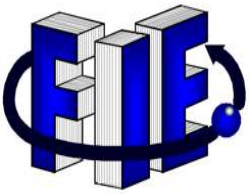

Universidad Michoacana de San Nicolás de Hidalgo

Facultad de Ingeniería Eléctrica

ANÁLISIS DE SENTIMIENTO EN TEXTOS DE TWITTER EN ESPAÑOL: EVALUACIÓN DE CLASIFICADORES DE TEXTO

TESIS

Que para obtener el grado de
INGENIERO EN COMPUTACIÓN

Presenta
Roberto Baldemar Sandoval Esquivel

Asesor de Tesis
M.I. Moisés García Villanueva

Morelia Michoacán, Abril 2017

Agradecimientos

Este trabajo es un escalón más que debo subir en este largo camino llamado vida, y estoy enormemente agradecido con todas aquellas personas que me han acompañado a lo largo de estos 23 años y que de alguna u otra forma me han extendido la mano para ayudarme a seguir creciendo como persona más y más, dándome una palabra de aliento o apoyo, las cuales fueron de suma importancia para mi y poder seguir luchando para alcanzar la meta.

Agradezco principalmente a mis señores padres Baldemar Sandoval Díaz y Ma. de Lourdes Esquivel Figueroa, no solo por haberme dado el regalo de la vida, también porque son las personas que me han acompañado en todo momento en mi vida y con su gran apoyo, cariño y comprensión me han enseñado cosas invaluables que nunca hubiera podido enseñarme toda mi formación académica, a estos seres maravillosos que se han sacrificado día a día para darme cariño, salud, alimentación, techo, ropa y educación. Por enseñarme todo a lo largo de mi vida, y aprender el gran valor de la humildad.

Agradezco a mi hermana Guadalupe Amayrani Sandoval Esquivel, porque es una gran compañía, que con ella he podido llorar, reír, correr, jugar, en fin compartir de todo y darme su apoyo incondicional y estar en todo momento conmigo dándome un empujón siempre que lo necesite.

Agradezco a mi querida Facultad de Ingeniería Eléctrica de la Universidad Michoacana de San Nicolás de Hidalgo en la cual recibí mi formación académica, en donde encontré grandes amistades y profesores de primer nivel. Entre los cuales quiero destacar y hacer un agradecimiento muy especial para aquella persona sin cuya participación no existiría este trabajo de tesis, al M.I. Moisés García Villanueva por aceptar el dirigir la tesis y orientarme durante el desarrollo de la misma, y agradecer por todo lo que me ha enseñado y con el cual tuve la fortuna de coincidir en el aula de clases.

Agradezco a todas esas personas que conforman mi entorno y que por motivos de límite de espacio, no pude mencionar de forma particular, pero que aún así, forman parte muy importante en mi vida.

Dedicatoria

Este trabajo de tesis la dedico a mis padres: Baldemar Sandoval Díaz y Ma. de Lourdes Esquivel Figueroa, por haberme brindado todo su confianza y apoyo total en esta larga, dura, difícil pero muy bonita aventura. Por todo ese cariño incondicional, y ese apoyo tanto moral como en el ámbito económico en cada momento que lo necesité. Ya que sin su ayuda, esfuerzo y sacrificio no hubiera sido posible la culminación de mi carrera profesional como Ingeniero en Computación. Esto es por ellos y para ellos.

Sin duda también la dedico a mi gran maestro Moisés García Villanueva por todo el tiempo dedicado, por su apoyo, y la sabiduría que logro transmitirme mediante el desarrollo de este trabajo, quien me instruyó aún fuera de las aulas y horarios de clase, y hasta ausente físicamente, por lo cual quiero compartir este logro obtenido.

Resumen

El análisis de sentimientos es una nueva tarea que combina técnicas de minería de texto y Procesamiento del Lenguaje Natural (PLN). Aunque existen ya varios trabajos relacionados con la temática, la mayoría de ellos únicamente usan textos en inglés. Sin embargo, el número de páginas webs, blogs u opiniones vertidas en Internet que usan cualquier idioma, no sólo el inglés, crece exponencialmente.

El crecimiento de internet junto con el desarrollo de la Web 2.0 (Web Social) posibilita que personas de todo el mundo compartan información global. Millones de mensajes aparecen diariamente en los sitios más populares de microblogging, comentarios de noticias en diarios web, blogs, twitter, etc. Los autores de estos mensajes escriben acerca de sus vidas, comparten sus opiniones sobre una variedad de temas y discuten sobre estos. Toda esta información que los usuarios generan en las publicaciones acerca de los productos que utilizan o visión política y religiosa, se vuelve un recurso de gran valor para el análisis de opiniones y sentimientos de la opinión pública.

El estudio de estos temas mediante un seguimiento continuo junto con la determinación sobre los acontecimientos o hechos causales de variaciones en la opinión pública son cruciales a la hora de tomar una decisión. Tanto a nivel de consumidor como de proveedor esta información tiene un gran valor estratégico, que les brinda una tendencia y/o comparativa a través del tiempo.

El objetivo fundamental de esta tesis es proporcionar una referencia de la evaluación que es necesario realizar, con el fin de tener elementos de decisión para un desarrollador o individuo que implemente un sistema clasificador de texto, que además se planea poner en producción. Un aspecto en específico que se plantea, mediante un sistema de clasificación, es determinar la polaridad de mensajes de texto, de la red social Twitter, tweets escritos en español. Se establecen dos clases para determinar la polaridad de los textos: Positiva y Negativa. Por su popularidad y facilidad de acceso, se adopta una aproximación con técnicas de aprendizaje automático para los clasificadores a evaluar. Se hace énfasis, para la evaluación de las técnicas de aprendizaje, en la etapa de pre-procesado de los datos, planteando 4 tipos que permiten verificar el comportamiento de los clasificadores. Además se describen, bajo las condiciones establecidas en la experimentación, algunas circunstancias de los clasificadores que es necesario observar, debido a que el desempeño o comportamiento de los sistemas de

aprendizaje, está sujeto al tipo de preprocesamiento de los datos y al conjunto de datos a utilizar.

Palabras clave: *análisis de sentimiento, minería de opiniones, análisis sintáctico, algoritmos de aprendizaje, Twitter.*

Abstract

The sentiment analysis is a new task that combines text mining techniques and Natural Language Processing (NLP). Although there are several works related to the subject, most of them use texts in English. However, the number of web pages, blogs or opinions spilled on the Internet that use any language, not just English, grows exponentially.

The growth of the internet coupled with the development of Web 2.0 (Social Web) enables people from around the world to share global information. Millions of messages appear daily on the most popular microblogging sites, news commentary in web diaries, blogs, twitter, etc. The authors of these messages write about their lives, share their opinions on a variety of topics and discuss them. All this information that the users generate in the publications about the products that use or political and religious vision, becomes a resource of great value for the analysis of opinions and sentiments of the public opinion.

The study of these issues through continuous monitoring together with the determination of events or causal factors of variations in public opinion are crucial when making a decision. Both at the consumer and supplier level, this information has a great strategic value, which gives them a trend and/or comparative value of the world over time.

The main objective of this thesis is to provide a reference of the evaluation that needs to be done, in order to have decision elements for a developer or individual who implements a text classifier system, which is also planned to put into production. One specific aspect that arises, through a classification system, is to determine the polarity of text messages, from the social network Twitter, that is, twitters written in Spanish. Two classes are established to determine the polarity of the texts: Positive and Negative. Because of its popularity and ease of access, it adopts an approach with automatic learning techniques for the classifiers to be evaluated. For the evaluation of the learning techniques, emphasis is placed on the preprocessing stage of the data, proposing 4 types that allow to verify the behavior of the classifiers. In addition, some circumstances of the classifiers that need to be observed are described, under the conditions established in the experimentation, because the performance or behavior of the learning systems is subject to the type of preprocessing of the data and to the set of data to use.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Símbolos

API	**A**pplication **P**rogramming **I**nterface (Interfaz de Programación de Aplicaciones)
BDD	**B**ase **D**e **D**atos
CSS	**C**ascading **S**tyle **S**heets (Hojas de Estilo en Cascada)
HTML	**H**yper**T**ext **M**arkup **L**anguage (Lenguaje de Marcado de Hiper-Texto)
ID	**I**dentifier **U**nique (Identificador Único)
idf	**i**nverse **d**ocument **f**requency (frecuencia inversa de documento)
IPTC	**I**nternational **P**ress **T**elecommunications **C**ouncil (Consejo Internacional de Telecomunicaciones de Prensa estándar)
JSON	**J**ava**S**cript **O**bjet **N**otation (Notación Literal de Objetos de JavaScript)
K-nn	**K**-**n**earest **n**eighbors (K-vecinos más cercanos o aprendizaje basado en instancias)
LLC	**L**ogical **L**ink **C**ontrol (Control de Enlace Lógico)
MAP	**M**aximum **A** **P**osteriori (Máximo a Posteriori)
ML	**M**aximum **L**ikelihood (Máxima Verosimilitud)
NB	**N**aive **B**ayes (Bayes Ingenuo)
NLP	**N**atural **L**anguage **P**rocessing (Procesamiento de Lenguaje Natural)
OAuth	**O**pen **A**uthorization (Autorización segura de una Api)
OM	**O**pinion **M**ining (Minería de Opinión)
SVM	**S**upport **V**ector **M**achine (Máquina de Vectores de Soporte)
tf	**t**erm **f**requency (frecuencia del término en el documento)
URL	**U**niform **R**esource **L**ocator (Identificador de Recursos Uniforme)
Web	**W**ord **W**ide **W**eb (Páginas Web)

Capítulo 1

Introducción

1.1. Identificación del problema

Con la explosión de la Web 2.0 el auge en los últimos años de los blogs, los foros y las redes sociales, ha hecho que millones de usuarios utilicen estos recursos para expresar sus opiniones sobre toda una variedad de temas. La diversidad y cantidad de críticas presentes en la web resultan de gran utilidad a empresas y vendedores, que ven en ellas un mecanismo para conocer de primera mano cómo sus artículos son percibidos por los consumidores. Esta situación ha despertado un gran interés a nivel empresarial, ya que se ve en estos recursos un mecanismo para conocer, de manera eficaz y global, el punto de vista de consumidores sobre una gran variedad de temas. Sin embargo, el análisis manual de este tipo de medios no es una solución viable, dado el flujo ingente de opiniones que en ellos se expresan. Los beneficios asociados a conocer toda esta información, sumados a la complejidad técnica del análisis de las opiniones, han provocado que se hayan comenzado a demandar soluciones capaces de monitorizar este flujo ingente de reseñas. A este respecto, la minería de opiniones (OM, por sus siglas en Inglés Opinion Mining), conocida también como análisis del sentimiento; es una reciente área de investigación centrada en tareas para determinar automáticamente si en un texto se opina o no, o si la polaridad o sentimiento que se expresa en él es positiva, negativa o mixta. También es útil de cara a la extracción automática de características, lo que permite conocer la percepción que se tiene sobre aspectos concretos de un tema [5, 6].

Así pues, la OM trata de clasificar los documentos en función de lo que expresa su autor. Esta nueva disciplina que combina Procesamiento de Lenguaje Natural (NLP por sus siglas en Inglés Natural Language Processing) y minería de textos, incluye una gran cantidad de tareas que han sido tratadas en mayor o menor medida [7]. Una de las principales aplicaciones consiste en determinar la polaridad de las opiniones a nivel de documento, frase o característica. De esta manera, podemos clasificar binariamente las opiniones en positivas o negativas [8].

La investigación en OM es una disciplina reciente concerniente a la recuperación de opiniones expresadas en un documento y no sobre el tema del mismo como es el caso de la recuperación de información. Más específicamente está relacionado con la opinión de un autor expresado en un documento o texto, como pueden ser blogs, micro-blogs, noticias, comentarios, etcétera [9]. Este análisis de sentimientos conlleva algunos desafíos, entre ellos, determinar si cada segmento de texto (sentencia, párrafo o sección) es una opinión o no; identificar quién expresa la opinión (una persona, organización, etc.) y determinar si la opinión es positiva, negativa o neutra, o de acuerdo a un conjunto de categorías preestablecidas [10].

Teniendo en cuenta la riqueza del lenguaje humano y su gran poder expresivo y ambigüedad inherente al mismo, el problema de la clasificación de sentimientos no es trivial [10], por lo que se hace necesario realizar compendios de datos para la experimentación y documentar el análisis de las técnicas involucradas que permitan interpretar los resultados para la toma de decisión en la implementación de estos sistemas.

Las aplicaciones en donde puede ser crucial el análisis de los sentimientos pueden ser:

- Resúmenes de opiniones[11].
- Opiniones de libros o películas [12, 13].
- Análisis de opiniones políticas: Análisis de candidatos políticos [14, 15], Gobierno Electrónico [16] para analizar impacto de decisiones.
- Análisis del impacto de productos, servicios y marcas[14].

Los recursos lingüísticos para OM definen algunas propiedades relacionadas a los sentimientos. Los avances sobre este tópico tratan con tres tareas principales:

- Determinación de la orientación del término: positivo, negativo, neutro.
- Determinación de la subjetividad de un término, si un término tiene una naturaleza subjetiva u objetiva.
- Determinación de la fuerza de la determinación del término (orientación o subjetividad), como el grado de positividad o negatividad del término, es decir, asignar en una escala el grado de polaridad del término.

Las investigaciones sobre OM han tomado tres líneas de investigación interrelacionadas [7]:

- Desarrollo de recursos lingüísticos para el análisis de sentimientos tal como corpus léxico anotado manualmente;
- Implementación de diferentes algoritmos para el análisis del texto y clasificación de acuerdo a su orientación semántica y subjetiva;
- Extracción de opiniones del texto, incluyendo diferentes tipos de relaciones con contenido asociado.[10]

Por otra parte, y a pesar de que las opiniones y comentarios compartidos en Internet no tienen restricción en cuanto al idioma utilizado, la gran mayoría de la investigación llevada a cabo relacionada con la OM se centra casi exclusivamente en textos escritos en inglés. Sin embargo, cada vez son más los textos que utilizan otros idiomas. Si bien el inglés es la lengua predominante en Internet, hay otros idiomas como el chino o el español que cada vez tienen más presencia en la red [8].

1.2. Antecedentes

Si bien una clasificación de textos se puede realizar manualmente, tarea que en lingüística se efectúa fundamentalmente en base a criterios multiniveles (lingüísticos, textuales, pragmáticos y funcionales) y que se conoce como tipologización textual [17, 18, 19],

también es deseable alcanzar un nivel de automatización de estos procedimientos. Esta tarea, desde una aproximación del procesamiento automático del lenguaje, es conocida como clasificación automática de documentos, donde el estándar es construir y usar las llamadas máquinas de aprendizaje supervisado. El proceso de crear una clasificación automática de textos consiste en descubrir variables que sean útiles en la discriminación de los textos que pertenecen a clases preexistentes distintas. En particular, los clasificadores (programas que ejecutan algoritmos de clasificación) son entrenados en un grupo de documentos, previamente clasificados y etiquetados acorde a algún criterio particular (tema, materia, origen, etc.), conformando una clase. De esta manera, el objetivo de estos clasificadores es decidir en qué categoría debe ir cada texto nuevo, partiendo de un esquema de clasificación previo [20]. También se dice que la clasificación o categorización automática de documentos puede ser entendida como una tarea en la cual, en base a la identificación por medios matemático-estadísticos, un documento nuevo es asignado a una clase particular de documentos pre-existentes [21].

Es importante destacar que la clasificación automática de documentos surge de los estudios realizados en recuperación de información [22, 23, 24]. Para ello, se ha utilizado mayoritariamente corpus textuales en lengua inglesa. En menor medida, y solo en los últimos años, estas técnicas han sido aplicadas a corpus en español [25, 20, 26, 27].

En términos prácticos, la utilidad de la clasificación automática de documentos se basa en la posibilidad de poder efectuar una adecuada recuperación de documentos no conocidos, asumiendo que aquellos textos que tratan, por ejemplo, de la misma materia están clasificados juntos, o en sectores cercanos (un ejemplo de la herramienta desarrollada se observa en el uso de procesos para los filtros de correo basura o spam) [28].

Diversas técnicas han sido propuestas, desde hace ya algunos años [29, 30, 22]. Buena parte de tales técnicas se basan en la utilización de medidas de semejanza (o de disparidad, dependiendo del punto de vista) entre dos documentos.

En síntesis, la clasificación automática de documentos puede concebirse como un proceso de aprendizaje matemático-estadístico, durante el cual un algoritmo implementado computacionalmente capta las características que distinguen cada categoría o clase de documentos de las demás, es decir, aquellas que deben poseer los documentos para pertenecer a

esa categoría. Estas características no tienen porqué indicar de forma absoluta e inequívoca la pertenencia a una clase o categoría, sino que más bien lo hacen en función de una escala o graduación. De esta forma, por ejemplo, documentos que posean una cierta característica tendrán un factor de posibilidades de pertenecer a determinada clase, de modo que la acumulación de dichas características arrojará un resultado que consiste en un coeficiente asociado a cada una de las clases ya conocidas. Este coeficiente lo que expresa en realidad es el grado de confianza o certeza de que el documento en cuestión pertenezca a la clase asociada o la categoría resultante [28].

1.3. Justificación

Existe una gran necesidad de desarrollar aplicaciones funcionales de BDD e Inteligencia Artificial, para ayudar a tener un mejor manejo de textos que se van generando con el paso del tiempo de una manera muy rápida, y se requiere de sistemas para ayudar a clasificar la opinión de los textos o artículos de una manera adecuada, esto tiene un gran valor económico, debido a que permite conocer las opiniones de los clientes en foros de servicios o productos. En una red social, permite estimar la opinión de diferentes sucesos que se están desarrollando en la actualidad.

1.4. Objetivo

Este trabajo tiene como objetivo el desarrollar una aplicación Web para categorizar y clasificar texto mediante el uso de técnicas de aprendizaje automático, abordar la problemática de clasificar la opinión de textos de una red social (Twitter) de textos en español, empleando técnicas de Aprendizaje de Máquina, en específico: SVM, Naive Bayes y K-nn.

1.4.1. Objetivos particulares

- Generar conjuntos de datos etiquetados de la red social twitter para la experimentación y generación de modelos de aprendizaje.

- Generar modelos de aprendizaje para cada clasificador.
- Implementar un sistema para clasificar textos en tiempo real en la Web.
- Evaluar en forma experimental la precisión de los clasificadores en relación a la cantidad de documentos utilizados en el entrenamiento para producir los modelos de aprendizaje.
- Comparar el desempeño obtenido en base al tipo de preprocesamiento de los datos.
- Realizar las pruebas y análisis de los resultados.

1.5. Descripción de los capítulos

- En el capítulo 1 se da una breve introducción a este trabajo. Se mencionan antecedentes, se plantea el problema y se establecen los objetivos generales y particulares de la tesis. Se señala además una descripción de cada capítulo.
- En el capítulo 2 se establece el marco teórico, en donde se describen brevemente las técnicas utilizadas para realizar este trabajo, señalando el Modelo Vectorial para representar los textos y los diferentes algoritmos a utilizar en este trabajo para la clasificación de documentos.
- En el capítulo 3 se describe el mecanismo de como se conformaron los conjuntos de datos (corpus) y el etiquetado de la polaridad de los datos que se emplearán en la parte experimental.
- En el capítulo 4 se describe la implementación del sistema de opinión en la Web, ilustrando la forma de uso.
- En el capítulo 5 se establecen las condiciones a las que se sometieron los clasificadores en la parte experimental, mostrando las pruebas realizadas e interpretación de los resultados.
- Finalmente en el capítulo 6 se presentan las conclusiones del presente trabajo, así como aportaciones y trabajos futuros.

Capítulo 2

Marco Teórico

2.1. El Modelado Vectorial

Las técnicas empleadas para la clasificación de documentos, como se mencionó, son originarias de los métodos clásicos de recuperación de información, es por ello que antes de describir las técnicas a utilizar en el presente trabajo, debe describirse brevemente el modelo vectorial, ya que, sin duda, es la base conceptual para las técnicas de clasificación actuales [25, 31]. El modelo vectorial fue definido inicialmente por Salton (1968) y es ampliamente usado en operaciones de recuperación de información, así como también en operaciones de categorización automática, filtrado de información, etc. [26, 31].

En esta sección se presenta sucintamente la forma en que es utilizado el modelo vectorial para la recuperación de información. Es de interés destacar este modelo, porque es la manera más sencilla de explicar cómo se llevan a cabo las operaciones matemático-estadísticas que permiten determinar la similitud entre documentos a partir de las palabras contenidas en ellos, empleando métodos en donde la frecuencia de las palabras es el elemento principal, y a partir de ellas clasificar documentos nuevos (tales como los textos de las redes sociales por mencionar algunos) en clases pre-existentes (disciplinas, opiniones o sentimiento).

Según Zazo et al.[26], en el modelo vectorial se intenta recoger la relación de cada documento D_i, de una colección de N documentos, con el conjunto de las m características

de la colección. Formalmente, un documento puede considerarse como un vector que expresa la relación del documento con cada una de esas características. La ecuación (2.1) da cuenta de esta representación vectorial de un documento [28].

$$D_i \rightarrow \vec{d} = (c_{i1}, c_{i2}, ..., c_{im}) \tag{2.1}$$

Se observa que el vector $(\vec{d})$ identifica en qué grado el documento D_i satisface cada una de las m características (c_{im}). En otras palabras en el vector $\vec{d}$, c_{im} es un valor numérico que expresa en qué grado el documento D_i posee la característica m. La noción de característica suele concretarse en la ocurrencia de determinadas palabras o términos en el documento, aunque nada impide tomar en consideración otros aspectos. Respecto de esto último, cabe señalar que este tipo de procedimientos se han utilizado en el reconocimiento de objetos, donde las características son de carácter viso-perceptual (color, forma, etc.) [32].

Si se consideran las palabras como características definitorias del documento, el proceso que debe seguir el sistema de clasificación se inicia con la selección de aquellas palabras útiles que permitan discriminar unos documentos de otros. En este punto, debemos señalar que no todas las palabras contribuyen con la misma importancia en la caracterización del documento. Desde el punto de vista lingüístico aplicado a la recuperación o clasificación de documentos, existen lexemas casi vacíos de contenido semántico, como los artículos, las preposiciones o las conjunciones. Estos lexemas son conocidos como palabras funcionales en la tradición lingüística y como stopwords (palabras vacías) en el procesamiento de lenguaje natural. Estas palabras, que en español comúnmente son entre 200 y 300, son poco útiles para el proceso de clasificación [27]. También son poco importantes aquellas palabras que por su frecuencia de aparición en toda la colección de documentos pierden su poder de discriminación, es por ello que o son eliminadas o son ponderadas con muy bajo peso estadístico.

Además de la eliminación de las palabras funcionales o poco informativas, en el proceso se pueden incluir aplicaciones léxicas como lematización o extracción de raíces, etiquetado de términos, detección de unidades multipalabra, etc. Todo esto permite reducir la cantidad de palabras a considerar en la matriz de análisis, sin embargo, el problema de estas aplicaciones es la pérdida de información relevante (género y número de las palabras),

que puede ser necesario para una clasificación más ajustada a la realidad de los textos.

Una vez seleccionado el conjunto de términos caracterizadores de la colección de documentos, es necesario calcular el valor de cada elemento del vector del documento (c_{im}). El caso más simple es utilizar una aproximación binaria, de forma que si en el documento D_i aparece el término k, el valor c_{ik} sería 1, y en caso contrario sería 0.

Ahora bien, como se sabe, una palabra puede aparecer más de una vez en el mismo documento y, además, algunas palabras pueden considerarse con más peso estadístico que otras, esto es, más significativas que otras, de forma que el valor numérico de cada uno de los componentes del vector obedece normalmente a cálculos más sofisticados que la simple asignación binaria. Por otro lado, también es importante normalizar los vectores para no privilegiar documentos.

