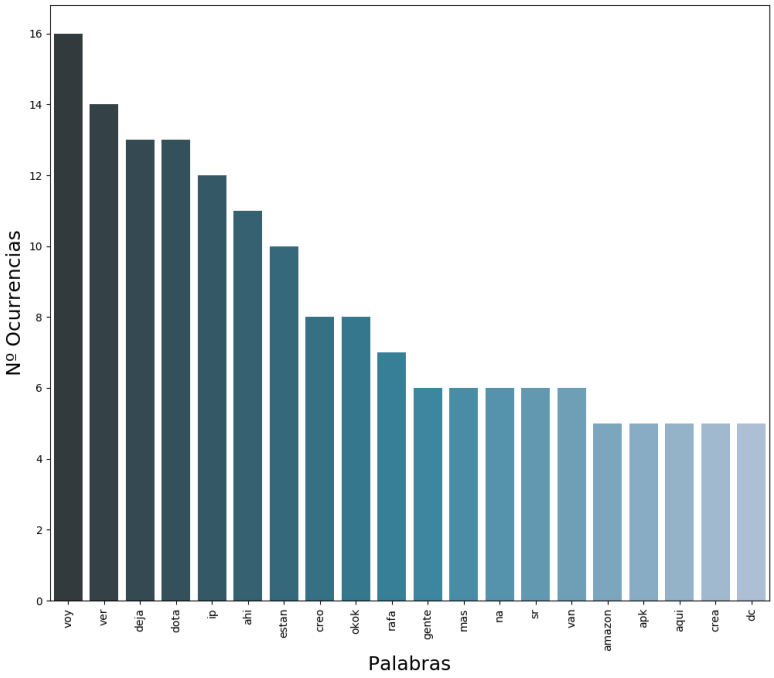



Figura 3.26: Ocurrencias de palabras.

Desde la Figura 3.27 hasta la Figura 3.29, se puede visualizar las palabras que más se repiten en cada categoría, mostrando las palabras que condicionan la categoría, siendo mayores las que más impacto tienen.

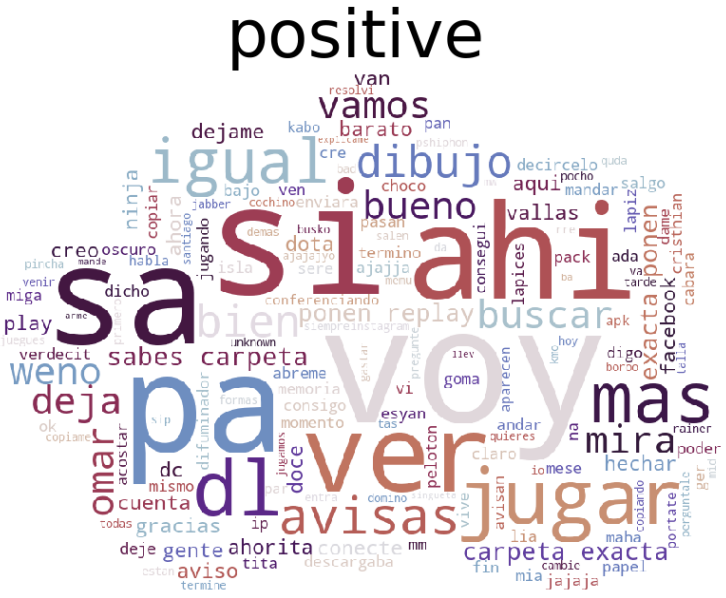



Figura 3.27: Nube de palabras positivas.

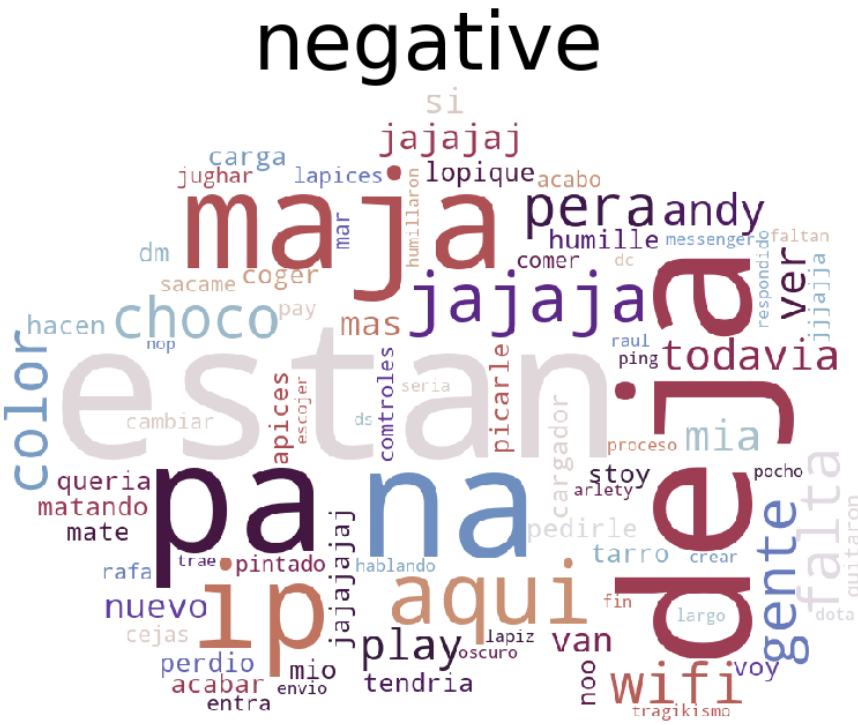



Figura 3.28: Nube de palabras negativas.