Se han propuesto diversos métodos para calcular el peso de cada término en el vector que representa al documento [23, 24, 33], pero en general, para estimarlos se parte de dos ideas en cierto sentido contrapuestas: si un término se consigna mucho en un documento, aquel es importante para caracterizar el documento. Pero si aparece en muchos documentos de la colección, este término no resulta beneficioso para distinguir un documento de los demás, dado su escaso poder discriminatorio, siendo por tanto poco útil para la recuperación o clasificación de nuevos documentos.

Para determinar la capacidad de representación de un término para un documento dado, se calcula el número de veces que este aparece en dicho documento, obteniéndose la frecuencia del término en el documento (tf por sus siglas en inglés: $term\ frequency$). Por otra parte, si la frecuencia de un término en toda la colección de documentos es extremadamente alta, se opta por eliminarlo del conjunto de términos de la colección. Podría decirse que la capacidad de recuperación de un término es inversamente proporcional a su frecuencia en la colección de documentos (idf por sus siglas del inglés: $inverse\ document\ frequency$). Así, para calcular el peso de cada elemento del vector ($w_i = c_{im}$) que representa al documento, se tiene en cuenta la frecuencia inversa del término en la colección, multiplicándola por la frecuencia del término dentro de cada documento [33], esto se define en la ecuación (2.2):

$$w_i = (tf_i) \times (idf_i) \tag{2.2}$$

Al respecto, Salton y Buckley[24] experimentaron con más de 200 sistemas de cálculo de pesos, pero uno de los más utilizados viene dado en la ecuación (2.3), que expresa el peso del término j en el documento i.

$$w_{ij} = tf_{ij} \times \lg \frac{N}{df_{ij}} \tag{2.3}$$

Donde df_{ij} es el número de documentos en que aparece el término j, y N el número de documentos de la colección.

Una aplicación de este proceso realizado para los documentos es el utilizado por los sistemas automatizados de recuperación o clasificación de documentos (bases de datos, por ejemplo) en el que los usuarios realizan consultas en lenguaje natural. Estas consultas pueden considerarse como un documento más, seguramente bastante breve, aunque no siempre. Así pues, el mecanismo de obtención de pesos también se aplica a las consultas, para de esta manera poder disponer de representaciones vectoriales homogéneas de consultas y documentos, que posibiliten obtener el grado de similitud entre ambos documentos, representados como vectores en un espacio multidimensional.

La resolución de la consulta consiste en establecer el grado de semejanza existente entre el vector que representa a la consulta y el vector que representa a cada uno de los documentos de la colección. Para una consulta determinada, cada documento arrojará un grado de similitud determinado; aquellos cuyo grado de similitud sea más elevado se ajustarán mejor a las necesidades expresadas en la consulta, desde el punto de vista del sistema de recuperación o clasificación de información. No obstante, es el usuario el que debe decidir la relevancia de los documentos recuperados, siendo esta una característica totalmente subjetiva del mismo [31].

El modo más simple de calcular la similitud entre una consulta y un documento, utilizando el modelo vectorial, es realizar el producto escalar de los vectores que los representan en la ecuación (2.5). En la siguiente ecuación se incluye la normalización de los vectores $\vec{u}$ y $\vec{v}$, a fin de obviar distorsiones producidas por los diferentes tamaños de los

documentos. El índice de similitud más utilizado es el coseno del ángulo formado por ambos vectores, el cual está dado por la ecuación (2.4):

$$\cos\theta = \frac{\sum_{i=1}^{m} u_i \cdot v_i}{\sqrt{\sum_{i=1}^{m} u_i^2 \cdot \sum_{i=1}^{m} v_i^2}} \tag{2.4}$$

Para una consulta $Q \rightarrow \vec{q}$, el índice de similitud con un documento D_i está dado por la ecuación (2.5).

$$simil(Q, D_i) = \frac{\sum_{j=1}^{m} q_j d_{ij}}{\sum_{j=1}^{m} q_j^2 \cdot \sum_{j=1}^{m} d_{ij}^2} \tag{2.5}$$

Cabe señalar, que existen otros métodos propuestos para calcular la similitud. Algunos ejemplos son: el coeficiente de emparejamiento (*matching coefficient*) [34], el coeficiente de Dice [35], el coeficiente de Jaccard (o Tanimoto) [36] y el coeficiente de solapamiento (*overlap coefficient*) . Una síntesis con la descripción de estas medidas de similitud puede encontrarse en Jurafsky y Martin[21], Manning y Schütze[31], Tzoukermann, Klavans y Strzalkowski[37].

Ahora bien, en este trabajo se compararán tres métodos de clasificación supervisada, basados en el modelo vectorial: el Bayes Ingenuo (*Multinomial Naïve Bayes*), k vecinos más cercanos (K-nn) y la Máquina de Soporte de Vectores (*Support Vector Machine*), con el fin de conocer cuál de ellas es más eficiente en la clasificación de los textos.

Se han elegido estos métodos, porque, si bien son comunes en la clasificación de documentos, aún no han sido suficientemente probados en corpus textuales en español (algunas excepciones son[25, 27]). Además, porque con estos métodos no se requiere necesariamente un marcaje estructural de los textos de ningún tipo, es decir, los textos no deben ser etiquetados morfosintácticamente; por lo tanto, tienen menor exigencia en cuanto a costo computacional y se pueden implementar más fácilmente en programas de clasificación automática, que sistemas basados en representaciones simbólicas guiadas por la sintaxis [21].

2.1.1. Ejemplo de la representación vectorial

En esta subsección se presenta mediante un ejemplo un poco más de descripción del modelo vectorial, debido a que es el más utilizado en la actualidad en los sistemas de recuperación de información.

Este modelo entiende que los documentos pueden expresarse en función de unos vectores que recogen la frecuencia de aparición de los términos en los documentos (ver ecuación (2.1)). Es posible entonces representar en una matriz los datos (texto), en donde las columnas representan los términos y cada renglón representa un documento de nuestro conjunto de datos. Dichos términos que forman esa matriz serían términos no vacíos, es decir, dotados de algún significado a la hora de recuperar información.

Si nuestro sistema de clasificación de texto contiene los siguientes documentos:

D_1*: es muy triste ver como se están extinguiendo poco a poco los animales.*

D_2*: es terrible ver como sufre, esta situación es triste.*

Su matriz correspondiente dentro del modelo o representación vectorial, eliminando algunas palabras vacías (*es, muy, se, como, a, los, esta*), estará representada como se muestra en la tabla 2.1.

Tabla 2.1: Ejemplo de la matriz de términos y documentos en el modelo vectorial.

	triste	**ver**	**extinguiendo**	**poco**	**animales**	**terrible**	**sufre**	**situación**
D_1	1	1	1	2	1	0	0	0
D_2	1	1	0	0	0	1	1	1

Por medio de un proceso denominado *stemming*, quizá el sistema hubiera truncado algunas de las entradas para reducirlas a un formato de raíz común, pero para continuar con la explicación resulta más sencillo e ilustrativo dejar los términos en su formato normal. En cuanto a las palabras vacías, hemos supuesto que el sistema elimina los determinantes, preposiciones y verbos (“el”, “pasa”, “por”, etc.), presentes en los distintos documentos.

Para la construcción de la matriz de términos y documentos utilizando las definiciones de *tfidf*, para nuestro ejemplo se consideran las siguientes definiciones:

- m = número de términos distintos en la colección de documentos

- tf_{ij} = número de ocurrencias de término t_j en el documento D_i (frecuencia del término o **tf**)
- df_j = número de documentos en que aparece el término j
- idf_j = el $log(N/df_j)$, donde N es el número total de documentos en la colección (frecuencia inversa del documento o **idf**)

El vector para cada documento tiene m características y contiene una entrada para cada término distinto en la colección entera de documentos. Los componentes en el vector se fijan con los pesos (w_i) calculados para cada término en la colección de documentos (ver ecuación 2.3). A los términos en cada documento automáticamente se le asignan pesos basándose en la frecuencia con que ocurren en la colección entera de documentos y en la aparición de un término en un documento particular.

El peso de un término en un documento aumenta si este aparece más a menudo en un documento y disminuye si aparece más a menudo en todos los demás documentos. El peso para un término en un vector de documento es distinto de cero sólo si el término aparece en el documento. Para una colección de documentos grande que consiste en numerosos documentos pequeños, es probable que los vectores de los documentos contengan ceros principalmente. Por ejemplo, una colección de documentos con 10, 000 términos distintos genera un vector 10, 000 dimensiones para cada documento. Un documento dado que tenga sólo 100 términos distintos tendrá un vector de documento que contendrá 9900 ceros en sus componentes.

El cálculo del factor de peso (w) para un término en un documento se define ***como combinación de la frecuencia de término (tf), y la frecuencia inversa del documento (idf)***. Para calcular el valor de la $j-esima$ entrada del vector que corresponde al documento i, de la cuál se emplea la ecuación (2.2) en la que se adapto a la ecuación (2.3). El cálculo de las frecuencias inversas de los términos en los documentos y la posterior aplicación de esta fórmula sobre la matriz de nuestro ejemplo (ver tabla 2.2), proporcionaría la siguiente matriz de pesos.

Cálculo de frecuencias inversas:

$idf(triste) = lg(2/2) = lg(1) = 0$

$idf(ver) = lg(2/2) = lg(1) = 0$

$idf(extinguiendo) = lg(2/1) = lg(2) = 0.301$

$idf(poco) = lg(2/1) = lg(2) = 0.301$

$idf(animales) = lg(2/1) = lg(2) = 0.301$

$idf(terrible) = lg(2/1) = lg(2) = 0.301$

$idf(sufre) = lg(2/1) = lg(2) = 0.301$

$idf(situación) = lg(2/1) = lg(2) = 0.301$

Tabla 2.2: Ejemplo de Matriz de términos y documentos en el Espacio Vectorial con los pesos calculados.

	triste	**ver**	**extinguiendo**	**poco**	**animales**	**terrible**	**sufre**	**situación**
D_1	0	0	0.301	0.301	0.301	0	0	0
D_2	0	0	0	0	0	0.301	0.301	0.301

Los procesos de equiparación de los documentos (D) de la colección con respecto a la consulta (Q) del usuario se realizan mediante el producto escalar (ver ecuación (2.5)). De esta forma, la similaridad de un documento y una consulta, es igual a la suma de los productos de sus pesos (y no se debe olvidar que cada peso representa a un término). La tabla 2.3 muestra este proceso.

Tabla 2.3: Producto escalar de pesos tf-idf.

	triste	**ver**	**extinguiendo**	**poco**	**animales**	**terrible**	**sufre**	**situación**
D_1	0	0	0.301	0.301	0.301	0	0	0
D_2	0	0	0	0	0	0.301	0.301	0.301
Q	0	0	0.301	0	0	0	0	0

El cálculo de la similaridad se aplica a cada uno de los documentos de la colección siguiendo el patrón expuesto en la tabla 2.3. Para el D_1 la similaridad con respecto a la consulta del usuario Q, será diferente que para el D_2. Obsérvese que sólo tienen incidencia aquellos términos presentes tanto en la consulta como en el documento, pues sus pesos se multiplican y se suman sucesivamente al resto.

Ahora corresponde calcular las similitudes existentes entre los distintos documentos (D_1 y D_2) y el vector Q. Hay que multiplicar componente a componente de los vectores y sumar los resultados. El modo más sencillo de obtener la similitud es por medio del producto

escalar de los vectores (es decir, multiplicando los componentes de cada vector y sumando los resultados).

Cálculo de similitudes:

simil(Q, D_1**)** = 0*0 + 0*0 + 0.301*0.301 + 0.301*0 + 0.301*0 + 0*0 + 0*0 + 0*0 = **0.09**

simil(Q, D_2**)** = 0*0 + 0*0 + 0*0.301 + 0*0 + 0*0 + 0.301*0 + 0.301*0 + 0.301*0 = **0**

Con estos valores de similitud, se obtiene la siguiente respuesta: D_1, D_2. Podemos observar en este ejercicio un ejemplo de acierto y un ejemplo de fallo de este modelo, ya que el primero de los documentos recuperados sí responde a la consulta (D_1) y al mismo tiempo los demás no responden adecuadamente (realmente la similitud es muy baja). Casos como el presente, justifican la presencia de documentos no relevantes en la respuesta de los sistemas de recuperación de información y que este esquema básico de alineamiento haya sufrido muchos cambios.

2.2. Algoritmos de Aprendizaje

En esta sección, se explica brevemente las características principales de cada método de clasificación. Cabe comentar que no es la intención de este trabajo profundizar en todos los aspectos referentes a los procedimientos y cálculos estadísticos que conciernen a cada uno de ellos. De este modo, para mayor información se recomiendan las siguientes fuentes: para Bayes Ingenuo [38, 31, 39] y para Máquinas de Soporte de Vectores [2, 40, 41].

2.2.1. Clasificador Bayes Ingenuo (NB)

Los clasificadores bayesianos [42] son clasificadores estadísticos, que pueden predecir tanto las probabilidades del número de miembros de una clase, como la probabilidad de que una muestra dada pertenezca a una clase particular. Este tipo de clasificadores, basados en el teorema probabilístico de Bayes, han demostrado una alta exactitud y velocidad cuando se han aplicado a grandes bases de datos textuales, particularmente en español [39, 27, 43].

Diferentes estudios en los que se han comparado diversos algoritmos de clasificación han determinado que el clasificador NB es comparable en rendimiento a un árbol de decisión y a clasificadores de redes neuronales, procedimientos que son mucho más complejos computacionalmente [39].

A continuación, se explican los fundamentos de los clasificadores bayesianos y, más concretamente, del clasificador NB.

El objetivo de este método de aprendizaje matemático-estadístico es determinar cuál es la mejor hipótesis (la más probable) dado un conjunto de datos pre-existentes. Si denotamos $P(D)$ como la probabilidad a *priori* de los datos y $P(D|h)$ como la probabilidad de los datos dada una hipótesis, lo que queremos estimar es $P(h|D)$, o sea, la probabilidad posterior de h dado ciertos datos conocidos, de aquí la noción de probabilidad condicionada. Esto se puede estimar con el teorema de Bayes con la ecuación (2.6):

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \tag{2.6}$$

Para estimar la hipótesis más probable (MAP por sus siglas en inglés: *maximum a posteriori*) se busca el mayor $P(h|D)$ como se muestra en la ecuación (2.7):

$$\begin{aligned} h_{MAP} &= \arg\max(P(h|D)) \\ &= \arg\max\left(\frac{P(D|h)P(h)}{P(D)}\right) \\ &= \arg\max(P(D|h)P(h)) \end{aligned} \tag{2.7}$$

Ahora bien, como $P(D)$ es una constante independiente de h, se asume que todas las hipótesis son igualmente probables, esto permite entonces concebir la hipótesis de máxima verosimilitud (ML por sus siglas en inglés: maximum likelihood) expresada en la ecuación (2.8):

$$h_{ML} = \arg\max(P(D|h)) \tag{2.8}$$

De modo más particular, el clasificador bayesiano ingenuo se utiliza cuando se quiere clasificar un ejemplo descrito por un conjunto de atributos (aka caracterśiticas c_m)

en un conjunto finito de clases (V). Esto es clasificar un nuevo ejemplo de acuerdo con el valor más probable dado los valores de sus atributos. Así, si se aplica la ecuación (2.8) al proceso de la clasificación, se obtendrá la ecuación (2.9):

$$\begin{aligned} V_{MAP} &= \arg\max(P(v_j|c_1,...,c_m)) \\ &= \arg\max\left(\frac{P(c_1,...,c_m|v_j)P(v_j)}{P(c_1,...,c_m)}\right) \\ &= \arg\max(P(c_1,...,c_m|v_j)P(v_j)) \end{aligned} \tag{2.9}$$

Además, el clasificador NB asume que los valores de los atributos son condicionalmente independientes dado el valor de la clase, por lo que se hace cierta la ecuación (2.10) y con ella la (2.11).

$$P(c_1,...,c_m|v_j) = \prod_i P(c_i|v_j) \tag{2.10}$$

$$P(v_j|c_1,...,c_m) = P(v_j) \times \prod_i P(c_i|v_j) \tag{2.11}$$

En este sentido, con los clasificadores bayesianos ingenuos se asume que el efecto de un valor del atributo en una clase dada es independiente de los valores de los otros atributos. Esta suposición se llama independencia condicional de clase [21, 39, 43]. Ella permite simplificar los cálculos involucrados, siendo por esto que se le considera ingenuo ($Naive$) al método y, por lo mismo, sus resultados deben ser entendidos como una simplificación de la realidad.

2.2.2. La Máquina de Vectores de Soporte (SVM)

La Máquina de Vectores de Soporte (SVM por sus siglas en inglés: Support Vector Machine) es un método de clasificación de datos bastante nuevo, con el que diversos investigadores han conseguido buen desempeño de generalización sobre una amplia variedad de problemas de clasificación, destacando recientemente en problemas de clasificación de textos. En relación a esto último, se le reconoce al método la capacidad de minimizar el error de generalización, es decir, los errores del clasificador sobre nuevos documentos [44, 40, 1, 45].

Particularmente la SVM es apropiada para trabajar con datos multidimensionales, tales como representaciones de vectores en un espacio de documentos textuales. En su formulación estándar se trabaja con problemas de clasificación binaria donde el número de clases es restringido a dos, aunque también se puede utilizar para la clasificación multiclases, a través de la reducción del problema de clasificación a sub-problemas de orden binario [44].

Una tarea normal de clasificación de textos involucra datos para entrenamiento y datos para prueba de un algoritmo a partir de características cuantificables de los textos. Cada unidad textual en el grupo de entrenamiento contiene un valor de clasificación, designado por una etiqueta de clase, y múltiples atributos o rasgos. El objetivo de la SVM, entonces, es producir un modelo que permita predecir los valores de clasificación (identificar la clase) en la etapa de prueba conociendo solo los atributos [1].

En términos geométricos, el problema que resuelve la SVM es identificar una frontera de decisión linear entre dos grupos, a través de una línea que los separe maximizando el espacio del hiperplano, de modo muy similar a lo que se realiza utilizando el análisis discriminante [46, 47]. Sin embargo, la SVM incluye una operación nueva llamada truco de kernel, la que le permite realizar separaciones no lineales de los datos y con ello optimizar la clasificación de los mismos. Así, por ejemplo, en la Figura 2.1, observamos que en un espacio multidimensional las posibilidades de separación de las clases pueden ser múltiples, sin embargo, lo que se necesita es una separación óptima del hiperplano, sustentada por márgenes definidos, que en la práctica serán los vectores de soporte (ver Figura 2.2).

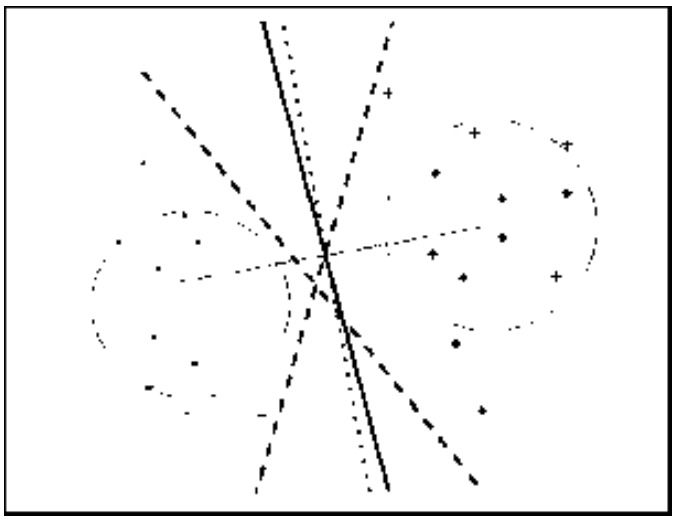

Figura 2.1: Fronteras de decisión lineal alternativas para un problema de clasificación binario[1].

En estas condiciones, una frontera de edición óptima sería aquella que minimice la posterior probabilidad de que un nuevo punto sea mal clasificado y que esta frontera sea el hiperplano ortogonal al segmento que conecta los centros de masa de las dos distribuciones (línea punteada de la Figura 2.1). Claramente, un hiperplano azaroso que solo por casualidad separe los puntos de entrenamiento (línea segmentada en la Figura 2.1) puede estar significativamente lejos de una frontera de separación óptima, aportando una generalización muy pobre ante datos nuevos [1], aumentando exponencialmente este problema en la medida en que aumentan la dimensionalidad del espacio de documentos.

Ante esta situación, Vapnick (2000) [48] propone, en su teoría de aprendizaje estadístico, un hiperplano de separación óptima el cual tiene dos propiedades importantes: es único para cada grupo de datos separables linealmente, y el riesgo asociado de sobreestimación es más reducido que para cualquier otro hiperplano de separación. El margen de separación M del clasificador será la distancia entre el hiperplano de separación y el ejemplo de entrenamiento más cercano. De este modo, el hiperplano de separación óptimo es aquel que tenga el máximo margen. Para calcularlo se comienza con la determinación de la distancia de un punto x del hiperplano de separación (ver Figura 2.2).

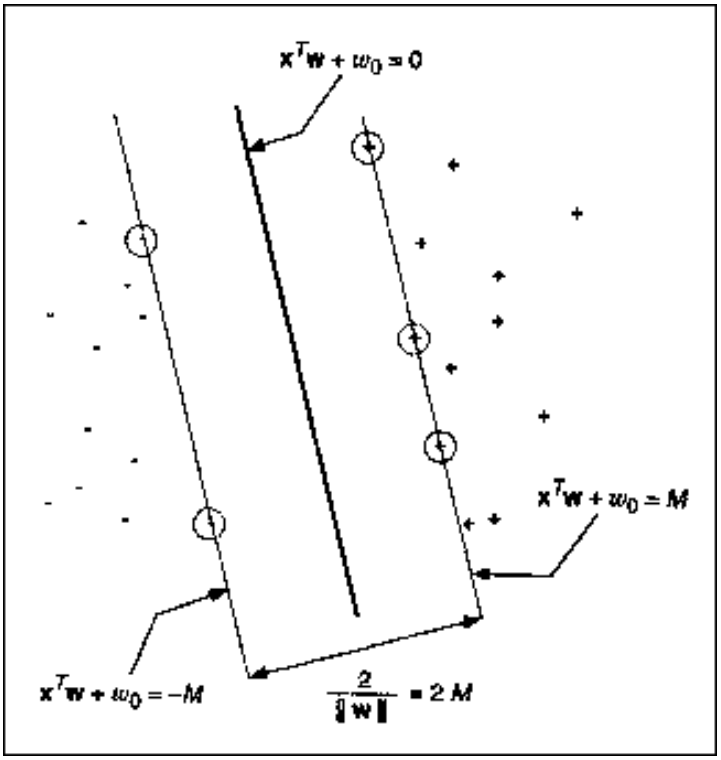

Figura 2.2: Ilustración del hiperplano de separación óptima y sus márgenes. Los datos en círculo indican los vectores de soporte [1].

Cabe señalar que la determinación final de la separación óptima del hiperespacio involucra una serie de pasos entre los que se considera la aplicación de la función de Langrage y el paquete de optimización estandarizada (para la solución del problema de programación cuadrática), el que debe satisfacer las condiciones de Karush-Kuhn-Tucker [48, 2, 1].

Ahora bien, la característica más relevante de la SVM es el uso de las funciones kernel (por ejemplo, lineal, polinominal, función de base radial, sigmoidal) para extender las aplicaciones de la determinación de separación óptima a casos no lineales [2]. Esto se hace traspasando los datos desde el espacio de entrada X a un amplio espacio de características X' mediante una función Φ, y resolviendo el problema de aprendizaje lineal en $X'(\Phi : X \to X')$. La función real Φ no necesita ser conocida, es suficiente tener una **función kernel** $\boldsymbol{k}$ que calcule el producto interno en el espacio de características: $k(X, X') = \langle \Phi(X), \Phi(X') \rangle$[2, 49].

Una ilustración de esto último, es la que se presenta en la Figura 2.3.

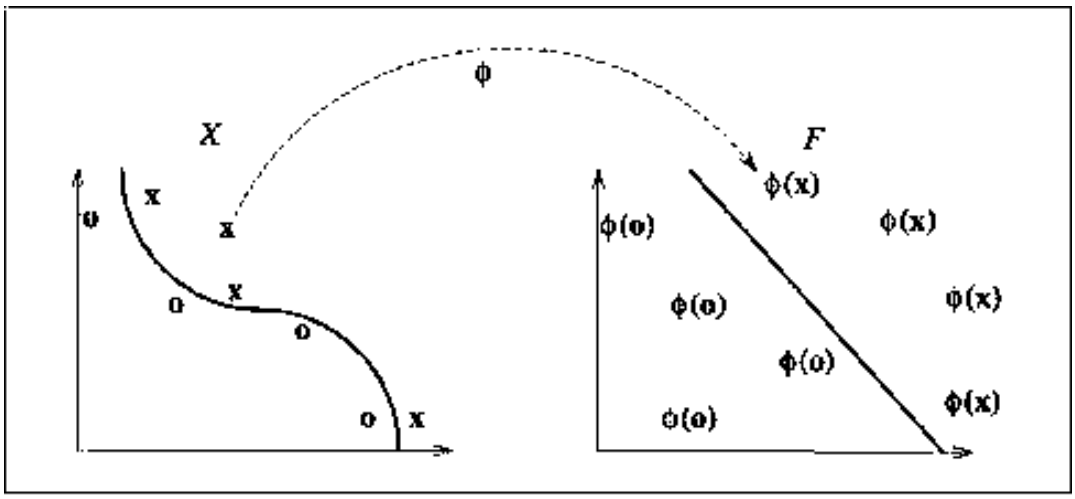

Figura 2.3: Traspaso de rasgos en el que se simplifica la tarea de clasificación [2].

2.2.3. Algoritmo de clasificadores: K-nn

La técnica de los k vecinos más cercanos o también conocida como algoritmo de aprendizaje basado en instancias, tiene un funcionamiento muy simple: se almacenan los ejemplos de entrenamiento de datos históricos y cuando se requiere clasificar a un nuevo objeto, se extraen los k objetos más parecidos y se usa su clasificación para clasificar al nuevo objeto. Los vecinos más cercanos a una instancia se obtienen, para el caso de los atributos continuos, utilizando la distancia Euclidiana sobre los n posibles atributos. El resultado de la clasificación por medio de este algoritmo puede ser discreto o continuo. En el caso discreto, el resultado de la clasificación es la clase más común de los k vecinos [50, 51].

Los ejemplos de entrenamiento son vectores en un espacio característico multi-dimensional, cada ejemplo está descrito en términos de p atributos considerando q clases para la clasificación. Los valores de los atributos del *i-ésimo* ejemplo (*donde* $1 \leq i \leq n$) se representan por el vector *p-dimensional*.

$$x_i = (x_{1i}, x_{2i}, ..., x_{pi}) \in X \tag{2.12}$$

El espacio es particionado en regiones por localizaciones y etiquetas de los ejemplos de entrenamiento [3].

- **Algoritmo de entrenamiento**

  Consiste en que cada tupla de ejemplo $\langle x, f(x)\rangle$, donde $x \in X$, se debe agregar a la estructura de conocimiento que actualmente están representando los ejemplos de aprendizaje. $f(x)$ se refiere a la clase que pertenece o ha sido etiquetado el elemento x.

- **Algoritmo de clasificación**

  Para la clasificación de nuevos objetos, se tiene que dado un ejemplar desconocido x_q y el cual debe ser clasificado, se obtienen los ejemplos $x_1, ..., x_k$, es decir los k vecinos más cercanos a x_q de acuerdo a una medida de distancia a partir de los ejemplos de aprendizaje; se debe regresar $\hat{f}(x)$, que corresponde a la clase que la mayoría de ejemplos x_k pertenecen, esto se específica por la ecuación (2.13).

$$\hat{f}(x) \leftarrow argmax_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i)) \tag{2.13}$$

  donde,

$$\delta(a, b) = 1 \text{ si } a = b; \text{ y } 0 \text{ en cualquier otro caso.} \tag{2.14}$$

  el valor $\hat{f}(x_q)$ devuelto por el algoritmo como un estimador de $f(x_q)$ es solo el valor más común de f entre los k vecinos más cercanos a x_q. Si elegimos $k = 1$; entonces el vecino más cercano a x_i determina su valor.

Elección del k

La mejor elección de k depende fundamentalmente de los datos; generalmente, valores grandes de k reducen el efecto de ruido en la clasificación, pero crean límites entre clases parecidas. Un buen k puede ser seleccionado mediante una optimización de uso. El caso especial en que la clase es predicha para ser la clase más cercana al ejemplo de entrenamiento (cuando $k = 1$) es llamada *Nearest Neighbor Algorithm*, Algoritmo del vecino más cercano. Se sugiere elegir valores de k impar, para evitar empates en la votación de los k vecinos más cercanos.

Posible variante del algoritmo básico

- **Vecinos más cercanos con distancia ponderada**

  Se puede ponderar la contribución de cada vecino de acuerdo a la distancia entre él y el ejemplar a ser clasificado x_q, dando mayor peso a los vecinos más cercanos. Por ejemplo podemos ponderar el voto de cada vecino de acuerdo al cuadrado inverso de sus distancias, ver las ecuaciones (2.15) y (2.16).

$$\hat{f}(x_q) \leftarrow argmax_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i)) \tag{2.15}$$

  donde

$$w_i \equiv \frac{1}{d(x_q, x_i)^2} \tag{2.16}$$

  De esta manera se ve que no hay riesgo de permitir a todos los ejemplos de entrenamiento contribuir a la clasificación de x_q, ya que al ser muy distantes no tendrían peso asociado. La desventaja de considerar todos los ejemplos seria su lenta respuesta (método global). Se quiere siempre tener un método local en el que solo los vecinos más cercanos son considerados.

  Esta mejora es muy efectiva en muchos problemas prácticos. Es robusto ante los ruidos de datos y suficientemente efectivo en conjuntos de datos grandes. Se puede ver que al tomar promedios ponderados de los k vecinos más cercanos el algoritmo puede evitar el impacto de ejemplos con ruido aislados.

- **Ejemplo del algoritmo K-nn**

  En la figura 2.4 se muestra el ejemplo u objeto que se desea clasificar, la figura geométrica rellena círculo. Para $k = 3$ este es clasificado con la clase triángulo, ya que hay solo un cuadrado y 2 triángulos, dentro del círculo de línea continua que los contiene. Si $k = 5$ este es clasificado con la clase cuadrado, ya que hay 2 triángulos y 3 cuadrados, dentro del círculo externo que es representado por la línea punteada.

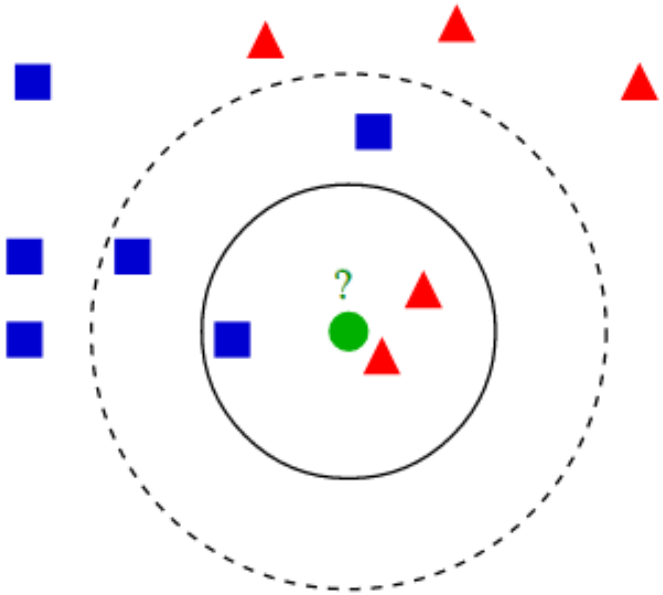

Figura 2.4: Ejmeplo del algoritmo k-nn, para clasificar el círculo verde. Para clasificar con $k = 3$ se consideran los objetos encerrados por el círculo de línea continua y menor diámetro, mientras que para $k = 5$ son considerados todos los objetos dentro de la circunferencia de la línea punteada. Fuente: [3].

Capítulo 3

Construcción del Conjunto de Datos

3.1. Introducción

Durante los últimos años ha habido, tanto en América como en Europa y Japón, un gran crecimiento del interés en la creación y explotación de corpus lingüísticos como parte de la infraestructura para el desarrollo de aplicaciones encaminadas al procesamiento del lenguaje. El tratamiento estadístico de los datos que facilitan los corpus ha demostrado ser eficaz para encontrar la solución a algunos problemas tradicionales de la lingüística computacional, de la traducción automática, etc. El auge que ha tomado esta disciplina ha hecho que actualmente en casi todos los centros de investigaciones lingüísticas se esté trabajando en la confección de algún tipo de corpus.

Hoy en día la informática facilita tanto la organización y la explotación de grandes cantidades de datos que sería impensable crear un corpus prescindiendo de este medio o herramienta [52].

Una de las tareas más desafiantes en la ciencia de la computación es construir máquinas o programas de computadoras que sean capaces de aprender. El darles la capacidad de aprendizaje a las máquinas abre una amplia gama de nuevas aplicaciones. El entender también como éstas pueden aprender nos puede ayudar a comprender las capaci-

dades y limitaciones humanas de aprendizaje.

3.1.1. Conjunto de datos en un sistema de aprendizaje

En un sistema de aprendizaje, a partir de un conjunto de datos se distinguen dos tipos principales: ***el conjunto de entrenamiento y el conjunto de prueba***. Para obtener estos, dividimos los datos muestrales en dos partes; una parte se utiliza como conjunto de entrenamiento para determinar los parámetros del clasificador y la otra parte, llamada **conjunto de prueba** (test ó conjunto de generalización) se utiliza para estimar el error de generalización ya que el objetivo final es que el clasificador consiga un error de generalización pequeño evitando el sobre ajuste (ó sobre-entrenamiento), que consiste en una sobre valoración de la capacidad predictiva de los modelos obtenidos; en esencia, no tiene sentido evaluar la calidad del modelo sobre los datos que han servido para construirlo ya que esta práctica nos lleva a ser demasiado optimistas acerca de su calidad.

El conjunto de entrenamiento suele a su vez dividirse en conjuntos de entrenamiento (propiamente dicho) y conjunto de validación para ajustar el modelo (ver figura 3.1).

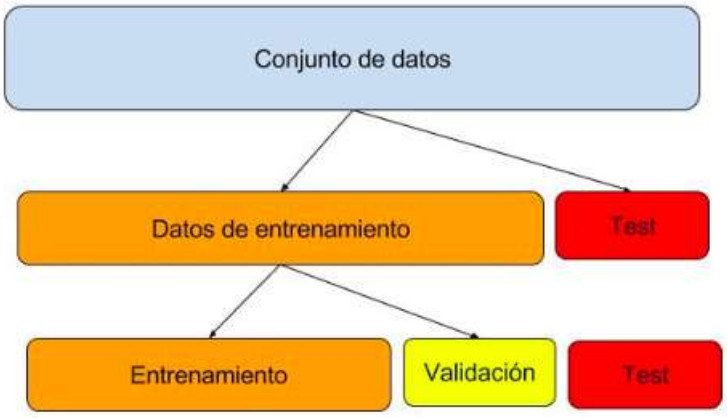

Figura 3.1: Particionamiento típico del conjunto de datos [4].

3.1.2. Conjunto de datos Twitter

El análisis de la polaridad en micro-blogging es una tarea muy reciente, por lo que, incluso en inglés, el número de recursos es muy reducido. Dado que en español existen muy pocos corpus de tweets, se hace necesario crear uno para poder llevar a cabo la parte experimental o de implementación de un sistema a las necesidades que el proyecto requiere. El proceso de descarga de los tweets o fragmentos de textos para el conjunto de datos, se facilita, gracias a que Twitter ofrece varias API's para ello, en este trabajo se utilizo el API llamada ***Tweepy***.

La principal característica de Twitter es que la longitud de los mensajes está limitada a 140 caracteres, por lo que los usuarios de esta red social tienen que expresar sus opiniones, pensamientos y estados de ánimo con muy pocas palabras, siendo los emoticonos y abreviaturas un elemento común de muchos de sus mensajes.

El API Tweepy permitió obtener una cantidad gran cantidad de tweets de diferentes temas y en español, para así implementar un Corpus (conjunto de datos), el cual se divide en dos clases: ***positivos*** y ***negativos***, de ambos es posible obtener un diccionario de palabras individuales y formar así un léxico para cada clase, o también formar un diccionario común, que es el caso en como el modelo vectorial representa los documentos para las técnicas de aprendizaje (ver figura 3.2).

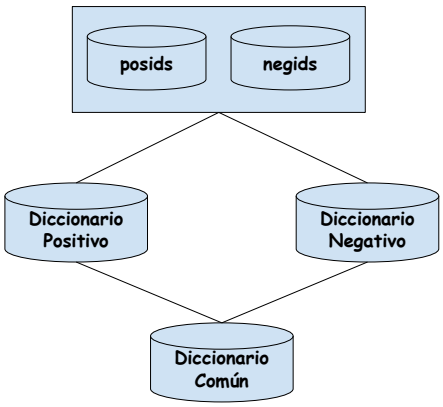

Figura 3.2: Del conjunto de tweets obtenidos, se deben etiquetar en dos clases: positivos y negativos. Se obtiene un diccionario común para representar vectorialmente los documentos o en forma individual, dependiendo del sistema de clasificación.

Los tweets descargados con el API Tweepy se analizaron y se les asignó una etiqueta de polaridad de acuerdo al contenido de cada uno de ellos, esto se realizó con la ayuda del ***API de análisis de sentimientos: MeaningCloud*** para así llevar a cabo de una forma más automática esta tarea y probarlo con nuestro sistema. De otra forma, se debe recurrir a un grupo de personas que etiqueten los textos y por votación asignarles la etiqueta de polaridad, lo cual consume demasiados recursos y tiempo.

3.2. API's para la Obtención del Corpus

En la actualidad, la mayoría de sistemas en la web ofrece a los desarrolladores API's para poder interactuar con los datos de los sitios web's, de tal forma que se pueda interactuar de una forma transparente para los usuarios de las aplicaciones y es tarea de los programadores crear los elementos de programación que permiten obtener información de dichas páginas, es decir, poder trabajar y explotar los datos existentes en ellas para sus propios intereses.

En esta sección se presentan diferentes API's para obtener datos de diferentes aplicaciones en la WEB, aún cuando el trabajo se basa en la red social de twitter, es necesario dar a conocer la existencia de otras API's para la construcción de corpus en otras temáticas diferentes.

3.2.1. API de Twitter: Tweepy

Python es un gran lenguaje para todo tipo de cosas. Existe una comunidad activa de desarrolladores que crea n bibliotecas que extienden el lenguaje y hacen más fácil usar servicios. Una de esas bibliotecas es tweepy. Tweepy es de código abierto, alojado en GitHub (plataforma de desarrollo colaborativo) y permite a Python comunicarse con la plataforma de Twitter y el uso de su API.

La versión actual de tweepy es la 1.13. Fue lanzado el 17 de enero del 2016 y ofrece varias correcciones de errores y nuevas funcionalidades en comparación con la versión anterior. La versión 2.x se está desarrollando actualmente, pero es inestable por lo que una gran mayoría de los usuarios debe utilizar la versión 1.13.

Instalación de Tweepy

La instalación de tweepy es fácil, existen dos maneras sencillas de hacerlo:

1. Se puede clonar desde el repositorio Github:

```
git clone https://github.com/tweepy/tweepy.git
python setup.py install
```

2. se puede instalar directamente:

```
pip install tweepy
```

De cualquiera de las dos maneras se proporciona la versión más reciente.

Usando Tweepy

Tweepy soporta el acceso a Twitter a través de la autenticación básica y el nuevo método, OAuth (Open Authorization). Twitter ha dejado de aceptar la autenticación básica por lo que el método OAuth es ahora la única manera de utilizar la API de Twitter.

La imagen 3.3 muestra un ejemplo de cómo acceder a la API de Twitter usando tweepy con OAuth:

```
1 import tweepy
2
3 # Consumer keys and access tokens, used for OAuth
4 consumer_key = 'lZ1bVA0zxI7s4tPtrANNazrTQ'
5 consumer_secret = 'ZCfs4dFNm9n2HD9UW6QVGGS8QGX2psyqoR5B3LkqfeTxObVZQb'
6 access_token = '2920220073-PWo1XoQUSNAqYJKo4QMJd6ftY1zkcRr2ppVTu8U'
7 access_token_secret = 'HAzqrW9clXdojqJe9CdSNNB5ek3kVYtfbIFAf8WaQBOlA'
8
9 # OAuth process, using the keys and tokens
10 auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
11 auth.set_access_token(access_token, access_token_secret)
12
13 # Creation of the actual interface, using authentication
14 api = tweepy.API(auth)
15
16 api.update_status('Hola Python desde mi Terminal ;)')
```

Figura 3.3: Ejemplo de cómo acceder a la API de Twitter usando tweepy con OAuth.

El resultado del código de la imagen 3.3 se muestra en la imagen 3.4:

Figura 3.4: Resultado de la autenticación (se twiteo desde la consola).

La principal diferencia entre la autenticación básica y la autenticación OAuth son los consumer y access keys. Con la autenticación básica, es posible proporcionar un nombre de usuario y contraseña y acceder a la API, pero desde 2010, cuando Twitter comenzó a requerir OAuth, el proceso es un poco más complicado. Una aplicación tiene que ser creada en `dev.twitter.com`.

OAuth es un poco más complicado a diferencia de la autenticación básica, ya que requiere más esfuerzo, pero los beneficios que ofrece son muy atractivos:

- Los tweets pueden ser personalizados para tener una cadena que identifica la aplicación que se utilizó.
- No revela la contraseña del usuario, por lo que es más seguro.
- Es más fácil de gestionar los permisos, por ejemplo, un conjunto de símbolos y claves se pueden generar para que sólo se permita la lectura de las líneas de tiempo, por lo que en caso de que alguien obtenga esas credenciales, él/ella no será capaz de escribir o enviar mensajes directos, minimizando el riesgo.

- La aplicación no responde de una contraseña, por lo que incluso si el usuario lo cambia, la aplicación seguirá funcionando.

Tras iniciar sesión en el portal, e ir a "Aplicaciones", una nueva aplicación se puede crear la cual proporcionará los datos necesarios para la comunicación con la API de Twitter.

Conclusiones de la sección

Para resumir, tweepy es una biblioteca de código abierto que proporciona acceso a la API de Twitter para Python. Aunque la documentación de tweepy es escasa y no tiene muchos ejemplos, el hecho de que en gran medida se basa en la API de Twitter, que tiene una excelente documentación, hace que sea probablemente la mejor biblioteca de Twitter para Python, especialmente cuando se considera la Transmisión de soporte de la API, que es donde sobresale tweepy. Otras bibliotecas como python-twitter proporciona muchas funciones también, pero tweepy tiene la comunidad más activa y la que más se compromete con el código en el último año.

3.2.2. API de Wikipedia

En ocasiones es necesario crear corpus que contengan diversos tópicos o clases, una fuente de datos muy importante para los investigadores en las áreas de NLP y Minería de Datos es Wikipedia. Wikipedia tienen una API que es una biblioteca de Python, la cual hace que sea fácil de acceder y analizar datos de esta fuente creciente de información. La acción de buscar en Wikipedia, obtener resúmenes de artículos, obtener datos como enlaces y las imágenes de una página, y mucho más son algunas de estas acciones.

Exploración de API de Wikipedia a través de Python

En el artículo "Exploring the Wikipedia JSON API"se deduce directamente el objetivo del API, en lugar de ver el API de Wikipedia a través del navegador web (es decir, a través de medios gráficos humanos-ambiente), se describe cómo hacer lo mismo en el código Python. Si, "haciendo lo mismo, pero en Python (o en cualquier lenguaje de programación)". Por lo que el artículo sirve además de guía para ver la forma de aprovechar

los bucles y funciones para hacer eficiente la minería de datos escalable, que generalmente es el objetivo principal de la programación.

Instalación

Para instalar el API de Wikipedia se utiliza el siguiente comando directamente en la terminal:

```
sudo pip install wikipedia
```

Funciones y clases

Algunas de las funciones y clases más importantes se describen en esta subsección.

- ***wikipedia.set_lang (prefix):*** Cambiar el idioma de la API que se solicita.
- ***wikipedia.summary(query, sentences=0, chars=0, auto_suggest=True, redirect=True):***
  1. ***sentences:*** Si se activa, devuelve las primeras frases frases (puede ser superior a 10).
  2. ***chars:*** si se activa, devuelve sólo los primeros caracteres chars (texto real devuelve puede ser ligeramente más largo).
  3. ***auto_suggest*** Wikipedia dejó encontrar un título de página válido para la consulta
  4. ***redirect:*** permitir redirección sin levantar RedirectError.
- ***wikipedia.search(query, results=10, suggestion=False):*** Realizar la busqueda en Wikipedia

  Los argumentos:

  1. ***results:*** El máximo numero resultados obtenidos.
  2. ***suggestion:*** Si es verdadero, resultados y sugerencias (si lo hay) de regreso en una tupla.

- ***wikipedia.suggest(query):*** Obtener una una sugerencia de wikipedia con query, si se encuentra regresa una cadena de lo contrario no regresa nada.

- ***wikipedia.page(title=None, pageid=None, auto_suggest=True, redirect=True, preload=False):*** Obtener una una página completa a través del titulo.

  1. ***title:*** El título de la página se cargue.
  2. ***pageid:*** El numero de página para cargar.
  3. ***auto_suggest:*** Wikipedia encontró un título de página válido para la consulta.
  4. ***redirect:*** Permitir redirección sin levantar.
  5. ***preload:*** Contenido de la carga, Resumen, imágenes, referencias y enlaces durante la inicializacion.

Uso del api de Wikipedia

Para realizar una pequeña simulación, se ha utilizado el código en python como se muestra en la figura 3.5

```
3  import wikipedia
4  import sys
5
6  #Especificar el idioma
7  wikipedia.set_lang("es")
8
9  # Obtener la definicion de una palabra
10 print wikipedia.summary(sys.argv[1], sentences=2)
11
12 # Realizar una busqueda en wikipedia
13 print wikipedia.search(sys.argv[1])
14
15 # Obtener una una sugerencia de wikipedia
16 print wikipedia.suggest(sys.argv[1])
17
18 # Obtener una una pagina completa a traves del titulo
19 pagina = wikipedia.page(sys.argv[1])
20
21 print pagina.title
22 print pagina.content
23 print pagina.url
24 print pagina.links
```

Figura 3.5: Clases y funciones que se utilizaron para la ejemplificación.