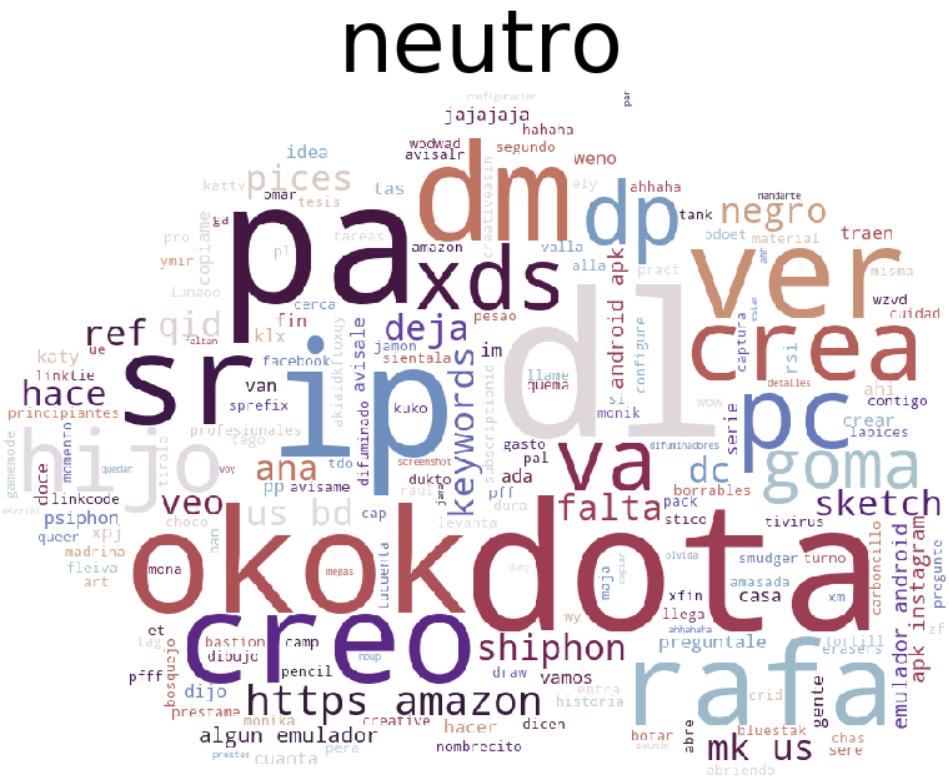



Figura 3.29: Nube de palabras neutras.

## 3.5. Conclusiones del capitulo

Al desarrollar el presente capítulo se arriban a las conclusiones siguientes:

1. Las métricas de evaluación prueban la calidad de la solución desarrollada, concluyendo que el método se encuentra listo para su utilización.
2. La validación cruzada permitió obtener los mejores resultados para el entrenamiento del clasificador y se comprobó la precisión de la solución implementada demostrando su utilidad y uso.
3. La aplicación del método en un entorno real evidenció la efectividad del mismo en el análisis de sentimientos en la red de mensajería institucional de la UCI, mostrando la polaridad de los mensajes escrito por un usuario y sus contactos.

# Conclusiones

Como resultado de la presente investigación, se obtuvo un método para detectar la polaridad en los mensajes de las conversaciones de los usuarios a partir del análisis de sentimientos realizado. En base a los resultados obtenidos se arriban a las conclusiones siguientes:

- El análisis del proceso de descubrimiento del conocimiento en bases de datos y los algoritmos de aprendizajes supervisados para la clasificación en la minería de texto permitieron identificar características competitivas para el análisis de sentimientos y tecnologías para definir e implementar un método de clasificación de polaridad de sentimientos.
- La utilización de un corpus en sistemas de aprendizaje automáticos para evaluar textos y clasificarlos en base a su polaridad es una de las formas más extendidas y efectivas para la construcción de estos sistemas.
- El método implementado en Python facilitó el procesamiento de un gran volumen de documentos de manera rápida y simple así como, detectar la polaridad de sentimiento en las conversaciones de los usuarios en la red a través de visualizaciones.
- Los resultado de la validación realizada al método demostró su efectividad y comprobar que cumple con la calidad de la solución propuesta, permitiendo obtener un método optimo bajo condiciones reales.

# Recomendaciones

Una vez cumplido el objetivo de la presente investigación se recomienda:

- Crear un corpus sobre los mensajes de la red de mensajería Jabber para posibles trabajos de análisis de sentimientos.
- Integrar este estudio al análisis de detección de tópicos.
- Continuar este estudio en base a otras características extraídas de las conversaciones de los usuario para obtener mejores resultados.

# Bibliografía


GridSearchCV - Analytics Lane. 2019a. URL https://www.analyticslane.com/2018/07/02/gridsearchcv. [Online; accessed 22. Sep. 2020].

¿Cuál es la diferencia entre parámetro e hiperparámetro? - Analytics Lane. 2019b. URL https://www.analyticslane.com/2019/12/16/cual-es-la-diferencia-entre-parametro-e-hiperparametro. [Online; accessed 22. Sep. 2020].

Anaconda Documentation — Anaconda documentation. 2020. URL https://docs.anaconda.com. [Online; accessed 26. Aug. 2020].

Cross-validation: evaluating estimator performance — scikit-learn 0.23.2 documentation. 2020. URL https://scikit-learn.org/stable/modules/cross_validation.html. [Online; accessed 3. Sep. 2020].

Evaluate the Performance of Machine Learning Algorithms in python using resampling. 2020. URL https://translate.google.com/translate?hl=es&sl=en&u=https://machinelearningmastery.com/evaluate-performance-machine-learning-algorithms-python-using-resampling/&prev=search&pto=aue. [Online; accessed 2. Sep. 2020].

Matplotlib: Python plotting — Matplotlib 3.3.2 documentation. 2020. URL https://matplotlib.org. [Online; accessed 20. Sep. 2020].

Mensajería instantánea - EcuRed. 2020. URL https://www.ecured.cu/Mensajer%C3%ADa_instant%C3%A1nea. [Online; accessed 1. Sep. 2020].

Natural Language Toolkit — NLTK 3.5 documentation. 2020. URL https://www.nltk.org. [Online; accessed 26. Aug. 2020].

PyCharm. 2020. URL https://www.jetbrains.com/es-es/pycharm. [Online; accessed 26. Aug. 2020].

scikit-learn: machine learning in Python — scikit-learn 0.23.2 documentation. 2020. URL https://scikit-learn.org/stable. [Online; accessed 25. Aug. 2020].

sklearn.model_selection.GridSearchCV — scikit-learn 0.23.2 documentation. 2020. URL https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. [Online; accessed 2. Sep. 2020].

sklearn.model_selection.train_test_split — scikit-learn 0.23.2 documentation. 2020. URL https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html. [Online; accessed 3. Sep. 2020].

Why Do I Get Different Results Each Time in Machine Learning? 2020. URL https://machinelearningmastery.com/different-results-each-time-in-machine-learning/&usg=ALkJrhiaQhiYEXNqK-3rI9ETe3b1hle4CQ. [Online; accessed 2. Sep. 2020].