Para ejecutarlo se utiliza la siguiente instrucción en la línea de comandos:

- «python wikipedia1.py expresion»

Le pasamos el parametro la palabra ("expresion"), lo que hace el programa realiza la búsqueda en wikipedia, en la figura 3.6 muestra los resultados de la búsqueda; el título de la página, el contenido de la página, la dirección de la página (url), por último los links de la página que se pueda encontrar relacionada la palabra ("expresion").

```
josejuan@Notebook:~/Descargas$ python wikipedia1.py expresion
/usr/local/lib/python2.7/dist-packages/requests-2.10.0-py2.7.egg/requests/packages/urllib3/util/ssl_.py:318: SNIMissingWarning: An HTTPS request has been made, but the SNI (Subject Name
 Indication) extension to TLS is not available on this platform. This may cause the server to present an incorrect TLS certificate, which can cause validation failures. You can upgrade
to a newer version of Python to solve this. For more information, see https://urllib3.readthedocs.org/en/latest/security.html#snimissingwarning.
  SNIMissingWarning
/usr/local/lib/python2.7/dist-packages/requests-2.10.0-py2.7.egg/requests/packages/urllib3/util/ssl_.py:122: InsecurePlatformWarning: A true SSLContext object is not available. This pre
vents urllib3 from configuring SSL appropriately and may cause certain SSL connections to fail. You can upgrade to a newer version of Python to solve this. For more information, see htt
ps://urllib3.readthedocs.org/en/latest/security.html#insecureplatformwarning.
  InsecurePlatformWarning
/usr/local/lib/python2.7/dist-packages/requests-2.10.0-py2.7.egg/requests/packages/urllib3/util/ssl_.py:122: InsecurePlatformWarning: A true SSLContext object is not available. This pre
vents urllib3 from configuring SSL appropriately and may cause certain SSL connections to fail. You can upgrade to a newer version of Python to solve this. For more information, see htt
ps://urllib3.readthedocs.org/en/latest/security.html#insecureplatformwarning.
  InsecurePlatformWarning
/usr/local/lib/python2.7/dist-packages/requests-2.10.0-py2.7.egg/requests/packages/urllib3/util/ssl_.py:122: InsecurePlatformWarning: A true SSLContext object is not available. This pre
vents urllib3 from configuring SSL appropriately and may cause certain SSL connections to fail. You can upgrade to a newer version of Python to solve this. For more information, see htt
ps://urllib3.readthedocs.org/en/latest/security.html#insecureplatformwarning.
  InsecurePlatformWarning
Traceback (most recent call last):
  File "wikipedia1.py", line 10, in <module>
    print wikipedia.summary(sys.argv[1], sentences=2)
  File "/usr/local/lib/python2.7/dist-packages/wikipedia/util.py", line 28, in __call__
    ret = self._cache[key] = self.fn(*args, **kwargs)
  File "/usr/local/lib/python2.7/dist-packages/wikipedia/wikipedia.py", line 231, in summary
    page_info = page(title, auto_suggest=auto_suggest, redirect=redirect)
  File "/usr/local/lib/python2.7/dist-packages/wikipedia/wikipedia.py", line 276, in page
    return WikipediaPage(title, redirect=redirect, preload=preload)
  File "/usr/local/lib/python2.7/dist-packages/wikipedia/wikipedia.py", line 299, in __init__
    self.__load(redirect=redirect, preload=preload)
  File "/usr/local/lib/python2.7/dist-packages/wikipedia/wikipedia.py", line 393, in __load
    raise DisambiguationError(getattr(self, 'title', page['title']), may_refer_to)
wikipedia.exceptions.DisambiguationError: "Expresión" may refer to:
Expresión genética
Expresión matemática
Expresión (informática)
Expresión regular
Expresión corporal
Expresión facial
Expresión sonora
Expresión oral
Antonomasia
Expresión (teatro)
Wikcionario
```

Figura 3.6: Clases y funciones que se utilizaron.

3.3. Análisis de Sentimientos con *meaning cloud*

MeaningCloud LLC es una empresa con base en New York, EE.UU. especializada en software de análisis semántico, que acumula una historia de casi 20 años de experiencia en estas tecnologías. La visión de esta empresa es hacer que el análisis de texto de alta calidad resulte accesible para todo tipo de empresas. El producto **MeaningCloud** es uno de los líderes en el sector de análisis de texto en la nube.

La API de Análisis de Sentimiento ***meaning cloud*** realiza un análisis de sentimiento multilingüe detallado a partir de información proveniente de diversas fuentes.

El texto proporcionado se analiza para determinar si expresa un sentimiento positivo, neutro o negativo (o si es imposible detectar algún sentimiento). Para ello, se identifica la polaridad local de las diferentes frases en el texto y se evalúa la relación entre ellas, lo que resulta en un valor de polaridad global para el texto en su conjunto.

Además de la polaridad global y a nivel de frase, la API usa técnicas avanzadas de procesamiento del lenguaje natural para detectar la polaridad asociada tanto a las entidades como a los conceptos del texto. Además permite al usuario detectar la polaridad de entidades y conceptos que él mismo defina, lo que convierte al servicio en una herramienta aplicable a cualquier tipo de escenario.

La primera pregunta cuando se habla de análisis automático de sentimiento es "¿Qué precisión se obtiene?". En realidad, discutir sobre si "por debajo del XX % de precisión la solución es inaceptable" no es una buena idea: la precisión y la cobertura no son independientes y habitualmente es necesario llegar a un compromiso entre una y otra. Y lo que en cada caso constituyen unas prestaciones aceptables depende del problema de negocio. Por ejemplo, en una aplicación de antiterrorismo el objetivo sería una cobertura del 100 %, admitiéndose una baja precisión y falsos positivos (que serían filtrados por revisores humanos). Por el contrario, para otras aplicaciones (p. ej.: percepción de marca en medios sociales) podría resultar aceptable perder casos -baja cobertura- a cambio de obtener una alta precisión.

Sin embargo, factores como volumen y latencia son tanto o más importantes que los anteriores. Si un equipo humano puede analizar cientos de mensajes al 85 % de precisión y un ordenador puede procesar millones al 75 % y en tiempo real las máquinas son sin duda una opción válida.

3.3.1. Clasificación de texto en Excel

La clasificación de texto realiza un análisis que se integra en la funcionalidad proporcionada por la API de clasificación de texto, es decir, que permite asignar una o varias categorías a cualquier texto de acuerdo con el modelo seleccionado. El modelo utilizado para clasificar el texto de entrada puede ser uno de los modelos incluidos en la API o uno de los modelos definidos por el usuario.

Para todos los idiomas hay un modelo por defecto que sigue el Consejo Internacional de Telecomunicaciones de Prensa estándar (IPTC) para las noticias, que clasifica los textos en alrededor de 1.400 categorías.

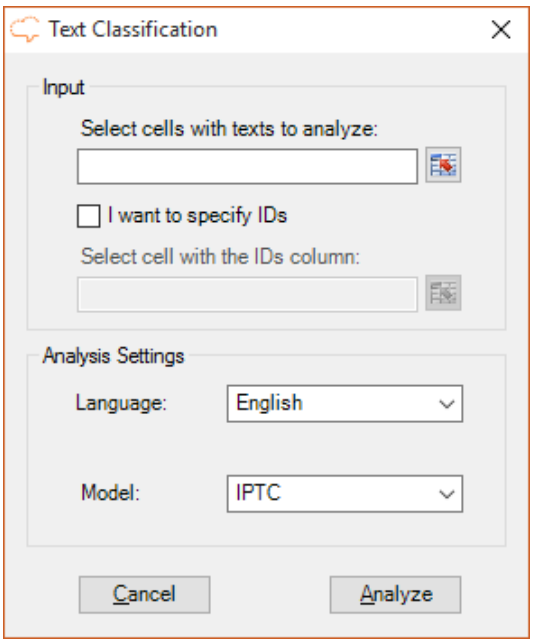

Figura 3.7: Esta es la interfaz que aparece cuando se hace clic en el botón de clasificación de textos.

Como se puede ver en la figura 3.7 hay dos secciones en la interfaz: **Entrada de Datos** y **Configuración de análisis**.

Entrada de Datos: Para todos los análisis disponibles, lo primero que se tiene que definir son los datos que se van a analizar. Para este fin, todas las interfaces asociadas a los diferentes análisis tienen una sección llamada de entrada. Su estructura será siempre el mismo: habrá un cuadro de texto para introducir los datos obligatorios para el análisis que se refiere, en general, los textos, y un cuadro de texto adicional para agregar los identificadores asociados a dichos textos.

La selección de los textos se llevará a cabo mediante la selección de un rango de celdas en la hoja de cálculo de Excel. Para ello, existen dos posibilidades:

- Escribe el rango directamente en el cuadro de texto correspondiente con el formato adecuado.

- Haga clic en el botón a la derecha del cuadro de texto; esto mostrará un cuadro de diálogo que le permitirá seleccionar el rango directamente desde la hoja de cálculo de Excel en el que está trabajando.

Hay algunos límites a los rangos que se pueden seleccionar; los datos obligatorios (cuadro de texto incluido en la sección de entrada) se limitan a una columna y un máximo de 10000 células.

Después de la introducción de los datos obligatorios, es posible especificar identificadores asociados a los textos que van a ser analizados para poder identificar unívocamente cada uno en su sistema. Esto es particularmente importante ya que el resultado se mostrará en otra hoja.

El cuadro de texto que aparece desactivado por defecto en la figura 3.7, permite introducir los identificadores, su uso es opcional. Para activarlo, hay una casilla de verificación con la etiqueta *I want to specify IDs*. Para especificar los ID, seleccione una de las celdas incluidas en la columna que contiene ellas: el complemento automáticamente tomará el valor correspondiente para cada fila incluido en la selección de texto.

En **Configuración de análisis** hay dos valores para seleccionar, los cuales se muestran en la figura 3.8:

- **Lenguaje**, para seleccionar el idioma de los textos. De forma predeterminada, se preselecciona el idioma establecido como preferidos en la configuración general; en caso de que no se ha establecido, el primer elemento de la lista será la seleccionada.
- **Modelo**, con el modelo que se utiliza para clasificar los textos. Los modelos que aparecen son determinados por el idioma seleccionado en el menú anterior. En esta lista se incluyen también los modelos definidos por el usuario asociados a la clave que está siendo utilizado: serán identificados con un icono de bloqueo antes del nombre, como se muestra en la figura 3.8. manera:

Ajustes Avanzados

Hay un menú de configuración avanzada con opciones de configuración adicionales para el análisis de clasificación de textos. Estas son las opciones disponibles y su valor por

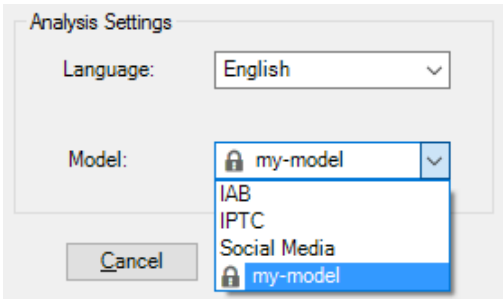

Figura 3.8: Configuración de análisis para seleccionar el Idioma y el Modelo para clasificar los textos.

defecto, como se muestra en la figura 3.9.

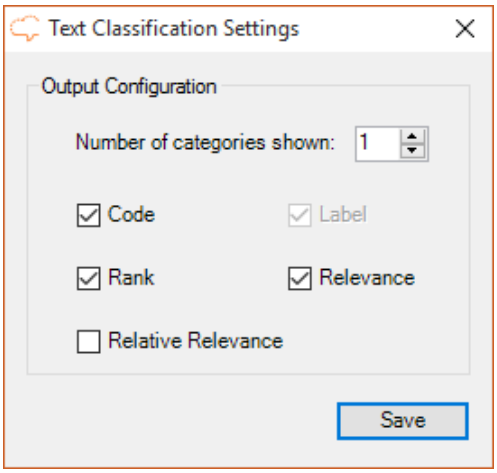

Figura 3.9: Ajustes de clasificación de texto.

Hay dos aspectos principales: para configurar el número de categorías que desee ver en los resultados, y los campos que desee emitir para cada categoría.

- **Número de categorías que se muestran:** establecido por defecto a 1 y con un valor máximo de 10.
- **Campos:**
  - **Código** : muestra el código asociado a la categoría.
  - **Etiqueta** : muestra la etiqueta de la categoría; es no configurable, por lo que siempre aparece en los resultados.

- **Rango** : muestra el rango u orden en el que una categoría se ha asociado a un texto.
- **Relevancia** : muestra la relevancia absoluto asociado a la categoría.
- **Importancia relativa** : muestra la importancia relativa asociada a la categoría.

Salida

Los resultados obtenidos a partir de la clasificación se muestran en una nueva hoja de Excel llamado "texto de clasificación". Esta hoja incluirá una columna con el texto original, una columna con los ID Si está activado, y luego una columna para cada uno de los campos de salida configurados en la configuración avanzada.

Cuando el análisis se configura a la salida más de una categoría, cada categoría adicional asociado a un texto se inserta como una nueva fila, lo que permite un uso más flexible de los resultados.

Este es un ejemplo de una posible salida de textos en Inglés clasificadas utilizando el modelo de IPTC y sin el uso de identificadores. La configuración está configurado para mostrar los campos de salida configurados por defecto y hasta 3 categorías, ver figura 3.10.

	A	B	C	D	E
1	**Text**	**Code**	**Rank**	**Label**	**Relevance**
2	Mobile phone group O2 has agreed to a takeover by the Spanish telecoms company Telefonica.	04003006	1	economy, business and finance - computing and information technology - telecommunication equipment	0,31982675
3		04016005	2	economy, business and finance - company information - merger, acquisition and takeover	0,21922602
4	Telefonica, the parent company of Big Brother producer Endemol, is paying £17.7bn for O2.	04008034	1	economy, business and finance - macro economics - business enterprises	0,18663651
5	The deal will give Telefonica a foothold in the UK and Germany, two of Europe's largest mobile phone markets. O2, which was spun off	04003006	1	economy, business and finance - computing and information technology - telecommunication equipment	0,24190137
6	BT four years ago, is Europe's sixth-largest mobile phone company.	04003007	2	economy, business and finance - computing and information technology - telecommunication service	0,16431189
7		04003009	3	economy, business and finance - computing and information technology - wireless technology	0,15058629
8	It has been a target for other predators, attracting the interest of Dutch company KPN and Deutsche Telecom, which owns T-Mobile. Some analysts believe there is a good chance of a rival company putting in a counter-bid to scupper Telefonica's move. Telefonica	04008034	1	economy, business and finance - macro economics - business enterprises	0,15756604
9	said O2 would retain its existing brand and continue to be based in the UK following today's deal.	04003006	2	economy, business and finance - computing and information technology - telecommunication equipment	0,123626016
10	The combination with O2 is a logical step for Telefonica in pursuing its strategic goal of providing its shareholders with both growth and cash returns, the Spanish firm said in a statement.	04016047	1	economy, business and finance - company information - shareholders	0,059287272
11					
12					
13					

Figura 3.10: Resultados obtenidos de clasificación de textos utilizando el modelo IPTC. El valor de Rlevancia (columna E) se refiere a el nivel de pertenencia a una categoría del texto. La columna B, es un identificador de los textos en la columna A. La columna B se refiere a las etiquetas de las categorías establecidas en el modelo seleccionado.

Capítulo 4

Implementación del Clasificador de Tweets en la Web

En este capítulo se proporciona una descripción de la interfaz de un sistema mínimo de clasificación de tweets en la WEB, en donde se han integrado los diferentes clasificadores que fueron previamente entrenados para clasificar texto en español (opinión o polaridad del texto en dos clases: POSITIVA, NEGATIVA).

4.1. Diagrama de flujo de procesamiento de predicción de tweets

La figura 4.1 ilustra el diagrama de flujo que siguen los datos que se utilizan en el Sistema implementado. Primero el texto de entrada (tweet) pasa a la etapa de preprocesado (ver la sección 5.2), por ejemplo, se eliminan los signos de puntuación y se convierten a minúsculas cada uno de los caracteres del texto. Entonces se pasa el texto resultante a la etapa del clasificador (algoritmos descritos en la sección 2.2), el cual previamente fue entrenado y recibe el modelo que contiene lo aprendido en el entrenamiento (verl el apéndice A de como se generan los modelos para los clasificadores). Finalmente, la salida del sistema es el resultado de los clasificadores, la cual consiste en una de las etiquetas siguientes: *pos* que indica la polaridad positiva para el texto que proporcionó el usuario; y la etiqueta *neg*

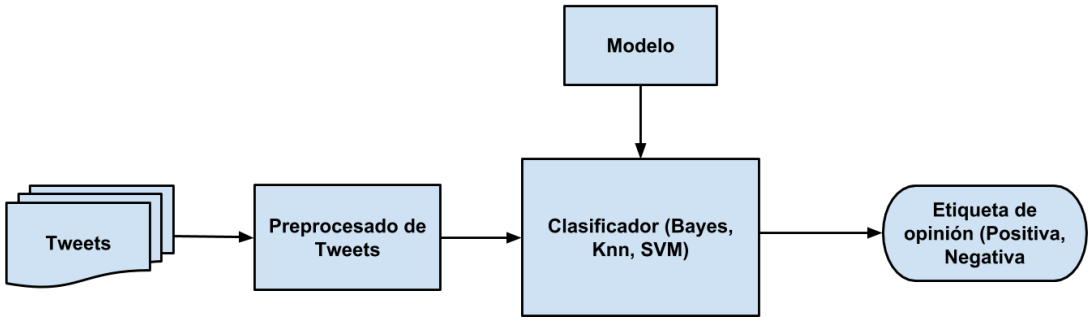

Figura 4.1: Diagrama de flujo del sistema de opinión o predicción de polaridad.

que indica la predicción que corresponde a la polaridad negativa para el texto analizado.

4.2. Diseño de interfaz Web del sistema de opinión.

El sistema de predicción de texto se implementó con la ayuda de 2 distintos lenguajes de programación: PHP y Python. Donde la implementación funcional principal se encuentra escrito en el lenguaje Python, mientras que para la interacción de la página Web con los sistemas de Clasificación se utilizó el lenguaje PHP. Para describir la apariencia o presentación de la página Web se recurrió al lenguaje de etiquetas HTML (HyperText Markup Language) y a las hojas de estilo CSS (por sus siglas en Inglés: Cascading Style Sheets), que es un lenguaje que describe la presentación de los documentos estructurados en hojas de estilo para diferentes métodos de interpretación, es decir, describe como se va a mostrar un documento en pantalla. Los códigos fuente para la interconexión e implementación de la interfaz Web se presentan en el apéndice B.

4.3. Funcionamiento del Sistema

En está sección se hace referencia a la vista que tiene el Sistema Web de predicción de opinión de textos en Español, en la que se describe a detalle cada elemento que contiene el sistema, y la acción que ejecuta, con la finalidad de mantener documentado para el usuario la forma de uso del sistema de opinión.

4.3.1. Vista principal del Sistema de opinión en la Web

En la figura 4.3.1 se observa la vista principal del Sistema de predicción de opinión en la Web. Los elementos con que cuenta son: el escudo oficial de la U.M.S.N.H en la parte superior izquierda, dicho escudo a la vez es un hipervínculo, que te direcciona a la página oficial de la Universidad. Del mismo modo pero en la parte superior derecha se encuentra el escudo de la Facultad de Ingeniería Eléctrica (F.I.E.), que a la vez también es un hipervínculo que direcciona a la página oficial de la facultad. Seguido de esto se encuentran en una etiqueta de texto un breve saludo de bienvenida.

Figura 4.2: Vista principal del Sistema Clasificación de opinión en la Web.

Etiqueta para ingresar el texto a predecir.

La forma de introducir información al sistema, es decir, el texto en español del cual se desea conocer la opinión en la que fue escrito, es por medio del elemento de entrada de texto. En la figura 4.3, se resalta el campo mediante un ovalo de color, en donde se señala que es posible ingresar el texto que se desea predecir. Dicho campo es una caja de texto sencilla de los elementos de formulario de HTML, tipo “text”, en la que se deja al usuario espacio suficiente para poder escribir un texto, ya sea corto o largo.

Una vez ingresado el texto en el área definida, podemos proceder a analizar la polaridad de dicho texto. La figura 4.4 muestra un ejemplo práctico de un usuario que ingresa texto al Sistema Web, el cual se desea analizar con algún(os) clasificador(es) específico(s).

Figura 4.3: Área para ingresar texto a predecir.

Figura 4.4: Ejemplo de entrada de texto al Sistema.

Selección del clasificador para analizar texto.

Una vez ingresado el texto al Sistema se procede a seleccionar el clasificador deseado para analizar la polaridad del texto (positiva o negativa). El Sistema Web cuenta con tres tipos de Clasificadores o algoritmos: NB, K-nn y SVM; los cuales tienen un mismo fin, asignar polaridad al texto ingresado al Sistema. En la figura 4.5 se señala con una marca de color la sección para elegir el Clasificador que se desea utilizar.

Figura 4.5: Selección de Clasificador en el Sistema Web.

Predicción del texto

Una vez que se seleccionó el Clasificador o Clasificadores deseados para analizar el texto ingresado, se procede a darle clic al botón "Analizar", la acción de este elemento del formulario es invocar el script que recibe los datos del usuario y realizar la acción de clasificación. En la figura 4.6 el elemento se encuentra encerrado por una marca de color.

Figura 4.6: Botón para realizar la acción de clasificación del texto ingresado

Posterior a presionar el botón de "Analizar", el sistema agrega sobre la misma vista la predicción de opinión o polaridad que se obtuvo con cada clasificador seleccionado. La figura 4.7 indica con la marca de color la salida del Sistema Web que el usuario obtendrá

para el texto que fue introducido.

Figura 4.7: Vista final del Sistema de opinión, indicando la polaridad que los clasificadores proporcionaron al texto de entrada.

Capítulo 5

Experimentos

En este capítulo se presenta una descripción del conjunto de datos que se obtuvieron de la red social Twitter. También se presenta la evaluación de los tres clasificadores de texto que fueron descritos como algoritmos en la sección 2.2, ver la figura 5.1 que nos muestra un diagrama de flujo que indica la fase de entrenamiento y prueba que debe realizarse en cada uno de los clasificadores para obtener su evaluación. Para cada uno de los clasificadores se presenta el comportamiento de desempeño en base a lo siguiente:

1. La cantidad de documentos utilizados en la etapa de entrenamiento.

2. Tipo de preprocesamiento de los datos en los corpus.

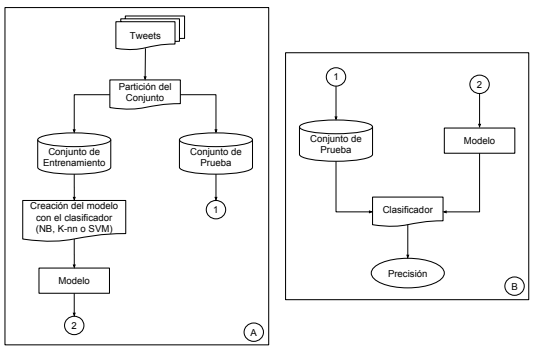

Figura 5.1: Diagrama de flujo del experimento implementado para obtener la evaluación de los clasificadores. A) Entrenamiento del Clasificador (aprendizaje). B) Prueba del Clasificador (evualuación del aprendizaje)

5.1. Características de los datos utilizados

Para la experimentación de los tres diferentes clasificadores que se implementaron se utilizaron conjuntos de Tweets de diferentes temas comentados por diversos usuarios de dicha red social. Los Tweets que se obtuvieron fueron dos conjuntos de datos y cada uno de ellos fueron divididos y clasificados en dos subconjuntos: Tweets Positivos y Tweets Negativos, ver la tabla 5.1 que señala la cantidad de Tweets en cada subconjunto o clase definida.

Tabla 5.1: Corpus de Tweets, capturados en 2 conjuntos de datos y etiquetados en dos clases.

Conjuntos	***Tweets***		
	Positivos	**Negativos**	***Total de Tweets***
C1	666	737	*1, 403*
C2	3, 343	2, 686	*6, 028*

La tabla 5.2 contiene la cantidad de palabras en los dos conjuntos de datos recolectados de la red social Twitter y que a su vez fueron etiquetados como: Tweets Positivos y Tweets Negativos.

Tabla 5.2: Cantidad de palabras en los conjuntos de datos utilizados en lo experimentos.

Conjuntos	***Cantidad de palabras***		
	Positivos	**Negativos**	***Tamaño Total***
C1	14, 292	15, 390	*29, 682*
C2	58, 619	47, 400	*106, 019*

En la tabla 5.3 se muestra el tamaño del vocabulario (cantidad de palabras únicas en los datos) en los subconjuntos de datos, este tamaño representa la dimensión vectorial o tamaño de características que han de utilizar los algoritmos de clasificación.