Fotis Aisopos, George Papadakis, Konstantinos Tserpes, y Theodora Varvarigou. Content vs. context for sentiment analysis: a comparative analysis over microblogs. En *Proceedings of the 23rd ACM conference on Hypertext and social media*, págs. 187–196. 2012.

Mahmoud Al-Ayyoub, Abed Allah Khamaiseh, Yaser Jararweh, y Mohammed N Al-Kabi. A comprehensive survey of arabic sentiment analysis. *Information processing & management*, 56(2):320–342, 2019.

Mario Alberto Amores Fernández. *Detección de la polaridad de las opiniones basada en nuevos recursos léxicos*. Tesis Doctoral, Universidad Central "Marta Abreu" de Las Villas, 2016.

Ana Isabel Rojão Lourenço Azevedo y Manuel Filipe Santos. Kdd, semma and crisp-dm: a parallel overview. *IADS-DM*, 2008.

Pierpaolo Basile, Valerio Basile, Malvina Nissim, Nicole Novielli, Viviana Patti, et al. Sentiment analysis of microblogging data. 2018.

Koyel Chakraborty, Siddhartha Bhattacharyya, y Rajib Bag. A survey of sentiment analysis from social media data. *IEEE Transactions on Computational Social Systems*, 7(2):450–464, 2020.

Juliet Díaz Lazo, Adriana Pérez Gutiérrez, y René Florido Bacallao. IMPACTO DE LAS TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES (TIC) PARA DISMINUIR LA BRECHA DIGITAL EN LA SOCIEDAD ACTUAL. *Cultivos Tropicales*, 32(1):81–90, 2011. ISSN 0258-5936. URL http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S0258-59362011000100009.

Usama Fayyad, Gregory Piatetsky-Shapiro, y Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37–37, 1996a.

Usama Fayyad, Gregory Piatetsky-Shapiro, y Padhraic Smyth. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, 1996b.

Susan Feldman. Nlp meets the jabberwocky: Natural language processing in information retrieval. *ONLINE-WESTON THEN WILTON-*, 23:62–73, 1999.

Klint Finley. What Exactly Is GitHub Anyway? *TechCrunch*, 2012. URL https://techcrunch.com/2012/07/14/what-exactly-is-github-anyway.

José Hernández Orallo, César Ferri Ramírez, y M.José Ramírez Quintana. *Introducción a la Minería de Datos*. 28042. Pearson Educación, 1a ed$^{ón}$., 2004. ISBN 84-205-4091-9.

Nitin Indurkhya y Fred J Damerau. *Handbook of natural language processing*. Chapman and Hall/CRC, 2010.

Ramandeep Kaur y Sandeep Kautish. Multimodal sentiment analysis: A survey and comparison. *International Journal of Service Science, Management, Engineering, and Technology (IJSSMET)*, 10(2):38–58, 2019.

Soo-Min Kim y Eduard Hovy. Determining the sentiment of opinions. En *Proceedings of the 20th international conference on Computational Linguistics*, pág. 1367. Association for Computational Linguistics, 2004.

Akshi Kumar y Mary Sebastian Teeja. Sentiment analysis: A perspective on its past, present and future. *International Journal of Intelligent Systems and Applications*, 4(10):1, 2012.

Lisandra Verdeja García Lazara Barrios Domínguez. *Recopilación y clasificacón automática de comentarios de los usuarios de aplicaciones de software*. Tesis Doctoral, Universidad de las Ciencias Informáticas, 2016.

Zuhe Li, Yangyu Fan, Bin Jiang, Tao Lei, y Weihua Liu. A survey on sentiment analysis and opinion mining for social multimedia. *Multimed. Tools Appl.*, 78(6):6939–6967, 2019. ISSN 1573-7721. doi: 10.1007/s11042-018-6445-z.

Elizabeth D Liddy. Natural language processing. 2001.

Bing Liu. *Web data mining: exploring hyperlinks, contents, and usage data*. Springer Science & Business Media, 2007.

Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

Bing Liu et al. Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2(2010):627–666, 2010.

Julio Amado López-Palma y Mayté Estévez-Crespo. Moas-les, herramienta para el análisis de sentimientos de textos en idioma español. En *XIV Congreso Internacional de Información Info'2016*. 2016.

Luca. ¿Qué es Python? 2020. URL https://web.archive.org/web/20200224120525/https://luca-d3.com/es/data-speaks/diccionario-tecnologico/python-lenguaje. [Online; accessed 26. Aug. 2020].

Diana Maynard y Adam Funk. Automatic detection of political opinions in tweets. En *Extended Semantic Web Conference*, págs. 88–99. Springer, 2011.

Wes McKinney. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. .ºReilly Media, Inc.", 2012.

Walaa Medhat, Ahmed Hassan, y Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113, 2014.

Yelena Aleksandrovna Mejova. Sentiment analysis within and across social media streams. 2012.

Miriah Meyer y Danyel Fisher. *Making Data Visual: A Practical Guide to Using Visualization for Insight*. O'Reilly Media, Incorporated, 2018.

M Dolores Molina-González, Eugenio Martínez-Cámara, María-Teresa Martín-Valdivia, y José M Perea-Ortega. Semantic orientation for polarity classification in spanish reviews. *Expert Systems with Applications*, 40(18):7250–7257, 2013.

Maria Ogneva. How companies can use sentiment analysis to improve their business. *Retrieved August*, 30, 2010.

F Oliva. *Minería de Opinión y Análisis de Sentimiento*. Tesis Doctoral, Tesis para optar por grado de ingenierio, Pontificia Universidad Católica de . . . , 2014.

Alessandro Ortis, Giovanni Maria Farinella, y Sebastiano Battiato. Survey on visual sentiment analysis. *IET Image Processing*, 14(8):1440–1456, 2020.

Alexander Pak y Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. En *LREc*, tomo 10, págs. 1320–1326. 2010.

Bo Pang, Lillian Lee, y Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. En *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, págs. 79–86. Association for Computational Linguistics, 2002.

Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135, 2008.

W Gerrod Parrott. *Emotions in social psychology: Essential readings*. Psychology Press, 2001.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, y E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Denilson Alves Pereira. A survey of sentiment analysis in the portuguese language. *Artificial Intelligence Review*, págs. 1–29, 2020.

Veronica Perez-Rosas, Carmen Banea, y Rada Mihalcea. Learning sentiment lexicons in spanish. En *LREC*, tomo 12, pág. 73. 2012.

Damien Poirier, Cécile Bothorel, Émilie Guimier De Neef, y Marc Boullé. Automating opinion analysis in film reviews: the case of statistic versus linguistic approach. En *Affective Computing and Sentiment Analysis*, págs. 125–140. Springer, 2011.

Ellen Riloff, Siddharth Patwardhan, y Janyce Wiebe. Feature subsumption for opinion analysis. En *Proceedings of the 2006 conference on empirical methods in natural language processing*, págs. 440–448. 2006.

Kevin Roebuck. *Sentiment Analysis: High-Impact Strategies-What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors*. Emereo Publishing, 2012.

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

Ignacio Saporiti y Juan Agustín Tibaldo. *Minería de opiniones y visualización de datos aplicables a estudios de mercado*. Tesis Doctoral, Universidad Nacional de La Plata, 2017.

Osvaldo Simeone. A very brief introduction to machine learning with applications to communication systems. *IEEE Transactions on Cognitive Communications and Networking*, 4(4):648–664, 2018.

José Carlos Sobrino Sande. *Análisis de sentimientos en Twitter*. Tesis Doctoral, Universitat Oberta de Catalunya, 2018.

Eduardo Sosa. Procesamiento del lenguaje natural: revisión del estado actual, bases teóricas y aplicaciones (parte i). *El profesional de la información*, 6, 1997.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, y Manfred Stede. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307, 2011.

Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. En *Proceedings of the 40th annual meeting on association for computational linguistics*, págs. 417–424. Association for Computational Linguistics, 2002.

Hao Wang y Jorge A Castanon. Sentiment expression via emoticons on social media. En *2015 ieee international conference on big data (big data)*, págs. 2404–2408. IEEE, 2015.

Janyce Wiebe, Rebecca Bruce, y Thomas P O'Hara. Development and use of a gold-standard data set for subjectivity classifications. En *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, págs. 246–253. 1999.

Janyce Wiebe, Theresa Wilson, y Claire Cardie. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210, 2005.

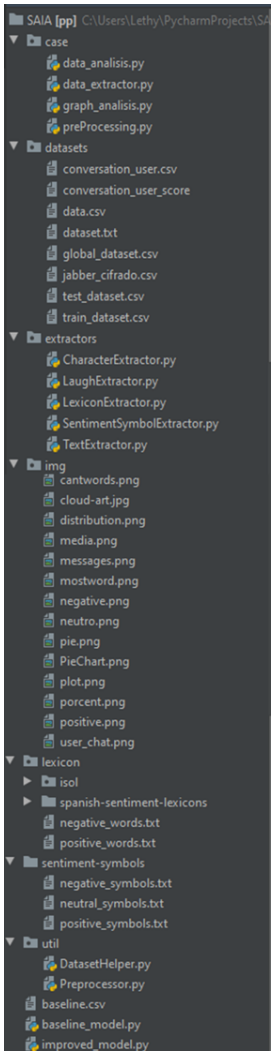
Janyce Wiebe et al. Learning subjetive adjectives from corpora. *Aaai/iaai*, 20(0):0, 2000.

Theresa Wilson, Janyce Wiebe, y Rebecca Hwa. Just how mad are you? finding strong and weak opinion clauses. En *aaai*, tomo 4, págs. 761–769. 2004.

Lin Yue, Weitong Chen, Xue Li, Wanli Zuo, y Minghao Yin. A survey of sentiment analysis in social media. *Knowledge and Information Systems*, págs. 1–47, 2019.

# Anexos

Estructura del proyecto




*Directorio del proyecto*

```python
import csv
import pandas as pd
from nltk import TweetTokenizer
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.svm import LinearSVC
from sklearn.tree import DecisionTreeClassifier
from util.DatasetHelper import DatasetHelper
from util.Preprocessor import Preprocessor

# global corpus 30% for cross-validation
message, label = DatasetHelper.csv_to_lists("datasets/train_dataset.csv")

# spanish stop words
spanish_stopwords = stopwords.words('spanish')

# metrics
scoring = {'accuracy': 'accuracy',
           'precision_macro': 'precision_macro',
           'recall_macro': 'recall_macro',
           'f1_macro': 'f1_macro',
           'precision_micro': 'precision_micro',
           'recall_micro': 'recall_micro',
           'f1_micro': 'f1_micro',
           'precision_weighted': 'precision_weighted',
           'recall_weighted': 'recall_weighted',
           'f1_weighted': 'f1_weighted',
           }

# pipeline
pipeline = Pipeline([('vectorizer', None),
                     ('classifier', None)])

# Tokenizer
tokenizer = TweetTokenizer().tokenize


# feature weights
bow_binary_term_ocurrences = CountVectorizer(binary=True, tokenizer=tokenizer)
bow_absolute_term_ocurrences = CountVectorizer(binary=False, tokenizer=tokenizer)
bow_term_frequency = TfidfVectorizer(use_idf=False, tokenizer=tokenizer)
bow_tfidf = TfidfVectorizer(use_idf=True, tokenizer=tokenizer)

parameters = [{
    'vectorizer': (bow_binary_term_ocurrences,
                   bow_absolute_term_ocurrences,
                   bow_term_frequency,
                   bow_tfidf),
    'vectorizer__preprocessor': (Preprocessor(text_features=Preprocessor.REMOVE).preprocess,
                                 Preprocessor(text_features=Preprocessor.REMOVE, stemming=True).preprocess,
                                 Preprocessor(text_features=Preprocessor.NORMALIZE).preprocess,
                                 Preprocessor(text_features=Preprocessor.NORMALIZE, stemming=True).preprocess),
    'vectorizer__stop_words': (None, spanish_stopwords),
    'classifier': (MultinomialNB(), LinearSVC(), DecisionTreeClassifier(), KNeighborsClassifier(n_neighbors=30))
}]

if __name__ == '__main__':
    skf = StratifiedKFold(n_splits=10, shuffle=True)
    grid_search = GridSearchCV(pipeline, param_grid=parameters, n_jobs=-1, cv=skf, verbose=5, scoring=scoring,
                               refit='f1_weighted', return_train_score=False)
    grid_search.fit(message, label)
    print("best_score:", grid_search.best_score_)
    print("best_parameters:", grid_search.best_params_)
    print("best_estimator:", grid_search.best_estimator_)
    pd.DataFrame(grid_search.cv_results_).to_csv(path_or_buf='baseline.csv',
                                                 quoting=csv.QUOTE_NONNUMERIC)