Tabla 5.3: Tamaño del vocabulario en los subconjuntos de datos.

Conjuntos	***Tamaño del Vocabulario***		
	Positivos	**Negativos**	***Tamaño Total***
C1	3, 185	2, 792	*5, 977*
C2	10, 197	7, 101	*17, 298*

5.2. Preprocesamiento de los datos

Uno de los fines de la evaluación, es identificar la efectividad de los clasificadores ante diferentes tipos de preprocesamiento de los datos, para con ello tomar la mejor decisión en la implementación dentro de un sistema o aplicación comercial. Los tipos de preprocesamiento que se plantean y a los cuales se sometieron los clasificadores son enumerados de la siguiente forma:

Preprocesamiento 1. **Tweets Originales.** Se refiere a no realizar ninguna modificación a los datos. En estos datos las palabras *bueno* y *Bueno* son dos características diferentes en el análisis.

Preprocesamiento 2. **Tweets Originales sin Signos de Puntuación.** En este caso el preprocesamiento de los datos se refiere a eliminar todos los signos de puntuación de los datos originales, tales como: *: ; , . ? ¿ ! ¡*. Con este procesamiento de los datos las palabras *bien!!!!!* y *bien?* son una misma característica en la representación vectorial.

En la tabla 5.4 se muestra el nuevo tamaño del vocabulario una vez que se ha aplicado la remoción de los signos de puntuación a los textos originales.

Tabla 5.4: Tamaño del vocabulario en el tipo de preprocesamiento 2.

Conjuntos	***Tamaño del Vocabulario en Tweets***		
	Positivos	**Negativos**	***Tamaño Total***
C1	3, 174	2, 789	*5, 963*
C2	9, 662	6, 661	*16, 323*

Preprocesamiento 3. **Tweets con Texto en Minúsculas.** Cada carácter en mayúsculas es convertido a minúsculas. Ejemplos de palabras que se convierten en la misma característica son: *MALDAD* y *maldad*.

En la tabla 5.5 se muestra el tamaño del vocabulario resultante después de que los caracteres en los textos originales se convirtieron a minúsculas.

Tabla 5.5: Tamaño del vocabulario en el tipo de preprocesamiento 3.

Conjuntos	***Tamaño del Vocabulario***		
	Positivos	**Negativos**	***Tamaño Total***
C1	2, 965	2, 610	*5, 575*
C2	9, 391	6, 588	*15, 979*

Preprocesamiento 4. **Tweets con Texto en Minúsculas sin Signos de Puntuación.** En este caso el preprocesamiento de los datos se refiere a eliminar todos los signos de puntuación de los datos originales, tales como: *: ; , . ? ¿ ! ¡*. Esto se realizo una vez que cada carácter en mayúsculas fue convertido en minúsculas, como se dijo en el punto anterior. Con este procesamiento de los datos las palabras *bien!!!!!* y *bien?* son una misma característica en la representación vectorial.

En la tabla 5.6 se muestra el nuevo tamaño del vocabulario después de que en los textos originales se convirtieran los caracteres a letras minúsculas y se además se removieran los signos de puntuación.

Tabla 5.6: Tamaño del vocabulario en el tipo de preprocesamiento 4.

Conjuntos	***Tamaño del Vocabulario***		
	Positivos	**Negativos**	***Tamaño Total***
C1	2, 954	2, 608	*5, 562*
C2	8,860	6, 149	*15, 009*

5.3. Descripción de la evaluación de los clasificadores

5.3.1. Cantidad de documentos en la etapa de entrenamiento

No siempre se cuenta con la cantidad suficiente de datos para entrenar un sistema de aprendizaje de máquina, es por ello que deben someterse estos sistemas a un proceso que permita observar su comportamiento en base a la cantidad de datos disponibles para el entrenamiento, algunos de los aspectos es observar en donde se encuentra la mejor partición de los datos para lograr el mayor desempeño del sistema, y también conocer la existencia de un sobre-entrenamiento del mismo.

El proceso al cual se somete cada clasificador consiste en tomar parte del total de los datos para entrenar y el resto para probar el sistema. La división de los datos va desde un 10 % hasta un 90 %, de tal forma que se inicia con un 10 % de los datos para entrenar cada uno de los sistemas y el resto (90 %) para evaluar el clasificador, posteriormente se dividen los datos en un porcentaje de 20 %-80 %, en donde el 20 % es para entrenar y el 80 % para probar el sistema, ello se repite hasta tener un 90 % de datos para entrenar y un 10 % para probar el sistema, señalando que es de vital importancia la elección aleatoria de los datos para una mayor certeza de los resultados.

Las tablas 5.7 y 5.8 nos muestran las cantidades en que se dividieron los datos de los corpus utilizados en el presente trabajo.

Tabla 5.7: División de los datos indicando las cantidades de documentos utilizados para el entrenamiento y prueba de los clasificadores (C1).

Corpus Conjunto C1		
División de Datos en %	**# Documentos a Entrenar**	**# Documentos de Prueba**
10 % - 90 %	139	1264
20 % - 80 %	280	1123
30 % - 70 %	420	983
40 % - 60 %	560	843
50 % - 50 %	701	702
60 % - 40 %	841	562
70 % - 30 %	981	422
80 % - 20 %	1121	282
90 % - 10 %	1262	141

Tabla 5.8: División de los datos indicando las cantidades de documentos utilizados para el entrenamiento y prueba de los clasificadores (C2).

Corpus Conjunto C2		
División de Datos en %	**# Documentos a Entrenar**	**# Documentos de Prueba**
10 % - 90 %	602	5426
20 % - 80 %	1205	4823
30 % - 70 %	1807	4221
40 % - 60 %	2411	3617
50 % - 50 %	3013	3015
60 % - 40 %	3616	2412
70 % - 30 %	4219	1809
80 % - 20 %	4822	1206
90 % - 10 %	5424	604

5.4. Experimentos con los diferentes Clasificadores

Los experimentos se realizaron para tres clasificadores de texto, los cuales se implementaron en el lenguaje python y son los siguientes:

- Clasificador Bayesiano Ingenuo
- Clasificador K-vecinos más cercanos
- Máquinas de Soporte Vectorial

Para cada división de datos por clasificador y por conjunto de datos, el experimento se efectuó 10 veces. La métrica de evaluación que se reporta es la precisión, la cual esta dada por la ecuación (5.1).

$$P = \frac{T_a}{T_a + T_e} \tag{5.1}$$

En donde T_a se refiere a los tweets que el clasificador identificó como aciertos y T_e se refiere a los tweets que el sistema clasificó erróneamente.

5.5. Resultados de los Experimentos

En esta sección se presentan los resultados de los diferentes experimentos que se realizaron, lo que permite tener evidencia de la efectividad de los diferentes clasificadores en los conjuntos de datos.

5.5.1. Clasificador Bayesiano Ingenuo (NB)

Preprocesamiento 1: Tweets Originales

La tabla 5.9 nos muestra los resultados de precisión que obtuvo el clasificador NB en las 10 veces que se realizó el experimento y para cada una de las divisiones o particionamientos que se realizaron con los datos para el entrenamiento y prueba del clasificador. Se observa que los mejores resultados se obtuvieron en la división de los datos 90 % - 10 %

Tabla 5.9: Precisión del clasificador NB en los 10 experimentos con los tweets originales (C1).

# Experimento	*Cantidad en % de Documentos para el entrenamiento*								
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
1	0.752	0.800	0.841	0.861	0.888	0.908	0.922	0.935	**0.959**
2	0.725	0.784	0.830	0.857	0.880	0.913	0.924	0.945	0.951
3	0.706	0.786	0.823	0.848	0.890	0.913	0.919	0.936	0.958
4	0.692	0.791	0.824	0.866	0.881	0.907	0.933	0.943	0.955
5	0.735	0.806	0.847	0.864	0.877	0.913	0.920	0.948	0.949
6	0.714	0.781	0.823	0.867	0.885	0.895	0.928	0.946	0.951
7	0.737	0.782	0.794	0.868	0.888	0.906	0.928	0.942	0.954
8	0.741	0.796	0.827	0.863	0.886	0.906	0.930	0.935	0.953
9	0.735	0.779	0.838	0.851	0.874	0.913	0.922	0.938	0.957
10	0.660	0.784	0.831	0.868	0.888	0.904	0.924	0.940	0.953

superando en su mayoría el 95 % de efectividad la clasificación de los tweets. La mayor precisión obtenida en este experimento fue de 0.959 en la escala de 0 a 1.

Considerando el conjunto de datos C2, para el mismo clasificador NB y el mismo tipo de preprocesamiento (tweets originales), los resultados de las 10 corridas en las diferentes divisiones de datos los cuales fueron seleccionados en forma aleatoria, se muestran en la tabla 5.10. El comportamiento observado es que a menor cantidad de datos para el entrenamiento, menor es la precisión obtenida por el clasificador y al ir aumentando los datos para el entrenamiento se mejora la precisión del clasificador. En este experimento el mejor resultado de precisión es de 0.954 en la división de datos 90 % - 10 %, lo que sugiere que es necesario recolectar la mayor cantidad de datos posibles para el sistema.

Al obtener el valor promedio de los datos obtenidos en los 10 experimentos que se realizaron para el clasificador en cada corpus de datos utilizados, se ratifica el comportamiento del clasificador. La tabla 5.11 contiene los datos del promedio de las 10 corridas del experimento con cada corpus del clasificador NB, en donde se observa que el mejor desempeño se encuentra en la división de datos del 90 %.

Preprocesamiento 2: Tweets Originales sin Signos de Puntuación

Al proceder con el segundo preprocesamiento de los datos y someter el clasificador Naive Bayes a 10 ejecuciones por cada particionamiento de los datos, se obtuvieron los

Tabla 5.10: Precisión del clasificador NB en los 10 experimentos con los tweets originales (C2).

# Experimento	***Cantidad en % de Documentos en el Entrenamiento***								
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
1	0.769	0.830	0.856	0.877	0.906	0.917	0.928	0.942	0.954
2	0.761	0.812	0.857	0.882	0.903	0.917	0.925	0.941	0.950
3	0.758	0.821	0.852	0.888	0.911	0.919	0.928	0.939	0.948
4	0.776	0.822	0.861	0.884	0.889	0.918	0.931	0.941	0.952
5	0.767	0.820	0.857	0.885	0.899	0.920	0.932	0.944	0.949
6	0.762	0.846	0.867	0.883	0.904	0.918	0.930	0.938	0.952
7	0.752	0.825	0.855	0.886	0.904	0.916	0.932	0.941	0.947
8	0.768	0.823	0.854	0.883	0.898	0.919	0.930	0.943	0.948
9	0.772	0.830	0.848	0.884	0.903	0.915	0.931	0.941	0.950
10	0.773	0.829	0.850	0.888	0.903	0.920	0.924	0.941	0.947

Tabla 5.11: Promedio de la precisión del clasificador NB, mostrando el comportamiento en ambos corpus de datos y la división de datos para el entrenamiento con los tweets originales.

Efectividad Promedio		
División de datos (%)	**Conjunto C1**	**Conjunto C2**
10	0.720	0.766
20	0.789	0.826
30	0.828	0.856
40	0.861	0.884
50	0.884	0.902
60	0.908	0.918
70	0.925	0.929
80	0.941	0.941
90	***0.954***	***0.950***

resultados promedio de precisión en la clasificación de cada una de los conjuntos, ver la tabla 5.12. Los mejores resultados se observan cuando el 90 % de datos son utilizados para entrenar el clasificador en ambos conjuntos de datos.

Los resultados del clasificador NB son muy similares al aplicar los preprocesamientos de los datos 1 y 2, uno de los motivos a los que se le puede atribuir esta situación es la diferencia de palabras entre los datos resultantes después de aplicar el preprocesamiento 2, se observa que es muy pequeña. Ver las tablas 5.11 y 5.12 en donde se observa que los mejores resultados promedio de precisión del clasificador se encuentran en la división de datos del 90 % en ambas muestras de tweets utilizadas en la experimentación.

Tabla 5.12: Precisión promedio de ambos conjuntos en el clasificador NB con el segundo preprocesamiento de los datos.

Efectividad Promedio		
Porcentaje (%)	**Conjunto C1**	**Conjunto C2**
10	0.715	0.768
20	0.789	0.822
30	0.827	0.863
40	0.858	0.886
50	0.883	0.904
60	0.906	0.918
70	0.925	0.932
80	0.940	0.941
90	***0.954***	***0.950***

Preprocesamiento 3: Tweets con Texto en Minúsculas

Los resultados del clasificador NB con el tercer tipo de preprocesamiento de los datos se muestran en la tabla 5.13. En ambos conjuntos de datos el comportamiento del clasificador en relación a la precisión se va mejorando conforme se agregan más datos al entrenamiento. Los mejores resultados se obtuvieron con la división de los datos del 90 % en ambos conjuntos.

Tabla 5.13: Promedio de la precisión del clasificador NB en 10 ejecuciones con el preprocesamiento 3 en ambos conjuntos de datos.

Efectividad Promedio		
Porcentaje (%)	**Conjunto C1**	**Conjunto C2**
10	0.743	0.780
20	0.802	0.842
30	0.845	0.875
40	0.876	0.904
50	0.902	0.922
60	0.921	0.940
70	0.941	0.954
80	0.955	0.965
90	***0.975***	***0.975***

Preprocesamiento 4: Tweets con Texto en Minúsculas sin Signos de Puntuación

Con este tipo de preprocesamiento de los datos, el clasificador NB tuvo un comportamiento de desempeño creciente respecto a la cantidad de datos que se utilizaron para el entrenamiento. Los resultados de precisión promedio en las 10 ejecuciones por cada particionamiento de los datos se muestran en la tabla 5.14. En ambos conjuntos de datos utilizadas en la experimentación se obtuvo 0.975 de precisión en la división de los datos del 90 %.

Tabla 5.14: Comportamiento promedio del clasificador NB con el preprocesamiento 4 de los datos en ambos conjuntos.

Efectividad Promedio		
Porcentaje (%)	**Conjunto C1**	**Conjunto C2**
10	0.732	0.782
20	0.805	0.840
30	0.843	0.877
40	0.873	0.904
50	0.905	0.927
60	0.924	0.941
70	0.942	0.955
80	0.960	0.965
90	***0.975***	***0.975***

En las figuras 5.2 y 5.3 se observa gráficamente el comportamiento del clasificador NB en relación a cada uno de los tipos de preprocesamiento de los datos que se utilizó en la experimentación. Para el caso del tipo de preprocesamiento 3 y 4 es cuando se tienen los mejores resultados. Tanto en el conjunto de datos 1 y 2, al menos en 5 divisiones de los datos para el entrenamiento el tipo 4 de preprocesamiento es mayor, esto indica que convertir a minúsculas y eliminar los signos de puntuación, a partir de la división del 50 % (de a cuerdo a las gráfica de la figuras 5.2 y 5.3) se pueden lograr los mejores resultados en la clasificación con este clasificador.

Con el preprocesamiento de los datos 3 y 4 se puede lograr una mejora en los resultados de precisión del clasificador de al menos un 2 %, en ambos conjuntos de datos.

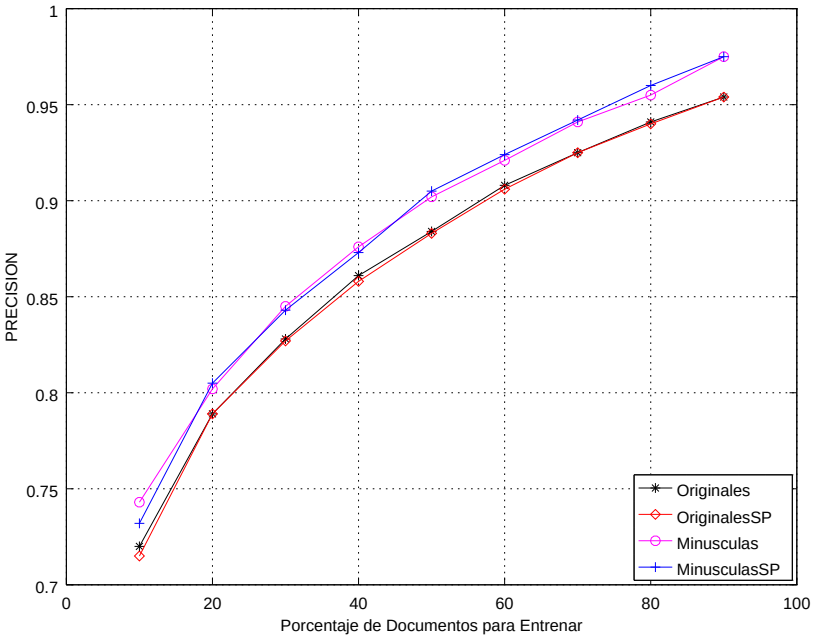

Figura 5.2: Comportamiento de desempeño en el valor de precisión del clasificador NB en el conjunto C1 de tweets, al considerar el tipo de preprocesamiento aplicado a los datos.

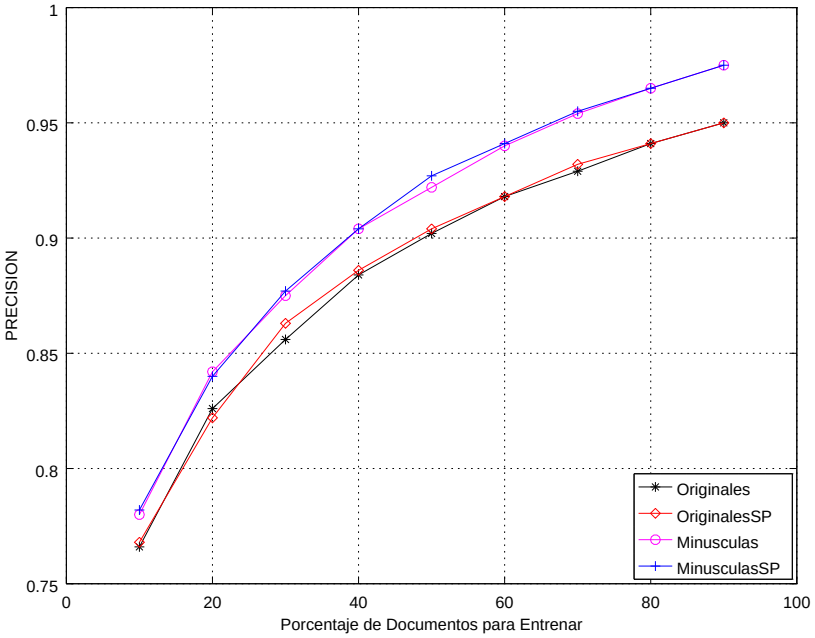

Figura 5.3: Comportamiento del desempeño, en el valor de precisión, para el clasificador NB con el conjunto C2 de tweets, al considerar el tipo de preprocesamiento aplicado a los datos.

5.5.2. Clasificador K-nn

En los experimentos con el clasificador **K-nn**, se utilizó el valor de $K = 3$. Se procedió de forma similar que con el clasificador NB. Los resultados obtenidos al aplicar las diferentes tipos de preprocesamiento de datos a ambos conjuntos de tweets colectados, se presentan en las siguientes subsecciones.

Preprocesamiento 1: Tweets Originales

Primero se presentan los resultados del clasificador con el conjunto de datos C1 en cada una de las ejecuciones con este tipo de preprocesamiento de datos. La tabla 5.15 nos muestra los resultados de precisión que obtuvo el clasificador Knn en las 10 veces que se realizó el experimento y para cada una de las divisiones o particionamiento que se realizó con los datos para el entrenamiento y prueba del clasificador. Se observa que los mejores resultados se obtuvieron en la división de los datos 90 %-10 % superando en su mayoría el 78 % de efectividad la clasificación de los tweets. La mayor precisión obtenida en este experimento fue de 0.858. Sin embargo los resultados de clasificación que se tienen en la columna que representa la división de datos del 60 %, superan en varias ocasiones a los obtenidos en las divisiones del 70 % al 90 %, esto permite señalar que el clasificador se encuentre en un sobreentrenamiento, es decir, se está memorizando los datos en lugar de generalizar el aprendizaje de los datos. Para cuando se le presenten nuevos datos al clasificador (datos de prueba que nunca ha visto) su desempeño es pobre. Una alternativa es el preprocesamiento de los datos o variar los parámetros del clasificador.

Al proceder con el conjunto de datos C2, para el mismo clasificador K-nn y el mismo experimento (tweets originales), los resultados de las 10 corridas en las diferentes divisiones de datos, las cuales fueron seleccionados en forma aleatoria, se muestran en la tabla 5.16. En este experimento el mejor resultado de precisión es de 0.863 en la división de datos 90 % - 10 %, sin observarse el fenómeno de sobreentrenamiento del clasificador en las pruebas para este tipo de preprocesamiento de datos.

Al obtener el valor promedio de los datos obtenidos en los 10 experimentos que se realizaron para el clasificador en cada corpus de datos utilizados, se tienen dos compor-

Tabla 5.15: Precisión del clasificador Knn en los 10 experimentos con los tweets originales (conjunto de datos C1).

# Experimento	***Cantidad en % de Documentos en el Entrenamiento***								
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
1	0.676	0.728	0.729	0.763	0.752	0.786	0.780	0.727	0.801
2	0.688	0.743	0.732	0.758	0.762	0.797	0.780	0.809	0.809
3	0.674	0.715	0.761	0.788	0.761	0.794	0.744	0.826	0.794
4	0.682	0.711	0.737	0.760	0.761	0.767	0.773	0.766	0.723
5	0.635	0.739	0.775	0.743	0.779	0.776	0.751	0.777	0.745
6	0.683	0.726	0.742	0.760	0.775	0.811	0.806	0.816	0.837
7	0.659	0.711	0.721	0.749	0.778	0.786	0.761	0.759	0.723
8	0.681	0.702	0.734	0.746	0.761	0.801	0.810	0.787	0.780
9	0.657	0.693	0.745	0.764	0.746	0.785	0.796	0.791	0.716
10	0.676	0.716	0.712	0.767	0.766	0.763	0.768	0.801	**0.858**

Tabla 5.16: Precisión del clasificador K-nn en los 10 experimentos con los tweets originales (C2).

# Experimento	***Cantidad en % de Documentos en el Entrenamiento***								
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
1	0.719	0.755	0.769	0.782	0.816	0.829	0.823	0.823	0.836
2	0.734	0.759	0.775	0.800	0.818	0.817	0.837	0.834	0.858
3	0.713	0.763	0.780	0.792	0.815	0.821	0.837	0.841	0.826
4	0.716	0.759	0.767	0.802	0.794	0.826	0.833	0.847	**0.863**
5	0.711	0.759	0.755	0.795	0.814	0.808	0.822	0.825	0.829
6	0.715	0.744	0.777	0.796	0.811	0.820	0.815	0.849	0.821
7	0.708	0.750	0.781	0.801	0.814	0.831	0.828	0.828	0.843
8	0.716	0.754	0.771	0.797	0.817	0.825	0.826	0.836	0.826
9	0.702	0.762	0.762	0.799	0.817	0.823	0.831	0.825	0.839
10	0.730	0.761	0.766	0.798	0.814	0.827	0.840	0.844	0.841

tamientos del clasificador, el primero es que para el conjunto C1, los mejores resultados se encuentran en la división de los datos en el 60 %. Como se ha señalado previamente, este es un indicativo de sobreentrenamiento del clasificador, que sugiere aplicar un preprocesamiento de los datos o variar los parámetros del clasificador, en este caso el parámetro k. El segundo comportamiento, que se observa en el conjunto de datos C2, ratifica el desempeño creciente del clasificador conforme se agregan datos al entrenamiento. La tabla 5.17 nos muestra el promedio de las 10 corridas del experimento con cada corpus del clasificador K-nn, en donde se observa que el mejor desempeño se encuentra en la división de datos del 60 % para el conjunto C1 de datos y en la división de datos del 90 % para la conjunto C2

de datos.