```

*baseline_model.py*

```python
from collections import Counter
from nltk import TweetTokenizer
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.metrics import classification_report, accuracy_score
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.svm import LinearSVC
from extractors.CharacterExtractor import CharacterExtractor
from extractors.LaughExtractor import LaughExtractor
from extractors.LexiconExtractor import LexiconExtractor
from extractors.SentimentSymbolExtractor import SentimentSymbolExtractor
from extractors.TextExtractor import TextExtractor
from util.DatasetHelper import DatasetHelper
from util.Preprocessor import Preprocessor

# global corpus 80% for training
message_train, label_train = DatasetHelper.csv_to_lists("datasets/train_dataset.csv")
message_test, label_test = DatasetHelper.csv_to_lists("datasets/test_dataset.csv")
message, label = DatasetHelper.csv_to_lists("datasets/global_dataset.csv")


# Tokenizer
tokenizer = TweetTokenizer().tokenize

# Preprocessor with stemming enabled
preprocessor = Preprocessor(text_features=None, stemming=True).preprocess

# Term Frequency
bow_term_frequency = TfidfVectorizer(use_idf=False, tokenizer=tokenizer, preprocessor=preprocessor)

pipeline = Pipeline([
    ('feats', FeatureUnion([
           ('vectorizer', bow_term_frequency),
           ('sentiment_symbol', SentimentSymbolExtractor()),
           ('lexicon', LexiconExtractor()),
           ('laugh', LaughExtractor()),
           ('character', CharacterExtractor()),
           ('text', TextExtractor())
    ])),
    ("fs", SelectKBest(score_func=chi2, k=500)),
    ('classifier', LinearSVC(C=0.5))
])


pipeline.fit(message_train, label_train)
y_prediction = pipeline.predict(message_test)

report = classification_report(label_test, y_prediction, digits=4)

print(report)
print(accuracy_score(label_test, y_prediction))

train = Counter(label_train)
print('Total train', train)

test = Counter(label_test)
print('Total test', test)

total = Counter(label)
print('Total', total)


def polarity(text):
    prediction = pipeline.predict(text)
    return prediction

```

*improved_model.py*

```python
import csv
from sklearn.model_selection import train_test_split

class DatasetHelper:

    @staticmethod
    def general_tass_to_list(filename):
        data = []
        # with os.listdir(filename) as file:
        with open(filename, 'r', encoding='utf-8') as file:
            for line in file:
                clear = line.replace('\t', ':').replace('\n', ' ').replace('"', '').split(sep=':')
                data.append(clear)

        return data

    @staticmethod
    def gold_standard_to_dict(filename):
        with open(filename, 'r') as csvfile:
            reader = csv.reader(csvfile, delimiter='\t')
            data = {rows[0]: rows[1] for rows in reader}

        return data

    @staticmethod
    def list_to_csv(data, filename):
        with open(filename, 'w', encoding='utf-8') as csvfile:
            writer = csv.writer(csvfile, delimiter=',', lineterminator='\n', quoting=csv.QUOTE_NONNUMERIC)
            writer.writerows(data)

    @staticmethod
    def generate_train_test_subsets(data, size):
        codes = [d[0] for d in data]
        labels = [d[1] for d in data]
        codes_train, codes_test, labels_train, labels_test = train_test_split(codes, labels, train_size=size,
                                                                              random_state=42)
        train_data = [d for d in data if d[0] in codes_train]
        test_data = [d for d in data if d[0] in codes_test]
        return train_data, test_data

    @staticmethod
    def csv_to_lists(filename):
        messages = []
        labels = []
        with open(filename, 'r', encoding='utf-8') as csvfile:
            reader = csv.reader(csvfile, delimiter=',')
            for row in reader:
                messages.append(row[0])
                labels.append(row[1])
        return messages, labels

data = []
data.extend(DatasetHelper.general_tass_to_list('datasets/dataset.txt'))

train, test = DatasetHelper.generate_train_test_subsets(data, size=0.80)

DatasetHelper.list_to_csv(data, 'datasets/global_dataset.csv')
DatasetHelper.list_to_csv(train, 'datasets/train_dataset.csv')
DatasetHelper.list_to_csv(test, 'datasets/test_dataset.csv')
```

*DatasetHelper.py*

```python
import re
from nltk import TweetTokenizer
from nltk.stem.snowball import SnowballStemmer


class Preprocessor:
    NORMALIZE = 'normalize'
    REMOVE = 'remove'
    CHARACTERS = 'characters'
    WEB = 'web'
    URL = 'url'
    LAUGH = 'laugh'
    DIACRITICAL_VOWELS = [('á','a'), ('é','e'), ('í','i'), ('ó','o'), ('ú','u'), ('ü','u')]
    SLANG = [('d','de'), ('[qk]','que'), ('xo','pero'), ('xa', 'para'), ('([xp]q|porq|xq|pq)','porque'),('es[qk]', 'es que'),
             ('fvr','favor'),('(xfa|xf|pf|plis|pls|porfa)', 'por favor'), ('dnd','donde'), ('tb', 'también'),
             ('(tq|tk)', 'te quiero'), ('(tqm|tkm)', 'te quiero mucho'), ('x','por'), ('\+','mas'), ('(sa|sip)', 'si')]

    _stemmer = SnowballStemmer('spanish')
    _tokenizer = TweetTokenizer().tokenize

    def __init__(self, text_features=None, stemming=False):
        self._text_features = text_features
        self._stemming = stemming

    def preprocess(self, message):
        # convert to lowercase
        message = message.lower()
        # remove numbers, carriage returns old-style method
        message = re.sub(r'[\d+|\n]', '', message)
        # remove vowels with diacritical marks
        for s,t in self.DIACRITICAL_VOWELS:
            message = re.sub(r'{0}'.format(s), t, message)
        # remove repeated characters
        message = re.sub(r'(.)\1{2,}', r'\1\1', message)
        # normalized laughs
        message = self.normalize_laughs(message)
        # translate slang
        for s,t in self.SLANG:
            message = re.sub(r'\b{0}\b'.format(s), t, message)

        message = self.process_text_features(message, self._text_features)

        if self._stemming:
            message = ' '.join(self._stemmer.stem(w) for w in self._tokenizer(message))

        return message

    @staticmethod
    def process_text_features(message, text_features):

        message = re.sub(r'[\.\,]https', '. http', message, flags=re.IGNORECASE)
        message = re.sub(r'[\.\,]#|@', '. #', message)
        # message = re.sub(r'[\.\,]@', '. @', message)

        if text_features == Preprocessor.REMOVE:
            # remove mentions, hashtags and urls
            message = re.sub(r'((?<=\s)|(?<=\A))(@|#)\S+', '', message)
            message = re.sub(r'\b(https?:\S+)\b', '', message, flags=re.IGNORECASE)
        elif text_features == Preprocessor.NORMALIZE:
            # normalize mentions, hashtags and urls
            message = re.sub(r'((?<=\s)|(?<=\A))@|#\S+', Preprocessor.CHARACTERS, message)
            # message = re.sub(r'((?<=\s)|(?<=\A))#\S+', Preprocessor.HASHTAG, message)
            message = re.sub(r'\b(https?:\S+)\b', Preprocessor.URL, message, flags=re.IGNORECASE)
        return message

    @staticmethod
    def normalize_laughs(message):
        message = re.sub(r'\b(?=\w*[j])[aeiouj]{4,}\b', Preprocessor.LAUGH, message, flags=re.IGNORECASE)
        message = re.sub(r'\b(juas+|lol)\b', Preprocessor.LAUGH, message, flags=re.IGNORECASE)
        return message

    def __str__(self):
        return "Preprocessor([text_features={0}, stemming={1}])".format(self._text_features, self._stemming)

    def __repr__(self):
        return "Preprocessor([text_features={0}, stemming={1}])".format(self._text_features, self._stemming)
```