Tabla 5.17: Promedio de efectividad del clasificador K-nn, mostrando el comportamiento en ambos corpus de datos y la división de datos para el entrenamiento con los tweets originales.

Precisión Promedio		
Partición (%)	**Conjunto C1**	**Conjunto C2**
10	0.671	0.717
20	0.718	0.757
30	0.739	0.770
40	0.760	0.796
50	0.764	0.813
60	***0.787***	0.823
70	0.777	0.829
80	0.786	0.835
90	0.779	***0.838***

Por otra parte, los resultados obtenidos para el clasificador K-nn son inferiores en más de un 10 % en este mismo experimento que los obtenidos para el clasificador NB.

Preprocesamiento 2: Tweets Originales sin Signos de Puntuación

Al proceder con el segundo preprocesamiento de los datos y someter el clasificador K-nn a 10 ejecuciones con cada particionamiento de los datos, se obtuvieron los resultados promedio de precisión en la clasificación de cada uno de los conjuntos, los resultados se muestran en la tabla 5.18. Los mejores resultados se observan cuando el 80 % de datos son utilizados para entrenar el clasificador en el primer conjunto de datos y el 90 % para el segundo conjunto de datos respectivamente. Lo que nos lleva a señalar que continua con un sobreentrenamiento el clasificador K-nn con el conjunto de datos C1, debido a que en la última partición de datos se ha obtenido un desempeño inferior en la clasificación.

De forma general el comportamiento del clasificador K-nn es muy similar al aplicar los tipos de preprocesamiento de los datos 1 o 2, es decir, en ambos se observa un sobre-entrenamiento en el conjunto de datos C1 y para el conjunto C2 su comportamiento es creciente en relación a la partición de los datos para entrenar y probar. Se observa que se pierde precisión en ambos conjuntos al aplicar este tipo de preprocesamiento de datos, ver las figuras 5.4 y 5.5. El clasificador K-nn requiere que se busque el mejor valor de K para el cual pueda responder a una mayor eficiencia en el conjunto de datos con el que trabaja.

Tabla 5.18: Precisión promedio de ambos conjuntos en el clasificador K-nn con el segundo preprocesamiento de los datos.

Efectividad Promedio		
Partición (%)	**Conjunto C1**	**Conjunto C2**
10	0.672	0.713
20	0.698	0.749
30	0.733	0.778
40	0.748	0.793
50	0.761	0.811
60	0.768	0.818
70	0.775	0.825
80	***0.782***	0.836
90	0.778	***0.843***

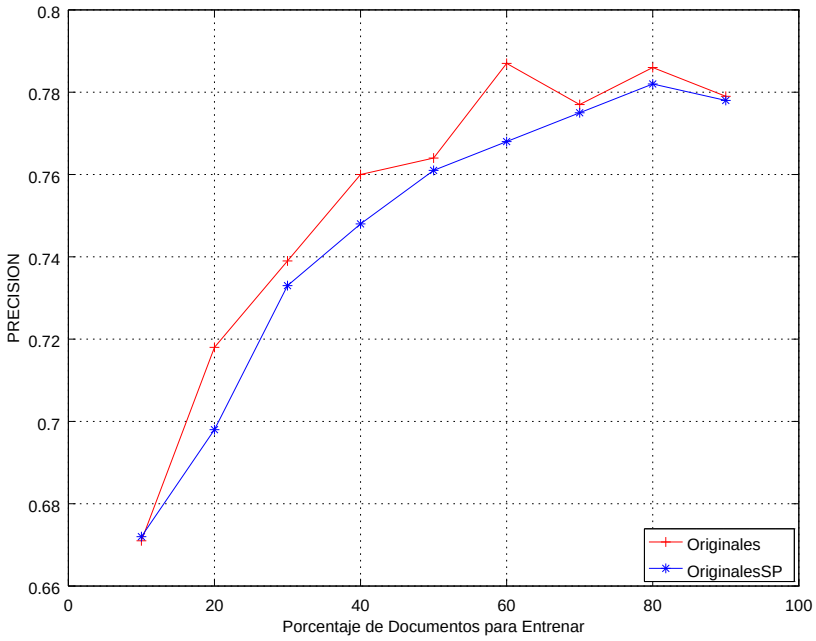

Figura 5.4: Precisión promedio del clasificador k-nn en el conjunto de datos C1 con los dos tipos de preprocesamiento de datos, se observa una disminución en el desempeño.

En las tablas 5.17 y 5.18 se observa que los mejores resultados promedio de precisión del clasificador varían bastante respecto al conjunto C1 ya que en el primer preprocesamiento la mejor precisión se encuentra en la división de datos del 60 % y en el segundo preprocesamiento se encuentra en la división de datos del 80 %. Indicando los dos puntos mencionados anteriormente: 1) sobreentrenamiento del clasificador; y 2) una disminución del desempeño.

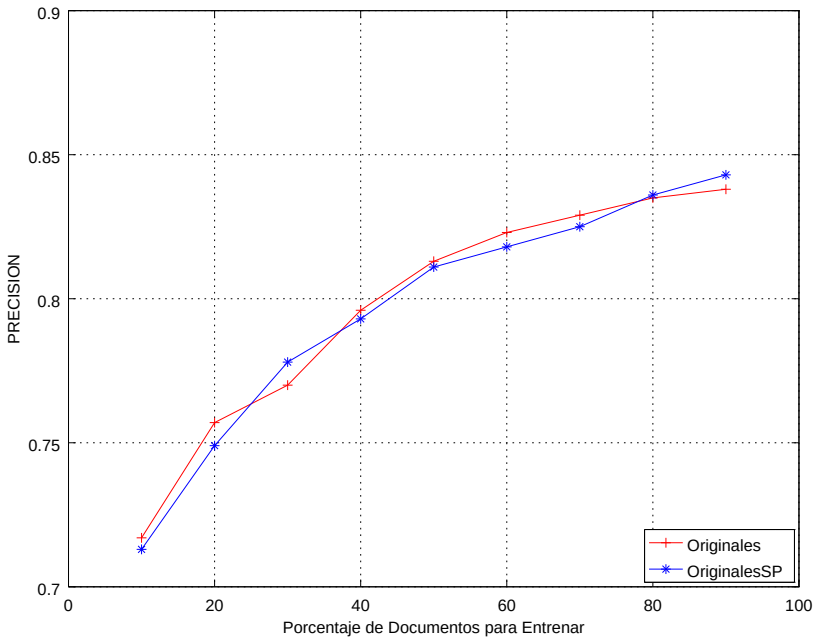

Figura 5.5: Precisión promedio del clasificador k-nn en el conjunto de datos C2, para 2 tipos de preprocesamiento de los datos. Se observa una disminución en la precisión al aplicar el preprocesamiento de datos 2.

Mientras que en el conjunto C2 se observa que los mejores resultados promedio de precisión del clasificador se encuentran en la división de datos del 90 % en ambos preprocesamientos utilizados en la experimentación. También se observó una disminución mínima en la precisión en algunos de los particionamientos de los datos.

Preprocesamiento 3: Tweets con Texto en Minúsculas

Los resultados del clasificador K-nn con el tercer tipo de preprocesamiento de los datos se muestran en la tabla 5.19. En ambos conjuntos de datos el comportamiento del clasificador en relación a la precisión se va mejorando conforme se agregan más datos al entrenamiento. Los mejores resultados se obtuvieron con la división de los datos del 90 % en ambos conjuntos. Sin embargo no se observó una mejora en el desempeño en relación a los dos preprocesamientos previos.

Tabla 5.19: Promedio de la precisión del clasificador K-nn en 10 ejecuciones con el preprocesamiento 3 en ambos conjuntos de datos.

Efectividad Promedio		
Porcentaje (%)	**Conjunto C1**	**Conjunto C2**
10	0.663	0.708
20	0.720	0.753
30	0.748	0.778
40	0.752	0.798
50	0.762	0.811
60	0.787	0.821
70	0.785	0.829
80	0.786	0.833
90	***0.792***	***0.841***

Preprocesamiento 4: Tweets con Texto en Minúsculas sin Signos de Puntuación

Al aplicar el cuarto preprocesamiento a los datos, el clasificador K-nn tuvo un comportamiento de desempeño creciente respecto a la cantidad de datos que se utilizaron para el entrenamiento. Los resultados de precisión promedio en las 10 ejecuciones por cada particionamiento de los datos se muestran en la tabla 5.20. En ambos conjuntos de datos utilizadas en la experimentación se obtuvo la mejor precisión en la división de los datos del 90 %.

Tabla 5.20: Comportamiento promedio del clasificador K-nn con el preprocesamiento 4 de los datos en ambos conjuntos.

Efectividad Promedio		
Porcentaje (%)	**Conjunto C1**	**Conjunto C2**
10	0.650	0.703
20	0.711	0.753
30	0.734	0.772
40	0.753	0.790
50	0.769	0.808
60	0.767	0.820
70	0.785	0.830
80	0.782	0.835
90	***0.785***	***0.835***

La figura 5.6 nos muestra en forma resumida el desempeño promedio de la precisión del clasificador K-nn para el conjunto C1 de datos en los cuatro tipos de preprocesamiento. Se señala que a partir del 60 % de datos para el entrenamiento, el mejor desempeño se obtiene

cuando se convierten a minúsculas los caracteres y se eliminan los signos de puntuación.

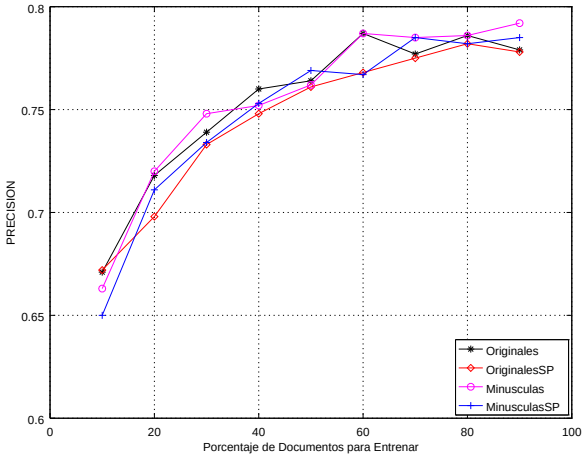

Figura 5.6: Comportamiento del desempeño en la precisión del clasificador K-nn en el conjunto de datos C1, para los cuatro tipos de preprocesamiento de los datos.

El resultado del comportamiento del clasificador, respecto de los tipos de proeprocesamiento de datos y la división de los mismos para el entrenamiento y prueba en el conjunto C2 de datos, se muestran en la figura 5.7. Los resultados en esta evaluación son muy similares.

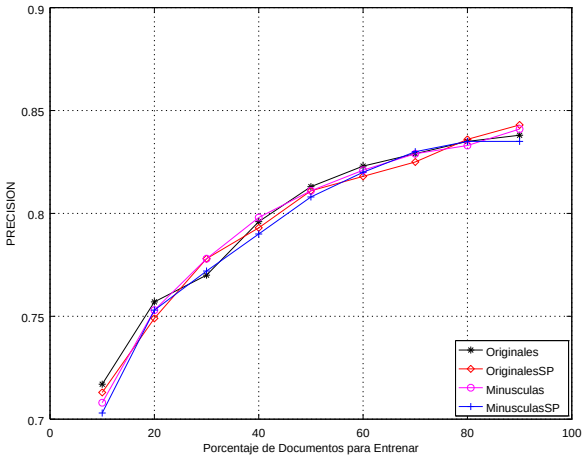

Figura 5.7: Comportamiento del desempeño en la precisión del clasificador K-nn en el conjunto de datos C2, para los cuatro tipos de preprocesamiento de los datos.

Los resultados del clasificador K-nn son muy similares al aplicar los preprocesamientos de los datos 2, 3 y 4. Con el valor de k utilizado en la experimentación, se observó

una disminución en lo general del desempeño del clasificador con los preprocesamientos de los datos 2,3 y 4 respecto del tipo 1.

Por otra parte, surge la pregunta, ¿qué desempeño podrá tener el clasificador K-nn al variar el parámetro K?. Por lo que para observar el comportamiento del clasificador K-nn al variar el valor de K, se realizaron experimentos para valores de K seleccionados de diferentes formas: a) siguiente valor impar del número por default (3); b) un número al azar, de preferencia impar; c) aproximadamente el 10 % de la cantidad de tweets en uno de los conjuntos; , ver la tabla 5.21 que muestra los diferentes valores utilizados en los experimentos.

Elección de K	**K (Conjunto C1)**	**K (Conjunto C2)**
Siguiente impar de 3	5	5
al azar	23	23
10 % de Tweets en C1	103	103

Tabla 5.21: Diferentes valores de K utilizados en los experimentos para el algoritmo K-nn.

Conjunto de datos C1

Se observará primero los resultados obtenidos al utilizar el conjunto de datos C1. En la figura 5.8 se muestran los resultados del clasificador K-nn para los diferentes valores de K descritos en la tabla 5.21, empleando el tipo de preprocesamiento 1 de los datos. Se observa que $K = 3$ proporciona los mejores resultados ante las condiciones de elección de K que se establecieron, seguido por $K = 5$ que solamente cuando la cantidad de datos para entrenar fue del 90 % se encuentra al mismo desempeño que con $K = 3$. También se observa que con los valores de $K = \{5, 23\}$ el clasificador no tiene un sobre entrenamiento de los datos, a diferencia de lo que se describió para $k = 3$ anteriormente. Para este conjunto de datos y para este tipo de preprocesamiento, no se observó una mejora significativa en el desempeño del clasificador, si no que al contrario, se va empeorando el desempeño conforme se incrementa el valor de K utilizado. Elegir el valor de K en relación a un porcentaje de la cantidad de documentos en el conjunto de datos ($K = 103$) se observó el peor desempeño del clasificador.

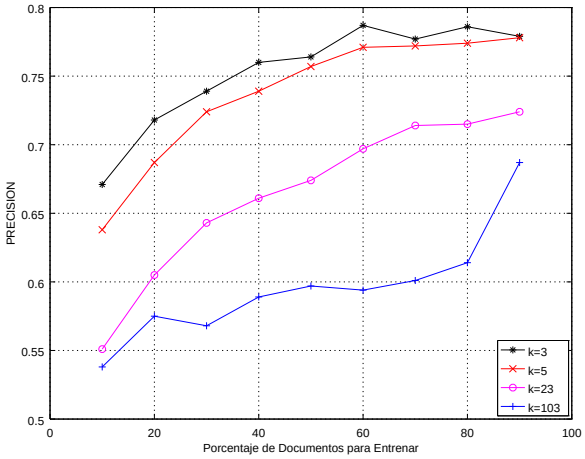

Figura 5.8: Comportamiento del desempeño del clasificador K-nn para diferentes valores impares de K en el conjunto de datos C1 y el tipo de preprocesamiento 1 de los datos.

Conjunto de datos C2

En el experimento con el conjunto C2, cuyos resultados se presenta en la figura 5.9, tienen las mismas características descritas para el conjunto C1, excepto que en ningún valor de K se observa sobreentrenamiento de los datos. Al igual que con el C1, con el valor de $K = 3$ se sigue obteniendo el mejor rendimiento. Como dato adicional, con este conjunto de datos C2 se observa que con los diferentes valores de K se tiene un un buen aprendizaje, ya que sus gráficas de desempeño muestran un comportamiento creciente en todos los casos.

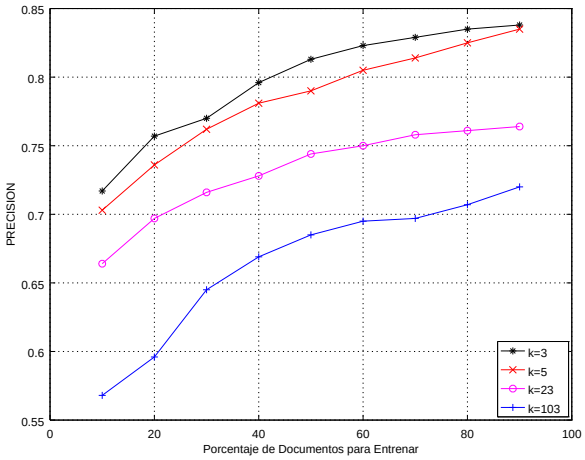

Figura 5.9: Comportamiento del desempeño del clasificador K-nn para diferentes valores impares de K en el conjunto de datos C2 y el tipo de preprocesamiento 1 de los datos.

La conclusión que se puede hacer en la experimentación con este clasificador se plantea de dos formas: 1) Es necesario sintonizar el valor de k para cada uno de los conjuntos de datos utilizados, con la finalidad de mejorar el desempeño; 2) El clasificador K-nn requiere de técnicas más sofisticadas para determinar las mejores características que permiten generalizar el aprendizaje de los datos. La decisión a tomar para implementar un sistema con este clasificador en las condiciones que se experimentó, nos indica que no es necesario realizar algún tipo de preprocesamiento de datos, el comportamiento de desempeño son similares y en ocasiones puede empeorar. Se recomienda el uso valores impares de K, como fue el caso de nuestro experimento, en donde al utilizar valores pares, el desempeño del clasificador tiende a disminuir.

5.5.3. Clasificador SVM

Con el clasificador SVM, los valores de los parámetros que se pueden manipular en la implementación de `sklearn`, fueron los que se encuentran por defecto. La evaluación realizada fue la misma que los dos clasificadores anteriores.

Preprocesamiento 1: Tweets Originales

Primero se obtienen los resultados de precisión que arrojó el clasificador SVM en las 10 veces que se realizó el experimento para cada una de las divisiones o particionamiento que se realizó con los datos para el entrenamiento y prueba del clasificador. Dichos resultados para el conjunto C1 se muestran en la tabla 5.22. Se observa que los mejores resultados se obtuvieron en la división de los datos 90 % - 10 % superando en su mayoría el 81 % de efectividad la clasificación de los tweets. La mayor precisión obtenida en este experimento fue de 0.887 en la primer ejecución de la división de datos correspondiente.

Para el conjunto C2 de datos, los resultados para cada una de las ejecuciones del clasificador se muestran en la tabla 5.23. Se observa que para esta muestra de datos se obtienen mejores resultados, los cuales también se obtuvieron en la división de los datos 90 %-10 %, superando en su mayoría el 89 % de efectividad la clasificación de los tweets. La mayor precisión obtenida en este experimento fue de 0.921, la cual se observa en el experimento número 6.

Tabla 5.22: Resultados de la precisión del clasificador SVM en 10 ejecuciones del experimento con los tweets originales (conjunto C1).

# Experimento	***Cantidad en % de Documentos en el Entrenamiento***								
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
1	0.722	0.742	0.807	0.824	0.805	0.835	0.810	0.840	**0.887**
2	0.727	0.743	0.770	0.804	0.822	0.851	0.851	0.862	0.830
3	0.688	0.766	0.790	0.803	0.805	0.811	0.822	0.840	0.816
4	0.719	0.761	0.783	0.791	0.813	0.786	0.841	0.855	0.823
5	0.694	0.766	0.751	0.797	0.825	0.835	0.844	0.862	0.816
6	0.703	0.737	0.741	0.803	0.803	0.833	0.844	0.805	0.766
7	0.696	0.745	0.775	0.804	0.821	0.811	0.851	0.823	0.858
8	0.691	0.758	0.781	0.791	0.806	0.822	0.839	0.823	0.858
9	0.730	0.740	0.776	0.789	0.809	0.806	0.844	0.809	0.816
10	0.715	0.758	0.798	0.785	0.823	0.813	0.813	0.823	0.837

Tabla 5.23: Precisión del clasificador SVM en los 10 experimentos con los tweets originales (conjunto C2).

# Experimento	***Cantidad en % de Documentos en el Entrenamiento***								
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
1	0.780	0.809	0.841	0.868	0.877	0.884	0.892	0.888	0.902
2	0.763	0.827	0.835	0.845	0.867	0.871	0.890	0.900	0.902
3	0.770	0.814	0.842	0.851	0.872	0.888	0.886	0.914	0.901
4	0.779	0.814	0.829	0.849	0.858	0.881	0.893	0.910	0.916
5	0.772	0.813	0.843	0.855	0.867	0.874	0.896	0.918	0.899
6	0.770	0.798	0.839	0.854	0.871	0.875	0.889	0.900	**0.921**
7	0.767	0.807	0.843	0.858	0.870	0.876	0.901	0.917	0.897
8	0.756	0.812	0.841	0.859	0.867	0.869	0.889	0.900	0.906
9	0.759	0.817	0.833	0.860	0.864	0.875	0.891	0.911	0.894
10	0.759	0.815	0.830	0.854	0.875	0.877	0.886	0.908	0.891

Al obtener el valor promedio de los datos obtenidos en los 10 experimentos que se realizaron para el clasificador en cada corpus de datos utilizados, se observa un un comportamiento creciente del clasificador hasta la división de los datos del 60 %. Las fluctuaciones posteriores nos indican que el clasificador sufre de una mala generalización en el aprendizaje de los datos, por lo que es necesario sintonizar alguno de los parámetros del clasificador o en su defecto aplicar algún tipo de preprocesamiento de los datos para obtener un conjunto de características que permitan mejorar el aprendizaje del clasificador. La tabla 5.24 nos muestra el promedio de las 10 corridas del experimento con cada corpus del clasificador SVM, en donde se observa que el mejor desempeño se encuentra en la división de datos del 70 %

para la muestra C1 y en la división de datos del 80 % para la muestra C2 respectivamente, lo que corresponde a los puntos de máximos de la gráfica que se muestra en la figura 5.10 para el tipo de preprocesamiento 1 de los datos.

Tabla 5.24: Promedio de efectividad del clasificador SVM, mostrando el comportamiento en ambos corpus de datos y la división de datos para el entrenamiento con los tweets originales.

Efectividad Promedio		
División de datos (%)	**Conjunto C1**	**Conjunto C2**
10	0.709	0.768
20	0.752	0.813
30	0.777	0.838
40	0.799	0.855
50	0.813	0.869
60	0.820	0.877
70	***0.836***	0.891
80	0.834	***0.907***
90	0.830	0.903

Preprocesamiento 2: Tweets Originales sin Signos de Puntuación

El segundo preprocesamiento de los datos, el cual contiene un número menor de palabras o características a discriminar por el clasificador, al cual se sometió el clasificador SVM en 10 ejecuciones por cada particionamiento de los datos, seleccionados en forma aleatoria en cada ocasión, y obteniendo los resultados promedio de precisión en la clasificación de cada uno de los conjuntos, los cuales se presentan en la tabla 5.25. Para ambos conjuntos, los mejores resultados se observan cuando el 90 % de datos son utilizados para entrenar el clasificador. En el caso de la muestra C1 se puede observar el comportamiento creciente del desempeño del clasificador en la división de datos del 60 %, primer máximo local encontrado en la figura 5.10 para los datos con preprocesamiento cuya leyenda en la misma gráfica es *originalesSP M1*.

Los resultados del clasificador SVM al aplicar el preprocesamientos de los datos 2 son inferiores respecto al preprocesamiento 1 hasta la división de los datos del 50 % en el conjunto C1 y en la división de datos del 40 % en el conjunto C2, posterior a ello son variables los resultados en ambos casos, ver la figura 5.10. Ver las tablas 5.24 y 5.25 en donde se observa que los mejores resultados promedio de precisión del clasificador se encuentran en

Tabla 5.25: Precisión promedio de ambos conjuntos en el clasificador SVM con el segundo preprocesamiento de los datos.

Efectividad Promedio		
Porcentaje (%)	**Conjunto C1**	**Conjunto C2**
10	0.703	0.766
20	0.745	0.807
30	0.769	0.835
40	0.795	0.854
50	0.811	0.871
60	0.827	0.882
70	0.826	0.893
80	0.837	0.902
90	***0.841***	***0.908***

la división de datos del 90 % en ambos conjuntos de tweets utilizadas en la experimentación con el preprocesamiento 2.

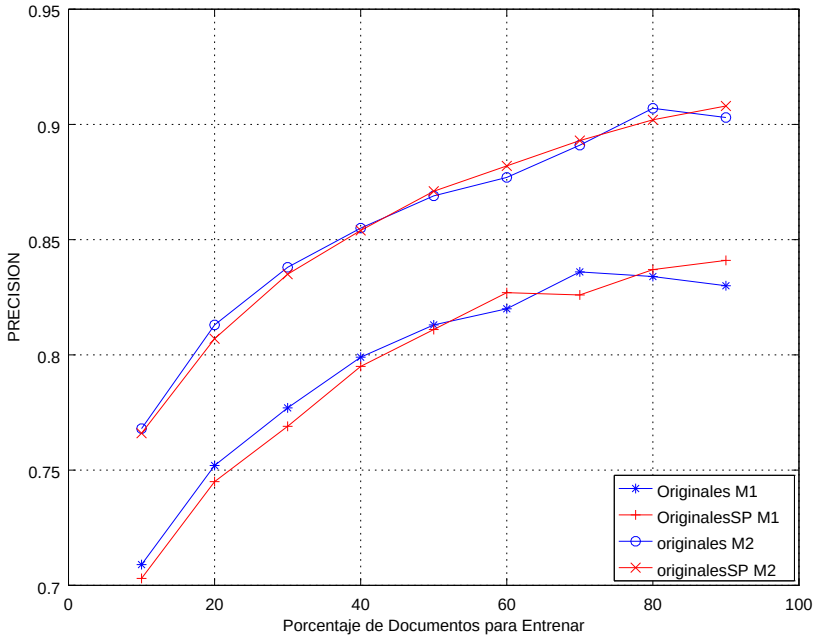

Figura 5.10: Puntos máximos en el comportamiento del clasificador SVM para los tipos de preprocesamiento 1 y 2 en ambos conjuntos de datos (C1 = conjunto C1, C2 = conjunto C2)

Tabla 5.26: Promedio de la precisión del clasificador SVM en 10 ejecuciones con el preprocesamiento 3 en ambos conjuntos de datos.

Efectividad Promedio		
Porcentaje (%)	**Conjunto C1**	**Conjunto C2**
10	0.703	0.770
20	0.745	0.810
30	0.780	0.834
40	0.8	0.855
50	0.820	0.869
60	0.830	0.880
70	0.827	0.897
80	0.838	0.899
90	***0.853***	***0.906***

Preprocesamiento 3: Tweets con Texto en Minúsculas

En el tercer tipo de preprocesamiento de los datos, los resultados del clasificador SVM indican que para el conjunto C1 de datos se tiene un punto máximo local en la división de datos del 60 %, lo que sugiere un cambio en los parámetros del clasificador o preprocesamiento de los datos; mientras que para el conjunto de datos C2, el comportamiento del clasificador es creciente, observándose el mejor desempeño en la división de datos del 90 % con una precisión promedio de 0.906, ver la tabla 5.26.

Los resultados en relación a los preprocesamientos de datos 1 y 2 se superan en los rangos de división de los datos del 30 % al 60 % y del 80 % al 90 % para el conjunto C1 de datos, mientras que para el conjunto C2 de datos los resultados son muy similares, ver las figuras 5.11 y 5.12 respectivamente.

Preprocesamiento 4: Tweets con Texto en Minúsculas sin Signos de Puntuación

Los resultados obtenidos para el clasificador SVM con este tipo de preprocesamiento, muestran un comportamiento creciente hasta la división de datos del 70 % en el conjunto C1. Para el conjunto C2, el comportamiento es totalmente creciente, obteniendo una precisión del .905 como máximo desempeño promedio en la división de datos del 90 %. Ver la tabla 5.27 y figuras 5.11 y 5.12 que muestran los resultados descritos.

De forma general, para ambos conjuntos de datos y aplicando los diferentes tipos de

Tabla 5.27: Comportamiento promedio del clasificador SVM con el tipo 4 de preprocesamiento de los datos en ambos conjuntos.

Efectividad Promedio		
Porcentaje (%)	**Conjunto C1**	**Conjunto C2**
10	0.705	0.765
20	0.752	0.805
30	0.770	0.834
40	0.796	0.854
50	0.805	0.869
60	0.822	0.881
70	0.832	0.893
80	0.829	0.896
90	***0.856***	***0.905***

preprocesamiento de datos planteados en este trabajo, no se observa una mejora significativa en el desempeño del clasificador, por lo que se consideran muy similares.

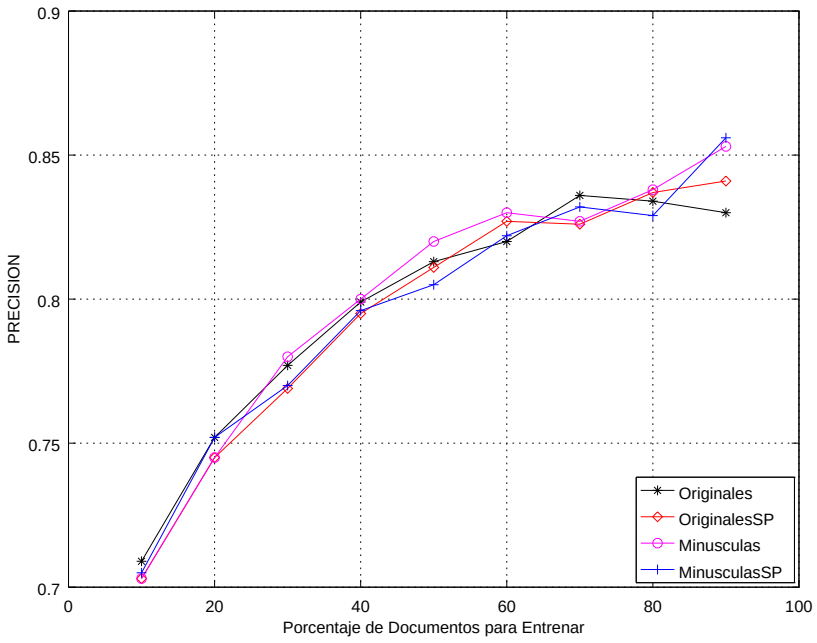

Figura 5.11: Comportamiento del clasificador SVM, para el conjunto C1 de datos, en relación a la partición de los datos utilizados para el entrenamiento con los cuatro tipos de preprocesamiento

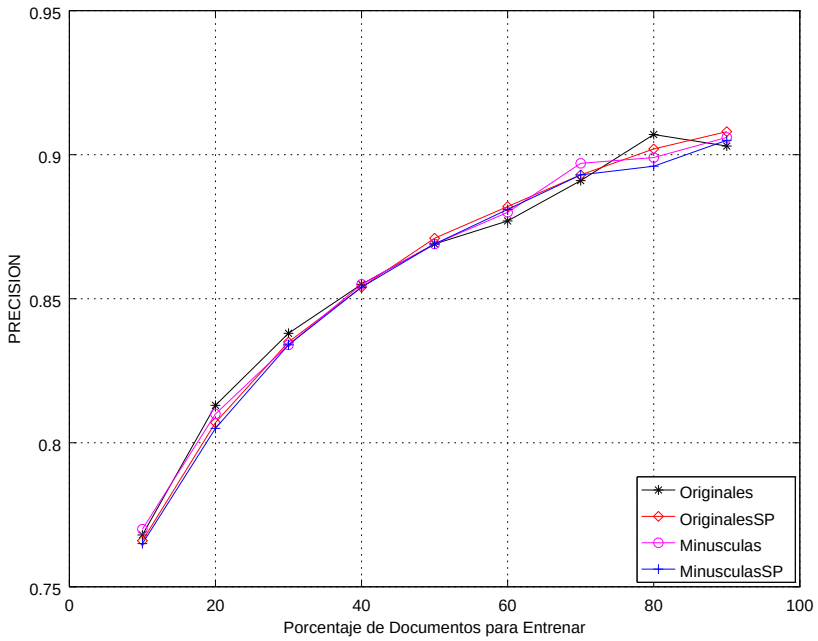

Figura 5.12: Comportamiento del clasificador SVM con el conjunto C2 de datos, en relación a la partición de datos para el entrenamiento y los cuatro tipos de preprocesamiento de datos planteados.

5.6. Resumen General

Los resultados en la experimentación de los tres distintos clasificadores mostrados en la sección anterior arrojaron datos interesantes en cuanto al comportamiento de cada uno de estos al aplicarles cada uno de los distintos tipos de preprocesamientos: *1) Tweets originales, 2) tweets originales sin signos de puntuación, 3) tweets en minúsculas y 4) tweets en minúsculas sin signos de puntuación.* Con la precisión obtenida en cada clasificador durante la experimentación bajo las condiciones planteadas en este trabajo, se pudo observar que solamente el clasificador NB obtuvó una mejora significativa del 2 % en el desempeño al aplicar los tipos de preprocesamiento en ambos conjuntos de datos, ver las tablas 5.28 y 5.29.

En ambos conjuntos de datos, el clasificador NB es el de mejor desempeño independientemente del tipo de preprocesamiento de datos que se aplicó. Para los clasificadores K-nn y SVM el conjunto de datos C2 es más fácil de realizar la tarea de aprendizaje y

Tabla 5.28: Resumen general en cuanto a la efectividad de los diferentes clasificadores con respecto a los 4 preprocesamientos realizados (Conjunto C1).

Clasificadores	***Preprocesamientos***			
	Originales	**Orig. sin s/p**	**Minúsculas**	**Min sin s/p**
NB	0.954	0.954	0.975	0.975
K-nn	0.787	0.782	***0.792***	0.785
SVM	0.836	0.841	0.853	***0.856***

Tabla 5.29: Resumen general en cuanto a la efectividad de los diferentes clasificadores con respecto a los 4 preprocesamientos realizados (Conjunto C2).

Clasificadores	***Preprocesamientos***			
	Originales	**Orig. sin s/p**	**Minúsculas**	**Min sin s/p**
NB	0.950	0.950	0.975	0.975
K-nn	0.838	***0.843***	0.841	0.835
SVM	0.907	***0.908***	0.906	0.905

clasificación, lo que les permitió obtener un mejor desempeño con este conjunto de datos, mientras que el clasificador NB su mejor desempeño se obtuvó con el conjunto C1. ver tabla 5.28.

Capítulo 6

Conclusiones y Trabajos Futuros

6.1. Conclusiones

En el desarrollo de un sistema en el que se hace necesario integrar alguna de las técnicas de Aprendizaje de Máquina, tales como clasificadores de texto, se hace necesario tener algunas referencias del desempeño de dichos mecanismos. Este trabajo contiene una serie de experimentos en la tarea de clasificación de texto, en la que se emplearon dos conjuntos de datos de texto en español, documentos cortos que se obtuvieron de una red social, en específico Twitter.

Se cuenta con una referencia del desempeño, en relación a la precisión, de tres de los clasificadores de texto más comúnmente utilizados en la literatura. Además se muestra su integración a un sistema web, mostrando el comportamiento de la precisión respecto a la cantidad de datos que se tienen para entrenar los clasificadores.

Para determinar el tipo de clasificador de texto a utilizar en un sistema en producción, se presenta el tipo de evaluación mínima que se debe realizar a los sistemas de clasificación de texto. Se incluyen también breves observaciones que pueden encontrarse en el comportamiento de un clasificador, proporcionando las sugerencias de que hacer con los datos o con el clasificador.

Se creo una colección de datos de texto para la clasificación de opinión. Este conjunto de datos consiste de dos muestras de tweets etiquetadas que pueden ser utilizadas

como referencia para otros trabajos futuros.

Factores como el conjunto de datos, su cantidad y tipo de preprocesamiento que se va a manejar en el sistema final, determina la técnica de aprendizaje a utilizar. En nuestra evaluación, el clasificador de Bayes para ambas muestras de datos nos proporcionó los mejores resultados, tomando como condiciones base en la experimentación los parámetros por defecto de los clasificadores.

6.2. Trabajos Futuros

Algunos de los trabajos futuros que se proponen en relación a este trabajo son:

1. Identificar la influencia de los emoticones o secuencia de caracteres que representan un sentimiento, en el comportamiento de desempeño de un sistema de opinión de sentimiento.

2. Normalización de abreviaturas más habituales: Como en el caso de los signos de puntuación, dada la falta de espacio de la que disponen los usuarios en entornos web para expresar sus opiniones, es habitual encontrarse con abreviaturas no reconocidas. Así elementos de palabras simples como *'x'* o *'q'* y son reemplazadas por *'por'* o *'que'*, respectivamente. También frases o expresiones como *'lol'* o *'tqm'* y son reemplazadas por *'risas'* o *'te quiero mucho'*.

3. Eliminación de hashtags ('#'). Los hastags son términos incluidos en Twitter que los usuarios preceden del símbolo '#' (comúnmente llamado gato), con el objetivo de etiquetar sus mensajes. Al hacer click sobre un hashtag el usuario es redireccionado al conjunto de tweets que contienen la misma etiqueta. Sin embargo, es habitual que el período de vida de los hashtags sea muy corto, dado que suelen referir eventos muy específicos (e.g. '#MonarcasVSChivas', '#SuperBowl'). Por ello, este tipo de hashtags que sirven para clasificar según eventos concretos, y que son situados bien al principio o al final del tweet, son eliminados. También es frecuente utilizarlos como medio para enfatizar una palabra contenida en un mensaje (e.g. 'La #felicidad es algo difícil de conseguir') o bien enfatizar un mismo mensaje (e.g. '#actitudPositiva').

4. Simplificación de enlaces. En esta red social es habitual que los usuarios enlacen recursos externos, como imágenes o direcciones a otras páginas web.

5. Realizar pruebas con un preprocesamiento adicional de los datos, el cual consiste en la eliminación de palabras vacías en los tweets. Las palabras vacías se refieren a los artículos, preposiciones, conjunciones, etcétera, es decir, aquellas palabras que no aportan ninguna información semántica al análisis, algunos ejemplos de unas pocas de estas palabras en el idioma español son: {*las, los, la, el, con, y, de, desde, por, en, sin* }

6. En investigaciones recientes han surgido nuevos modelos para la representación de palabras, uno de ellos se refiere a la distribución vectorial de las palabras, en donde cada palabra puede ser representada en un vector de dimensión baja y entonces poder realizar operaciones vectoriales que permiten obtener relaciones semánticas entre las palabras. Una de las técnicas más populares en esete sentido se refieren a `word2vec` [53, 54].

7. Incluir en los clasificadores binarios una tercer clase, *NEUTRO*, en donde se identifique el sentimiento neutro en los fragmentos de texto.

8. Realizar la evaluación de los clasificadores con otras métricas que son empleadas en los sistemas de Recuperación de Información, tales como: recall y F1, en sus formas macro y micro.

9. Implementar algunas de las técnicas de selección de características y realizar la evaluación de desempeño de los clasificadores.

10. Utilizar como características pares de palabras (bigramas) o $n - palabras$ (n-gramas) en la representación de los documentos.

11. Construir conjuntos de datos de tópicos específicos (temas de política actual, comentarios de productos, servicios, deportes, etcétera) y construir un sistema que a partir del corpus nos permita visualizar las tendencias de opinión en dichos conjuntos de datos, es decir, señalar la mayoría del sentido de las opiniones.

12. Experimentar con diferentes valores de parámetros en los clasificadores Knn y SVM y realizar una caracterización del comportamiento de los clasificadores en base al tipo de datos utilizados.

Apéndice A

Código fuente de los clasificadores: NB, Knn y SVM.

1. **Código fuente del algoritmo que guarda el modelo del clasificador NB.**

```
import re, math, collections, itertools, os
import nltk, nltk.classify.util, nltk.metrics
import cPickle
import sys
import random
from cPickle import dump,dumps,load,loads

from nltk.metrics.scores import (accuracy, precision, recall, \
f_measure,log_likelihood, approxrand)

from nltk.classify import NaiveBayesClassifier
from nltk.metrics import BigramAssocMeasures
from nltk.probability import FreqDist, ConditionalFreqDist

# ruta  donde se leen los archivos para entrenar el sistema
POLARITY_DATA_DIR = os.path.join('polarityData', 'Tweets')

RT_POLARITY_POS_FILE = os.path.join(POLARITY_DATA_DIR, 'Positivos1.txt')
RT_POLARITY_NEG_FILE = os.path.join(POLARITY_DATA_DIR, 'Negativos1.txt')

#Esta FUNCION toma un mecanismo de SELECCION de CARACTERISTICAS y devuelve
#su rendimiento en una variedad de METRICAS
def evaluate_features(feature_select):
    posFeatures = []
    negFeatures = []
    num=sys.argv[1]
    numInt=int(num)

    #Divide las frases (tweets) en listas de palabras individuales
    #(seleccionadas por el mecanismo de entrada) y agrega 'pos' o 'neg'
```

```
    #DESPUES de cada palabra en la lista
    with open(RT_POLARITY_POS_FILE, 'r') as posSentences:
            for i in posSentences:
                    posWords = re.findall(r"[\w']+|[.,!?;]", i.rstrip())
                    posWords = [feature_select(posWords), 'pos']
                    posFeatures.append(posWords)
    with open(RT_POLARITY_NEG_FILE, 'r') as negSentences:
            for i in negSentences:
                    negWords = re.findall(r"[\w']+|[.,!?;]", i.rstrip())
                    negWords = [feature_select(negWords), 'neg']
                    negFeatures.append(negWords)

    #selecciona un porcentaje (%) de los Tweets  para usarse como
    # entrenamiento y el resto es usado para pruebas
    posCutoff = int(math.floor(len(posFeatures)*numInt/10))
    negCutoff = int(math.floor(len(negFeatures)*numInt/10))

    #Se mezclan aleatoriamente los arreglos que contienen los tweets para entrenar con
    diferentes caracteristicas
    random.shuffle(posFeatures)
    random.shuffle(negFeatures)

    #Se juntan los Tweets positivos y negativos, para el entrenamiento y prueba
    trainFeatures = posFeatures[:posCutoff] + negFeatures[:negCutoff]
    testFeatures = posFeatures[posCutoff:] + negFeatures[negCutoff:]

    #Se entrena el Clasificador NB
    classifier = NaiveBayesClassifier.train(trainFeatures)

    #Se guarda el modelo del Clasificador
    with open('modeloEspanol.pkl', 'wb') as fid:
            cPickle.dump(classifier, fid)

    #Inicia los conjuntos de referencia y los conjuntos de prueba
    referenceSets = collections.defaultdict(set)
    testSets = collections.defaultdict(set)

    #coloca las oraciones correctamente etiquetadas en los conjuntos de referencia y la
    version etiquetada predictivamente en el conjunto de prueba

    for i, (features, label) in enumerate(testFeatures):
            referenceSets[label].add(i)
            predicted = classifier.classify(features)
            testSets[predicted].add(i)

    #Crea un mecanismo de SELECCION de funciones que utiliza todas las palabras (Diccionario)

def make_full_dict(words):
    return dict([(word, True) for word in words])

def create_word_scores():
    #Crea listas de todas las palabras positivas y negativas
    posWords = []
    negWords = []
    with open(RT_POLARITY_POS_FILE, 'r') as posSentences:
            for i in posSentences:
                    posWord = re.findall(r"[\w']+|[.,!?;]", i.rstrip())
```

```
                posWords.append(posWord)
    with open(RT_POLARITY_NEG_FILE, 'r') as negSentences:
            for i in negSentences:
                negWord = re.findall(r"[\w']+|[.,!?;]", i.rstrip())
                negWords.append(negWord)
    posWords = list(itertools.chain(*posWords))
    negWords = list(itertools.chain(*negWords))

    #Se construye la DISTRIBUCION de frecuencias de todas las palabras y luego las
    distribuciones de frecuencia de las palabras dentro de las etiquetas positivas
    y negativas
    word_fd = FreqDist()
    cond_word_fd = ConditionalFreqDist()
    for word in posWords:
            word_fd[word.lower()] += 1
            cond_word_fd['pos'][word.lower()] += 1
    for word in negWords:
            word_fd[word.lower()] += 1
            cond_word_fd['neg'][word.lower()] += 1

    #Encuentra el NUMERO de palabras positivas y negativas, ASI como el NUMERO
    total de palabras
    pos_word_count = cond_word_fd['pos'].N()
    neg_word_count = cond_word_fd['neg'].N()
    total_word_count = pos_word_count + neg_word_count

    #Construye un diccionario de puntuaciones de palabras
    word_scores = {}
    for word, freq in word_fd.iteritems():
            pos_score = BigramAssocMeasures.chi_sq
            (cond_word_fd['pos'][word],
            (freq, pos_word_count), total_word_count)
            neg_score = BigramAssocMeasures.chi_sq(cond_word_fd['neg'][word]
            , (freq, neg_word_count), total_word_count)
            word_scores[word] = pos_score + neg_score

    return word_scores

#Encuentra la PUNTUACION de las palabras
word_scores = create_word_scores()

#finds the best 'number' words based on word scores
#Encuentra el mejor numero de palabras basadas en puntajes de palabras
def find_best_words(word_scores, number):
    best_vals = sorted(word_scores.iteritems(), key=lambda (w, s): s, reverse=True)
    [:number]
    best_words = set([w for w, s in best_vals])
    return best_words

#Crea un mecanismo de SELECCION de funciones que SOLO utiliza las mejores palabras
def best_word_features(words):
    return dict([(word, True) for word in words if word in best_words])
```

2. **Código fuente del algoritmo que carga el Modelo del Clasificar NB para calcular la PRECISION de este.**

```
import nltk.classify.util
from nltk.classify import NaiveBayesClassifier
import cPickle
from nltk.tokenize import word_tokenize
import sys, re

def word_feats(words):
    return dict([(word, True) for word in words])

negFeatures=[]
posFeatures=[]

#Recibe como parametros los Tweets positivos y negativos
with open(sys.argv[1], 'r') as posSentences:
                for i in posSentences:
                        posWords = re.findall(r"[\w']+|[.,!?;]", i.rstrip().lower())
                        posWords = [word_feats(posWords), 'pos']
                        posFeatures.append(posWords)
with open(sys.argv[2], 'r') as negSentences:
                for i in negSentences:
                        negWords = re.findall(r"[\w']+|[.,!?;]", i.rstrip().lower())
                        negWords = [word_feats(negWords), 'neg']
                        negFeatures.append(negWords)

testfeats=negFeatures+posFeatures

#Carga el Clasificador NB
with open('modeloEspanol.pkl', 'rb') as fid:
    classifier=cPickle.load(fid)

#Retorna la efectividad de la prueba
print nltk.classify.util.accuracy(classifier, testfeats)
```

3. **Código fuente del algoritmo que guarda el Modelo aprendido del Clasificar KNN.**

```
import re, math, collections, itertools, os
import nltk, nltk.classify.util, nltk.metrics
import cPickle
import sklearn
from cPickle import dump,dumps,load,loads
from nltk.metrics import BigramAssocMeasures
from nltk.probability import FreqDist, ConditionalFreqDist
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
#Se crean los arreglos
posFeatures = []
negFeatures = []
posTrain = []
negTrain = []
```

```
posTest = []
negTest = []

#Extraigo los Tweets positivos y negativos de los archivos
archivoPos = open('PositivosSP.txt','r')
leeLineaPos = archivoPos.readlines()
archivoPos.close()

for listaPos in leeLineaPos:
    posFeatures.append(listaPos)

archivoNeg = open('NegativosSP.txt','r')
leeLineaNeg = archivoNeg.readlines()
archivoNeg.close()

for listaNeg in leeLineaNeg:
    negFeatures.append(listaNeg)

#Asigno un porcentaje de los Tweets para entrenamiento y el resto para prueba
posCutoff = int(math.floor(len(posFeatures)*10/10))
negCutoff = int(math.floor(len(negFeatures)*10/10))

#Asigno los Tweets para entrenamiento y prueba
trainFeatures = posFeatures[:posCutoff] + negFeatures[:negCutoff]
testFeatures = posFeatures[posCutoff:] + negFeatures[negCutoff:]

#creo los vectores con las respectivas etiquetas SEGUN los datos de entrenamiento
for i in posFeatures[:posCutoff]:
    posTrain.append('pos')

for i in negFeatures[:negCutoff]:
    negTrain.append('neg')

#creo los vectores con las respectivas etiquetas segun los datos de prueba
for i in posFeatures[posCutoff:]:
    posTest.append('pos')

for i in negFeatures[negCutoff:]:
    negTest.append('neg')

#Se crea el vector X para los datos de entrenamiento
vectorizer = TfidfVectorizer(encoding='latin1')
X_train = vectorizer.fit_transform(f for f in trainFeatures)

#Se crea el vector Y para los datos de entrenamiento
Y_train = posTrain + negTrain

Y_test = posTest + negTest

# Para el clasificador Knn
neigh = KNeighborsClassifier(n_neighbors=3)

neigh.fit(X_train, Y_train)

# Guardar los modelos aprendidos para poder utilizarlos posteriormente
with open('modelotweetsKnn.pkl','wb') as fid:
                cPickle.dump(neigh,fid)
```

```
with open('modelotweetsVectorizerTFIDF.pkl','wb') as fid:
                cPickle.dump(vectorizer,fid)
```