*Preprocessor.py*

```python
import re
from itertools import groupby
from sklearn import preprocessing
from sklearn.base import BaseEstimator, TransformerMixin
from util.Preprocessor import Preprocessor


class CharacterExtractor(BaseEstimator, TransformerMixin):

    def __init__(self):
        pass

    def transform(self, data, y=None):
        result = []

        for text in data:
            text = Preprocessor.process_text_features(text, text_features=Preprocessor.REMOVE)
            result.append([
                text.count('!'),
                self._max_consecutive_equals_characteres(text),
                len(re.findall(r'[A-Z]', text))
            ])

        return preprocessing.normalize(result)

    def fit(self, df, y=None):
        return self

    def _max_consecutive_equals_characteres(self, text):
        if len(text) == 0:
            return 0

        groups = groupby(text)
        return max(num for char, num in [(char, sum(1 for _ in group)) for char, group in groups])
```

*CharacterExtractor.py*

```python
import io
from nltk import TweetTokenizer
from nltk.util import ngrams
from sklearn import preprocessing
from sklearn.base import BaseEstimator, TransformerMixin
from util.Preprocessor import Preprocessor


class LexiconExtractor(BaseEstimator, TransformerMixin):

    NGRAM_LENGTH = 3
    REVERSE_WORDS = ['no', 'ni', 'tampoc', 'ningun']

    _tokenizer = TweetTokenizer()
    _preprocessor = Preprocessor(text_features=Preprocessor.REMOVE, stemming=True)

    def __init__(self):
        self._neg_words = self.file_to_list('lexicon/negative_words.txt')
        self._pos_words = self.file_to_list('lexicon/positive_words.txt')

    def transform(self, data, y=None):
        result = []

        for text in data:
            text = self._preprocessor.preprocess(text)
            result.append(self.count_polarity_words(text))

        return preprocessing.normalize(result)

    def count_polarity_words(self, text):
        num_pos_words = 0
        num_neg_words = 0

        list_ngrams = list(ngrams(self._tokenizer.tokenize(text), self.NGRAM_LENGTH, pad_left=True))

        for ngram in list_ngrams:
            pre_words = ngram[:self.NGRAM_LENGTH-1]
            word = ngram[self.NGRAM_LENGTH-1]

            if word in self._pos_words:
                if any(w in pre_words for w in self.REVERSE_WORDS):
                    num_neg_words += 1
                else:
                    num_pos_words += 1

            elif word in self._neg_words:
                if any(w in pre_words for w in self.REVERSE_WORDS):
                    num_pos_words += 1
                else:
                    num_neg_words += 1

        return [num_pos_words, num_neg_words]

    def fit(self, df, y=None):
        return self

    def file_to_list(self, filename):
        return io.open(filename).read().splitlines()
```

*LexiconExtractor.py*

```python
from sklearn.base import BaseEstimator, TransformerMixin
from util.Preprocessor import Preprocessor


class LaughExtractor(BaseEstimator, TransformerMixin):

    def __init__(self):
        pass

    def transform(self, data, y=None):
        result = []

        for text in data:
            text = Preprocessor.normalize_laughs(text)
            result.append([text.count(Preprocessor.LAUGH)])

        return preprocessing.normalize(result)

    def fit(self, df, y=None):
        return self
```

*LaughExtractor.py*

```python
from sklearn import preprocessing
from sklearn.base import BaseEstimator, TransformerMixin
from util.Preprocessor import Preprocessor


class TextExtractor(BaseEstimator, TransformerMixin):

    def __init__(self):
        pass

    def transform(self, data, y=None):
        result = []

        for text in data:
            text = Preprocessor.process_text_features(text, text_features=Preprocessor.NORMALIZE)
            result.append([text.count(Preprocessor.CHARACTERS),
                           text.count(Preprocessor.URL),
                           text.count(Preprocessor.WEB)])

        return preprocessing.normalize(result)

    def fit(self, df, y=None):
        return self
```