4. **Código fuente del algoritmo que lee el Modelo aprendido del Clasificar KNN.**

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
# -*- coding: 850 -*-

import re, math, collections, itertools, os
import nltk, nltk.classify.util, nltk.metrics
import cPickle
import sklearn
import random
from cPickle import dump,dumps,load,loads
from nltk.metrics import BigramAssocMeasures
from nltk.probability import FreqDist, ConditionalFreqDist
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from nltk.tokenize import word_tokenize
import sys

def word_feats(words):
    return dict([(word, True) for word in words])

# leer el clasificador
with open('modelotweetsKnn.pkl', 'rb') as fid:
    classifier=cPickle.load(fid)
with open('modelotweetsVectorizerTFIDF.pkl', 'rb') as fid:
    vectorizer=cPickle.load(fid)

negFeatures=[]
#se abre el documentos que contiene las frases a predecir
with open('frases.txt', 'r') as negSentences:
                for i in negSentences:
                        negWords = re.findall(r"[\w']+|[.,!?;]", i.rstrip().lower())
                        negFeatures.append(' '.join(map(str,negWords)))
X_test=vectorizer.transform(negFeatures)
prediccion=classifier.predict(X_test)

datos=[]
j=0
#se le asigna la etiqueta adecuada para cada prediccion.
for frase in negFeatures:
    if prediccion[j] == 'pos':
            clase='pos'
    else:
            clase='neg'
    datos.append(str(clase))
    j=j+1
datos.sort()
```

```
archfrases=open("clusters.ordenados","w")
archfrases.write("\"x\n")
j=1
for frase in datos:
    archfrases.write("\""+str(j)+"\" "+frase+"\n")
```

5. **Código fuente del algoritmo que guarda y lee el Modelo del Clasificar SVM a si mismo obtiene la efectividad.**

```
import re, math, collections, itertools, os
import nltk, nltk.classify.util, nltk.metrics
import cPickle
import sklearn
import sys
import random
from cPickle import dump,dumps,load,loads
from nltk.metrics import BigramAssocMeasures
from nltk.probability import FreqDist, ConditionalFreqDist
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.svm import SVC
from sklearn import metrics
posFeatures = []
negFeatures = []
posTrain = []
negTrain = []
posTest = []
negTest = []
num=sys.argv[1]
numInt=int(num)

#Extraigo los Tweets positivos y negativos de los archivos
archivoPos = open('Positivos2.txt','r')
leeLineaPos = archivoPos.readlines()
archivoPos.close()

for listaPos in leeLineaPos:
    posFeatures.append(listaPos)

archivoNeg = open('Negativos2.txt','r')
leeLineaNeg = archivoNeg.readlines()
archivoNeg.close()

for listaNeg in leeLineaNeg:
    negFeatures.append(listaNeg)

#Se mezclan aleatoriamente los arreglos que contienen los tweets para entrenar
con diferentes caracteristicas
random.shuffle(posFeatures)
random.shuffle(negFeatures)

#Asigno un porcentaje de los Tweets para entrenamiento y el resto para prueba
```

```
posCutoff = int(math.floor(len(posFeatures)*numInt/10))
negCutoff = int(math.floor(len(negFeatures)*numInt/10))

#Asigno los Tweets para entrenamiento y prueba
trainFeatures = posFeatures[:posCutoff] + negFeatures[:negCutoff]
testFeatures = posFeatures[posCutoff:] + negFeatures[negCutoff:]

#creo los vectores con las respectivas etiquetas segun los datos de entrenamiento
for i in posFeatures[:posCutoff]:
    posTrain.append('pos')

for i in negFeatures[:negCutoff]:
    negTrain.append('neg')

#creo los vectores con las respectivas etiquetas segun los datos de prueba
for i in posFeatures[posCutoff:]:
    posTest.append('pos')

for i in negFeatures[negCutoff:]:
    negTest.append('neg')

#Se crea el vector X para los datos de entrenamiento
vectorizer = TfidfVectorizer(encoding='latin1')
X_train = vectorizer.fit_transform(f for f in trainFeatures)

#Se crea el vector Y para los datos de entrenamiento
Y_train = posTrain + negTrain

Y_test = posTest + negTest

# Para el clasificador SVM
SVM = SVC()

SVM.set_params(kernel='linear', tol=0.0000001).fit(X_train, Y_train)

# Guardar los modelos aprendidos para poder utilizarlos posteriormente
with open('modelotweetsSVM.pkl','wb') as fid:
                cPickle.dump(SVM,fid)
with open('modelotweetsVectorizerTFIDF.pkl','wb') as fid:
                cPickle.dump(vectorizer,fid)

X_test = vectorizer.transform(f for f in testFeatures)
pred = SVM.predict(X_test)

#Se calcula la efectividad
score = metrics.accuracy_score(Y_test, pred)
print  score
```

Apéndice B

Interconexión y código fuente del Sistema Web.

Interconexión de PHP con HTML

PHP y HTML interactúan: PHP puede generar HTML, y HTML puede pasar información a PHP.

Cómo obtener variables desde fuentes externas

Cuando se envía un formulario a un script de PHP, la información de dicho formulario pasa a estar automáticamente disponible en el script de PHP.

A través de un formulario de HTML se captura la frase que se desea a predecir, el parámetro se guarda en la variable **analizar** a través de la caja de texto y se envía con el atributo **action** que indica la acción que va a realizar, el formulario en este caso el parámetro se envía al script **conexion.php**.

En el script de PHP (**conexion.php**) se dispone el contenido del formulario ingresado en **index.php** el cual se envía a través del método POST, el parámetro se gurda en la variable **predice** (la cual fue definida en el script de PHP como tipo *array*), a la vez se definió otra variable de tipo *array* llamada **salida** (en dicha variable se almacena el contenido que devuelve la predicción de cada clasificador). Con la función **isset** determina

si una variable está definida y no es NULL.

Cuando un usuario rellena un formulario en una página web los datos hay que enviarlos de alguna manera. Vamos a considerar las dos formas de envío de datos posibles:

1. Método GET envía los datos usando la URL
2. Método POST envía los datos de forma que no podemos ver la información enviada (en un segundo plano u ocultos al usuario).

La diferencia entre los métodos GET y post radica en la forma de enviar los datos a la página.

La interconexión de PHP es un lenguaje de programación que se realiza una sincronización con HTML para poder ejecutar el sistema de pagina Web.

Código fuente de interfaz Web del sistema.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
        <!DOCTYPE HTML>

<head>
        <!--link type="text/css" href="estilos.css" rel="stylesheet"-->
<title>Login</title>
</head>

<body BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#0000FF" VLINK="#800080" ALINK="#FF00
00" BACKGROUND="Wallpaper.jpg" LINK=>

        <center><b><a HREF="http://www.umich.mx/"><img src="Umich.png" alt=
        "visita aulaclic" width="160" height="170" border="0"></a><font
        size=5.5 color="Black"> Universidad Michoacana De San Nicolas De Hidalgo</
        font><a HREF="http://www.fie.umich.mx/" ><img src="Fie.gif" alt=
        "visita aulaclic" width="190" height="150" border="0"></a> </b>
        </center>
        <center><b> <font size=5 color="Navy">Facultad De Ingenieria Electrica
        </font></b></center>
        <center><b><a HREF="http://www.umich.mx/licenciatura-ingenieria-computacion.html"><font
        size=5 color="Silver">Ingenieria En Computacion</font>
        </a></b></center>
        <br></br>
        <center><b><font size=6 color=BLACK>Sistema de Predicci&oacute;n de la
        Polaridad en un Texto </font></b></center>
        <center><b><font size=5 color="#0000FF"> Bienvenido!</font>
```

```
        </b></center>
        <br></br>
<center>
        <form action="conexion.php" method="POST" >
        <p><font size=5 color=WHITE> Ingresa Frase a Analizar: </font><br><textarea
        name="analizar" rows="3" cols="40"></textarea> </p>
        <p><font size=5 color=WHITE> Selecciona Clasificador (es) a utilizar:</font><br>
        <input name="bayes" type="checkbox" /><font size=5 color=WHITE >NB</font>
        <input name="knn" type="checkbox" /><font size=5 color=WHITE >Knn</font>
        <input name="svm" type="checkbox" /><font size=5 color=WHITE >SVM</font>
        <br><br>
        <button>
                <b><font size=6 color=BLACK >Analizar</font></b>
        </button>
</form>
</center>
</body>

</html>
```

Código fuente para analizar la polaridad con algún(os) clasificador(es) especifico(s).

```
    <html>
<body>
<center>
<?php
include('index.php');
        $salidaNB = array();
        $salidaKnn = array();
        $salidaSVM = array();
        $predice = array();
        $predice1 = NULL;
        $predice = NULL;
        $predice =$_POST['analizar'];
        $predice1 =$_POST['analizar'];

        echo "<b><font size=5 color=WHITE> Texto Ingresado: </font></b>";
        echo "<br>";
        echo "<font size=5 color=WHITE>$predice</font>";
        echo "<br><br>";
        echo "<b><font size=5 color=WHITE> Polaridad Predecida por los
        Clasificadores: </font></b>";

        if(isset($_POST['bayes'])){
                $comando="python cBayesPredic.py \"$predice\"";
                $output=exec($comando,$salidaNB);
                echo "<br>";
                echo "<b><font size=5 color=WHITE>(NB): </font></b>";
                echo "<font size=5 color=WHITE>$salidaNB[1]</font>";
        }

        if(isset($_POST['knn'])){
```

```
                $comando="python KnnLoadPag.py \"$predice\"";
                $output=exec($comando,$salidaKnn);
                echo "<br>";
                echo "<b><font size=5 color=WHITE>(Knn): </font></b>";
                echo "<font size=5 color=WHITE>$salidaKnn[0]</font>";
        }

        if(isset($_POST['svm'])){
                $comando="python svmLoadPag.py \"$predice\"";
                $output=exec($comando,$salidaSVM);
                echo "<br>";
                echo "<b><font size=5 color=WHITE>(SVM): </font></b>";
                echo "<font size=5 color=WHITE>$salidaSVM[0]</font>";
        }
?>
</center>
</body>
</html>
```

Apéndice C

Código fuente de utilerías para el desarrollo del sistema

1. **Código fuente para obtener la cantidad de palabras y el tamaño del vocabulario del conjunto de datos.**

```
import re, collections

tweetsPositivos = []
tweetsNegativos = []
posWordsAux = []
negWordsAux = []

#Extraigo los Tweets positivos y negativos de los archivos
archivoPos = open('PosMinSP1.txt','r')
leeLineaPos = archivoPos.readlines()
archivoPos.close()

for tweetPos in leeLineaPos:
    tweetsPositivos.append(tweetPos)

archivoNeg = open('NegMinSP1.txt','r')
leeLineaNeg = archivoNeg.readlines()
archivoNeg.close()

for tweetNeg in leeLineaNeg:
    tweetsNegativos.append(tweetNeg)

#Extraigo palabra por palabra de cada Tweet positivo y lo agrego a posWordsAux
for i in tweetsPositivos:
    posWords = re.findall(r"[\w']+|[.,!?;]", i.rstrip())
    posWordsAux += posWords
print ""
print "TWEETS POSITIVOS"
```

```
print "# Palabras: ", len(posWordsAux)

#Se obtiene cada palabra que hay en los Tweets con su respectiva frecuencia
#frecuenciaPalabras = collections.Counter(posWordsAux)
#print frecuenciaPalabras

#Se obtiene la cantidad de palabras unicas, donde:
    #clave = palabra unica en los tweets
    #valor = fracuencia de cada palabra unica

palabrasPos = []
contadorPos = collections.Counter(posWordsAux)
for clavePos,valorPos in contadorPos.items():
    #print clavePos, " : ", valorPos
    palabrasPos.append(clavePos)
print "# Palabras unicas", len(palabrasPos)
print ""

#Extraigo palabra por palabra de cada Tweet negativo y lo agrego a negWordsAux
for j in tweetsNegativos:
    negWords = re.findall(r"[\w']+|[.,!?;]", j.rstrip())
    negWordsAux += negWords
print "TWEETS NEGATIVOS"
print "# Palabras: ", len(negWordsAux)

#Se obtiene cada palabra que hay en los Tweets con su respectiva frecuencia
#frecuenciaPalabras = collections.Counter(posWordsAux)
#print frecuenciaPalabras

palabrasNeg = []
contadorNeg = collections.Counter(negWordsAux)
for claveNeg,valorNeg in contadorNeg.items():
    #print claveNeg, " : ", valorNeg
    palabrasNeg.append(claveNeg)
print "# Palabras unicas", len(palabrasNeg)
print ""

print "PALABRAS TOTALES  EN AMBAS PORCIONES"
print "# Palabras totales: ", len(posWordsAux)+len(negWordsAux)
print "# Palabras unicas totales: ", len(palabrasNeg)+len(palabrasPos)
print ""
```

2. Código fuente para eliminar signos de puntuación de un archivo de texto.

```
#include<stdio.h>
#include<ctype.h>

void eliminaSignosPuntuacion(FILE * arch1, FILE * arch2){
    char C;
    while((C=fgetc(arch1))!=EOF){
            if(ispunct(C)==0){
                        fprintf(arch2,"%c",C);
            }
    }
}
```

```
main(){
    FILE *cor,*corSP;
    cor = fopen("Negativos1.txt","r+");
    corSP = fopen("corpusSP.txt","w+");

    eliminaSignosPuntuacion(cor,corSP);

    fclose(cor);
}
```

3. Código fuente para Convertir mayúsculas en minúsculas de un archivo.txt

```
cat archivo1.txt | tr [:upper:] [:lower:] > archivo2.txt
```

4. Código fuente para Convertir minúsculas en mayúsculas de un archivo.txt

```
cat archivo1.txt | tr [:lower:] [:upper:] > archivo2.txt
```

5. Código fuente para categorizar los Tweets.

```
#Programa que invierte una cadena en base a un archivo que resive como entrada
#!/usr/bin/python
import time
Pos= open('Positivos.txt','w')
Neg= open('Negativos.txt','w')
Neu= open('Neutros.txt','w')
File= open('tuitsPolaridad.txt','r')
#File= open('clasi.txt','r')
invertida=[]
for linea in File.readlines():
    l=linea.split(' ')
    if l[1] == 'N\r\n' or l[1] == 'N+\r\n' :
            Neg.write(l[0]+'\n')
    elif l[1] == 'P\r\n' or l[1] == "P+\r\n":
            Pos.write(l[0]+'\n')
    elif l[1] == "NEU\r\n":
            Neu.write(l[0]+'\n')
File.close()
Pos.close()
Neg.close()
Neu.close()
```

6. Código fuente para descargar los Tweets.

```
# Como ejecutar el script:
# python descargarTweets.py "TemasAdescargar"

import tweepy
import json, sys
```

```
# Consumer keys and access tokens, used for OAuth
consumer_key = 'HKFHUKLVK8XNJ0ZwvZkHg'
consumer_secret = 'QHZZJ8qA6j0oDwhXS1HBb1qWAKeN1pWkw88tVZxUo'
access_token = '63#336355-SEvxN7KC5xxqaEhKYSOgdHSQkKnHvz8PUSCii0kp'
access_token_secret = 'AUmF0JmpSGIOsbrqnwrxtwMopwE6vDhkdH8Dx00W'

# OAuth process, using the keys and tokens
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

# Creation of the actual interface, using authentication
api = tweepy.API(auth)

max_tweets=1000
query=sys.argv[1]

resultados = tweepy.Cursor(api.search,  q=query,lang = 'es').items(max_tweets)

#Creacion del fichero que contendra el corpus
File= open('corpus.txt','w')
#searched_tweets = [status.text for status in tweepy.Cursor(api.search,  q=query).
items(max_tweets)]
#json_strings = [json.dumps(json_obj) for json_obj in searched_tweets]
text=[]
for sts in resultados:
    #Process the status here
    File.write(str(sts.id))
    File.write(sts.source_url)
    File.write(sts.text.encode('utf8')+'\n')
File.close()
```

Referencias

[1] Pierre Baldi, Paolo Frasconi, and Padhraic Smyth. Modeling and understanding human behavior on the web. *Modeling the Internet and the Web: Probabilistic Methods and Algorithms*, pages 171–209, 2003.

[2] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[3] Wikipedia. K-vecinos más cercanos — wikipedia, la enciclopedia libre, 2015. [Internet; descargado 10-mayo-2016].

[4] Cristina García Cambronero and Irene Gómez Moreno. Algoritmos de aprendizaje: knn & kmeans. *Intelgencia en Redes de Comunicación, Universidad Carlos III de Madrid*, 2006.

[5] David Vilares, Miguel A Alonso, and Carlos Gómez-Rodríguez. Una aproximación supervisada para la minería de opiniones sobre tuits en español en base a conocimiento lingüístico. *Procesamiento del lenguaje natural*, 51:127–134, 2013.

[6] David Vilares, Miguel A Alonso, and Carlos Gómez-Rodríguez. Clasificación de polaridad en textos con opiniones en espanol mediante análisis sintáctico de dependencias. *Procesamiento del lenguaje natural*, 50:13–20, 2013.

[7] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

[8] Eugenio Martínez Cámara, María Teresa Martín Valdivia, José Manuel Perea Ortega,

and Luis Alfonso Ureña López. Técnicas de clasificación de opiniones aplicadas a un corpus en español. 2011.

[9] Bing Liu. Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2:627–666, 2010.

[10] Pablo Kogan and Sandra Roger. Análisis de opinión como un sistema multiagente distribuido. In *XIII Workshop de Investigadores en Ciencias de la Computación*, 2011.

[11] Aurélien Bossard, Michel Généreux, and Thierry Poibeau. Cbseas, a summarization system integration of opinion mining techniques to summarize blogs. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*, pages 5–8. Association for Computational Linguistics, 2009.

[12] Satoshi Morinaga, Kenji Yamanishi, Kenji Tateishi, and Toshikazu Fukushima. Mining product reputations on the web. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 341–349. ACM, 2002.

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

[14] Cuneyt Gurcan Akcora, Murat Ali Bayir, Murat Demirbas, and Hakan Ferhatosmanoglu. Identifying breakpoints in public opinion. In *Proceedings of the First Workshop on Social Media Analytics*, pages 62–66. ACM, 2010.

[15] Mário J Silva, Paula Carvalho, Luís Sarmento, E de Oliveira, and Pedro Magalhaes. The design of optimism, an opinion mining system for portuguese politics. *New trends in artificial intelligence: Proceedings of EPIA*, pages 12–15, 2009.

[16] George Stylios, Dimitris Christodoulakis, Jeries Besharat, M Vonitsanou, Ioanis Ko-

trotsos, Athanasia Koumpouri, and Sofia Stamou. Public opinion mining for governmental decisions. *Electronic Journal of e-Government*, 8(2):203–214, 2010.

[17] Guiomar Elena Ciapuscio. *Tipos textuales*. Universidad de Buenos Aires, Instituto de Lingüística, 1994.

[18] Gutiomar E Ciapuscio. Hacia una tipología del discurso especializado. *Revista iberoamericana de Discurso y Sociedad*, 2(2):39–71, 2000.

[19] Giovanni Parodi. Lingüística de corpus y análisis multidimensional: Exploración de la variación en el corpus pucv-2003. *Revista española de lingüística*, 35(1), 2005.

[20] Carlos G-Figuerola, Angel F Zazo, and José-Luis Alonso-Berrocal. Categorización automática de documentos en español: algunos resultados experimentales. 2000.

[21] Daniel Jurafsky and James H Martin. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition. 2000.

[22] Gerard Salton. Automatic information organization and retrieval. 1968.

[23] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

[24] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[25] Luis Carlos García Figuerola. La investigación sobre recuperación de la información en español. 2000.

[26] A Zazo, C Figuerola, JL Alonso, and R Gómez. Recuperación de información utilizando el modelo vectorial. *Recuperación de información utilizando el modelo vectorial*, 2002.

[27] U Cervino Beresi, JJ Garcıa Adeva, R Calvo, and A Ceccatto. Automatic classification of news articles in spanish. In *Actas del Congreso Argentino de Ciencias de Computación*, 2004.

[28] René Venegas. Clasificación de textos académicos en función de su contenido léxico-semántico. *Revista signos*, 40(63):239–271, 2007.

[29] RA Fairthorne. The mathematics of classification. *Towards information retrieval*, pages 1–10, 1961.

[30] Robert M Hayes. Mathematical models in information retrieval. *Natural Language and the Computer (Edited by PL Garvin), McGraw-Hill, New York*, 287, 1963.

[31] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.

[32] Thomas K Landauer. On the computational basis of learning and cognition: Arguments from lsa. *Psychology of learning and motivation*, 41:43–84, 2002.

[33] Donna Harman. Relevance feedback and other query modification techniques., 1992.

[34] Robert S Bader. Similarity and recency of common ancestry. *Systematic Biology*, 7(4):184–187, 1958.

[35] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[36] David J Rogers, Taffee T Tanimoto, et al. A computer program for classifying plants. *Science (Washington)*, 132:1115–18, 1960.

[37] Evelyne Tzoukermann, Judith L Klavans, and Tomek Strzalkowski. Information retrieval. *The Oxford handbook of computational linguistics*, pages 530–544, 2003.

[38] Dallas E Johnson. *Métodos multivariados aplicados al análisis de datos*. Number 311.2/J67aE. 2000.

[39] J Molina and J García. Técnicas de análisis de datos en aplicaciones prácticas utilizando microsoft excel y weka [en línea], 2004.

[40] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[41] Gustavo A Betancourt. Las máquinas de soporte vectorial (svms). *Scientia et Technica*, 1(27), 2005.

[42] Richard O Duda, Peter E Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[43] F Bordignon, J Peri, G Tolosa, D Villa, and L Paoletti. Experimentos en clasificación automática de noticias en español utilizando el modelo bayesiano [en línea], 2004.

[44] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[45] A Téllez. Extracción de información con algoritmos de clasificación. *Extracción de información con algoritmos de clasificación*, 2005.

[46] S Sharma. Applied multivariate techniques, new york, john willey & sons, 1996.

[47] Joseph F Hair and Mónica Gómez Suárez. *Análisis multivariante*, volume 491. Prentice Hall Madrid, 1999.

[48] Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[49] Ernesto Bautista-Thompson, E Guzmán-Ramírez, and Jesús Figueroa-Nazuno. Predicción de múltiples puntos de series de tiempo utilizando support vector machines. *Computación y sistemas*, 7(3):148–155, 2004.

[50] Eduardo Morales. Descubrimiento de conocimiento en bases de datos. *Obtenido el día*, 11, 2009.

[51] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Elsevier, 2011.

[52] Joan Torruella and Joaquim Llisterri. Diseño de corpus textuales y orales. *Filología e informática. Nuevas tecnologías en los estudios filológicos*, pages 45–77, 1999.

[53] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[54] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014.