*TextExtractor.py*

```python
# -*- coding: utf-8 -*-
from datetime import datetime
import pandas as pd
import re, os
import numpy as np

# this is only for seeing output in terminal
pd.set_option('display.max_rows', 20)
pd.set_option('display.max_columns', 7)
pd.set_option('display.width', 1000)

user_file = 'C:/Users/Lethy/PycharmProjects/SAIA/datasets/jabber_cifrado.csv'
data = []

# # Read User file
if os.path.exists(user_file):
    prev_data = pd.read_csv(user_file, names=['id_conversation', 'id_sender', 'id_sender_client',
                                              'id_receiver', 'id_receiver_client', 'time', 'message'])
    prev_data.replace([np.inf, -np.inf], np.nan)
    prev_data.dropna(inplace=True)
    pattern = re.compile(r"(?<!\d)\d{9}(?!\d\A)")  # tambien se puede usar r"[0-9]{9}\A"
    for row_index, rows in prev_data.iterrows():
        id_conversation = rows['id_conversation']
        if pattern.match(str(id_conversation)):
            id_sender = rows['id_sender']
            id_receiver = rows['id_receiver']
            time = rows['time']
            time = datetime.fromtimestamp(time // 1000)
            message = rows['message']
            data.append([id_conversation, id_sender, id_receiver, time, message])

post_data = pd.DataFrame(data, columns=("Conversation", "Sender", "Receiver", "Time", "Message"))

post_data['Date'] = [d.date() for d in post_data['Time']]
post_data['Hours'] = [d.time() for d in post_data['Time']]

indices = post_data.index.tolist()
np.random.shuffle(indices)
indices = np.array(indices)
text_message = post_data.reindex(index=indices)
text_message.to_csv("datasets/data.csv", index=False)

print(post_data)
```

*preProcesing.py*

```python
import pandas as pd
import numpy as np
import os, re
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords

csv = 'C:/Users/Lethy/PycharmProjects/SAIA/datasets/data.csv'
user = '€äxçßçþ¬'
pd.set_option('display.max_rows', 20)
pd.set_option('display.max_columns', 12)
pd.set_option('display.width', 1000)

if os.path.exists(csv):
    file = pd.read_csv(csv, names=['Conversation', 'Sender','User', 'Time',
                                   'Message', 'Date', 'Hours'])
    group = []
    file_name = file.loc[file['Sender'] == user]
    df_grouped = file.groupby(file_name['Sender'], as_index=False)
    for group_name, df_group in df_grouped:
        for row_index, rows in df_group.iterrows():
            conversation = rows['Conversation']
            receiver = rows['User']
            time = rows['Time']
            message = rows['Message']
            date = rows['Date']
            hours = rows['Hours']
            group.append([conversation, receiver, time, message, date, hours])
    df = pd.DataFrame(group, columns=['Conversation', 'Receiver', 'Time',
                                      'Message', 'Date', 'Hours'])


    # Message words
    df['Message'] = df['Message'].str.replace('\'(.*?): ', '')
    df['Words'] = df['Message'].str.strip().str.split('[\W_]+')
    # Word length
    df['Word Length'] = df['Words'].apply(len) - 1
    # Get the Length of Message
    df['Message Characters'] = df['Message'].map(str).apply(len) - 3
    df['ListofCleanWords'] = np.empty((len(df), 0)).tolist()
    df['CleanWordsasText'] = np.empty(len(df), dtype=str)
    stop_words_sp = set(stopwords.words('spanish'))
    stop_words_en = set(stopwords.words('english'))
    stop_words = stop_words_en.union(set(stopwords.words('spanish')))
    lemmatizer = WordNetLemmatizer()

    for x in range(len(df)):
        listOfWords = [lemmatizer.lemmatize(w, pos='v') for w in
                       re.sub("[^a-zA-Z]", " ", df['Message'][x]).lower().split()
                       if not lemmatizer.lemmatize(w, pos='v') in stop_words]
        df.at[x, 'ListofCleanWords'] = listOfWords
        txt = ' '.join(listOfWords)
        txt = txt if txt is not np.nan else ' '
        df.at[x, 'CleanWordsasText'] = txt

    indices = df.index.tolist()
    np.random.shuffle(indices)
    indices = np.array(indices)
    data_extractor = df.reindex(index=indices)
    data_extractor.to_csv('datasets/conversation_user.csv', index=False)
    print(df)
```

*data_extractor.py*

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
from PIL import Image
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer

# Media de nº de letras por tamaño de documento
def media(data):
    data['docLen'] = data['Message'].apply(lambda words: len(words.split(" ")))
    max_seq_len = np.round(data['docLen'].mean() + data['docLen'].std()).astype(int)
    sns.distplot(data['docLen'], hist=True, kde=True, color='b', label='doc len')
    plt.axvline(x=max_seq_len, color='k', linestyle='--', label='max len')
    plt.title('Media de Nº de palabras por tamaño de documento\n')
    plt.legend()
    plt.savefig('img/media.png')
    return plt.show()




# Palabras más frecuentes según la categoria
def words_(data, title):
    plt.figure(figsize=(10,10))
    cloud_mask = np.array(Image.open("img/cloud-art.jpg"))
    wc = WordCloud(background_color="white", mask=cloud_mask)
    dt = data.loc[data.loc[:,'Score'].str.contains(title)]
    wc.generate(str(" ".join(dt.CleanWordsasText.values)))
    plt.title(title, fontsize=50)
    plt.imshow(wc.recolor(colormap='twilight_shifted', random_state=17), alpha=0.98)
    plt.axis('off')
    plt.savefig('img/'+title+'.png')
    return plt.show()


# % de comentarios por clase
def porcent(total):
    labels = ('Neutral', 'Negative', 'Positive')
    y_pos = np.arange(len(labels))
    plt.bar(y_pos, total, align='center', alpha=0.5)
    plt.xticks(y_pos, labels)
    plt.title('% de comentarios por clases \n')
    plt.ylabel('%')
    plt.savefig('img/porcent.png')
    return plt.show()


# Distribution of Users with Message Count Bar Chart
def bar_chart(users):
    ax = users.plot(kind='bar',
                    color=['red', 'gold', 'skyblue', 'green', 'orange', 'teal', 'cyan', 'lime', 'orangered',
                           'aqua'],
                    fontsize=12)
    ax.set_title("Cantidad de mensajes por usuario\n", fontsize=18)
    ax.set_xlabel("Users", fontsize=12)
    ax.set_ylabel("Messages", fontsize=12)
    plt.savefig('img/messages.png')
    return plt.show()


# Date and User chat Bar Chart
def date_users_bar_chart(date):
        ax = date.plot(kind='bar',
                       color=['red', 'gold', 'skyblue', 'green', 'orange', 'teal', 'cyan', 'lime',
                              'orangered', 'aqua'],
                       fontsize=18)
        ax.set_title("Frecuencia del chat del usurario en el tiempo \n", fontsize=18)
        ax.set_xlabel("Tiempo", fontsize=12)
        ax.set_ylabel("Messages", fontsize=12)
        plt.savefig('img/distribution.png')
        return plt.show()
```

*graph_analisis.py*

```python

# Date with User Chat Line Graph
def user_line_chart(time):
    ax = time.plot(kind='line', color='green', fontsize=12)
    ax.set_title("Frecuencia del chat del usuario por fechas\n", fontsize=18)
    ax.set_xlabel("Fecha", fontsize=10)
    ax.set_ylabel("Messages", fontsize=10)
    plt.savefig('img/user_chat.png')
    return plt.show()


# Messages Word Length Pie
def pie_chart(word_count):
    fig, ax = plt.subplots()
    explodex = []
    for i in np.arange(len(word_count)):
        explodex.append(0.05)
    ax = word_count.plot(kind='pie', colors=['teal','gold','skyblue','green','orange','red','cyan',
                                             'lime','orangered','aqua'],shadow=True, fontsize=12,
                         autopct='%1.1f%%', startangle=180, pctdistance=0.85, explode=explodex)
    ax.axis('equal')
    ax.set_title("Distribución por cantidad de palabras escritas\n", fontsize=18)
    plt.tight_layout()
    plt.savefig('img/cantwords.png')
    return plt.show()

def user_chat_pie(user):
    fig, ax = plt.subplots()
    explodex = []
    for i in np.arange(len(user)):
        explodex.append(0.05)
    ax = user.plot(kind='pie',
                   colors=['red', 'gold', 'skyblue', 'green', 'orange', 'teal', 'cyan', 'lime', 'orangered',
                           'aqua'], fontsize=12, autopct='%1.1f%%', startangle=180, pctdistance=0.85,
                   explode=explodex)
    inner_circle = plt.Circle((0, 0), 0.70, fc='white')
    fig = plt.gcf()
    fig.gca().add_artist(inner_circle)
    ax.axis('equal')
    ax.set_title("Distribución por usuarios\n", fontsize=18)
    plt.tight_layout()
    plt.savefig('img/pie.png', bbox_inches='tight')
    return plt.show()


# Most Frequent words
def most(data):
    cv = CountVectorizer()

    bow = cv.fit_transform(data['ListofCleanWords'])
    word_freq = dict(zip(cv.get_feature_names(), np.asarray(bow.sum(axis=0)).ravel()))
    word_counter = Counter(word_freq)

    words_to_remove = ('uci', 'UCI', 'css3', 'python', 'java', 'dl', 'si', 'sip', 'pa', 'sa','dm', 'ok')
    words_to_remove = [x.lower() for x in words_to_remove]

    for k in words_to_remove:
        word_counter.pop(k, None)

    word_counter_df = pd.DataFrame(word_counter.most_common(20), columns=['word', 'freq'])
    fig, ax = plt.subplots(figsize=(12, 10))
    aa = sns.barplot(x="word", y="freq", data=word_counter_df, palette="PuBuGn_d", ax=ax)
    aa.set_xticklabels(aa.get_xticklabels(), rotation=90)
    plt.title("Ocurrencias de palabras \n ", fontsize=18)
    plt.ylabel('Nº Ocurrencias', fontsize=18)
    plt.xlabel(' Palabras ', fontsize=18)
    plt.savefig('img/mostword.png')
    return plt.show()
```

*graph_analisis.py*

```python
from collections import Counter
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from improved_model import polarity
from case.graph_analisis import media, porcent, words, bar_chart, date_users_bar_chart, user_line_chart, \
    user_chat_pie, pie_chart, most, pie_sentiment

pd.set_option('display.max_rows', 20)
pd.set_option('display.max_columns', 20)
pd.set_option('display.width', 1000)

data_selc = pd.read_csv("datasets/conversation_user.csv", encoding='utf-8', parse_dates=['Date',])
data_selc = data_selc.dropna()
data_selc.reset_index(inplace=True)

for x in range(len(data_selc)):
    data_selc.at[x, 'ListofCleanWords'] = data_selc["ListofCleanWords"][x]
    data_selc.at[x, 'CleanWordsasText'] = str(data_selc["CleanWordsasText"][x])
    # data1.at[x, 'Score'] = str(data1['Score'][x])

data_selc['Score'] = polarity(data_selc['CleanWordsasText'])
data_selc['Positive'] = data_selc['Score'].str.count('positive')
data_selc['Negative'] = data_selc['Score'].str.count('negative')
data_selc['Neutral'] = data_selc['Score'].str.count('neutro')

score = np.array(data_selc['Score'])

leng =len(score)
counter = Counter(score)
dictionary = dict(counter)

def cant(d, s):
    for k, v in d.items():
        if s in k:
            return v


sum_pos = cant(dictionary, 'positive')
print('Total de mensajes calsificados como positive', sum_pos)
sum_neg = cant(dictionary, 'negative')
print('Total de mensajes calsificados como negative', sum_neg)
sum_neu = cant(dictionary, 'neutro')
print('Total de mensajes calsificados como neutral', sum_neu)

total = np.array([sum_neu, sum_neg, sum_pos]) / len(score) * 100

# visualizaciones
by = data_selc.groupby([pd.Grouper(key='Date', freq='D'), 'Score']).size().unstack('Score')
by.plot()
plt.savefig('img/plot.png')
plt.show()
porcent(total)
most(data_selc)
media(data_selc)
words(data_selc, 'positive')
words(data_selc, 'negative')
words(data_selc, 'neutro')
```

*data_analisis.py*

```python

# Distribution of Users with Message Count Bar Chart
users = data_selc.groupby('Receiver')['Receiver'].count().nlargest(15)
bar_chart(users)

# # Date and User chat Bar Chart
date=data_selc.groupby('Hours')['Hours'].count().nlargest(15)
days_count=date[0:]
date_users_bar_chart(days_count)

#Date with User Chat Line Graph
time = data_selc.groupby('Date')['Date'].count().nlargest(15)
user_line_chart(time)

# User Chats Pie Chart
user_d = data_selc.groupby('Receiver')['Receiver'].count().nlargest(15)
user_chat_pie(user_d)

# Messages Word Length Pie
word_count=data_selc.groupby(['Receiver'])['Word Length'].sum().nlargest(15)
pie_chart(word_count)

score = data_selc.groupby(['Score'])['Score'].count().nlargest(15)
pie_sentiment(score)
#
indices = data_selc.index.tolist()
np.random.shuffle(indices)
indices = np.array(indices)
analisis = data_selc.reindex(index=indices)
analisis.to_csv("datasets/conversation user score", index=False)

print(data_selc)
print(by)
```

*data_analisis.py*