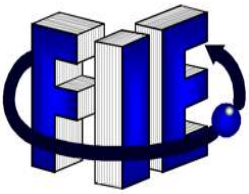


Universidad Michoacana de San Nicolás de Hidalgo

Facultad de Ingeniería Eléctrica

EVIDENCIA EMPÍRICA DEL DESEMPEÑO DE CLASIFICADORES DE TEXTO: TEXTOS DE LA RED SOCIAL TWITTER

TESIS

Que para obtener el grado de
INGENIERO EN COMPUTACIÓN

Presenta
José Juan Valdés Felipe

Asesor de Tesis
M.I. Moisés García Villanueva

Morelia, Michoacán, Julio 2017

Agradecimientos

Quiero agradecer a la Universidad Michoacana de San Nicolás de Hidalgo, a mi honorable Facultad de Ingeniería Eléctrica y a todos los profesores que a lo largo de mi camino por la carrera que me brindaron su apoyo y la oportunidad de integrar los conocimientos suficientes para poder llamarme ingeniero, a si mismo quiero mencionar en particular al M.I Moisés García Villanueva como profesor y asesor de tesis que me brindó todo el apoyo posible para culminar la elaboración de esta tesis.

Agradezco a mis padres por haberme dado la oportunidad de cumplir mis metas y haberme dado el legado más importante de la vida que es la educación por lo cual estaré eternamente agradecido.

A mis hermanos por estar conmigo en los momentos más difíciles y por haber brindado su comprensión cuando lo necesité.

También agradezco a todos los que fueron mis compañeros de clase durante todos los niveles de Universidad ya que gracias al compañerismo, amistad y apoyo moral han aportado en un alto porcentaje a mis ganas de seguir adelante en mi carrera profesional.

Finalmente quiero agradecer a Dios por darme salud, buenos amigos, y una gran familia a la cual quiero mucho, por darme la sabiduría suficiente y haberme guiado en todo este camino dándome la fuerza suficiente para lograrlo.

Dedicatoria

Esta tesis la dedico a Domingo Valdés García y Francisca Gonzalez Felipe que privilegio de tenerlos como padres que gran regalo crecer sin olvidar, porque creyeron en mi y porque me sacaron adelante, gracias a ustedes hoy puedo ver alcanzada mi meta, ya que siempre estuvieron impulsándome en los momentos más difíciles de mi carrera, y porque el orgullo que sienten por mi, fue lo que me hizo ir hasta el final. Va por ustedes, por lo que valen, porque admiro su fortaleza y por lo que han hecho de mí.

A mis hermanos José Jaime Valdés Felipe, María Guadalupe Valdés Gonzalez y Andres Valdés Gonzalez que siempre han estado presentes en todo este camino que he tenido que recorrer, siendo un gran apoyo para mi el saber que me apoyan incondicionalmente y dándome esos ánimos que necesité en todo momento.

Agradezco a mis profesores, personas de gran sabiduría quienes se han esforzado por ayudarme a llegar al punto en el que me encuentro, sencillo no ha sido el proceso, pero gracias a las ganas de transmitirme sus conocimientos y dedicación que los ha regido, en especial al profesor Moisés García Villanueva quien aún fuera de las aulas y horarios de clase seguía instruyéndome, he logrado importantes objetivos como culminar el desarrollo de mi tesis con éxito y obtener una afable titulación profesional.

Dedico esta tesis a Dios quien siempre me ha estado guiando por el mejor camino posible y cuidando en todo momento.

Resumen

Actualmente, las redes sociales se han convertido en una importante fuente de comunicación a nivel mundial. Dado que los servicios de las redes sociales se expanden geográficamente desde hace varios años, cada vez en mayor medida, y penetran cada vez más en diversos segmentos de la población, el valor de la información que se genera en este tipo de plataformas en línea aumenta drásticamente.

Hoy en día se hace necesario realizar estudios de cómo desarrollar e implementar sistemas de minería de opinión, además de documentar el efecto que tiene el preprocesamiento de los datos, los emoticonos al señalar un sentimiento en el texto y la selección de características; con el objeto de observar el desempeño de los algoritmos de clasificación de texto y llevar de forma más rápida su implementación.

Análisis de Sentimientos, dentro del área de estudio del Procesamiento de Lenguaje Natural, consiste en la identificación y extracción de opiniones emitidas en textos con el objetivo de clasificarlos, mediante procesamiento computacional, según la polaridad de las emociones que expresan sobre determinados objetos, situaciones o personas. Dicho de otra manera, es el estudio por el cual se determina la opinión de las personas en las redes sociales sobre algún tema en específico, prediciendo la polaridad de los usuarios (a favor, en contra, neutro, etc), abarcando temas que van desde productos, películas, servicios a intereses socioculturales como elecciones, guerras, fútbol, etc.

Para este proyecto de tesis se estableces los siguientes objetivos:

1. El objetivo principal de este trabajo es obtener una evidencia empírica del efecto que tienen factores como: el preprocesamiento de los datos, la selección de características y el uso de emoticonos en los algoritmos de clasificación de opinión.

2. Proporcionar información que permita identificar el mejor desempeño de un clasificador respecto del otro, de acuerdo al conjunto de datos utilizado.

3. Proporcionar un Sistema Clasificador de Tweets en la Web, que un desarrollador de sistemas pueda utilizar al momento de implementar o desarrollar un sistema de minería de opinión.

Palabras clave: análisis de sentimientos, minería de textos, selección de características, redes sociales, twitter, algoritmos de aprendizaje.

Abstract

Actually, social networks have become an important source of communication worldwide. As social network services have been geographically expanding for a number of years, and increasingly penetrating into various segments of the population, the value of the information generated on this type of online platform is drastically.

Nowadays it is necessary to carry out studies on how to develop and implement mining systems of opinion, additionally to documenting the effect of the pre-processing of the data, the emoticons by pointing out a feeling in the text and the selection of characteristics; in order to observe the performance of the text classification algorithms and carry out their implementation faster.

Analysis of sentiment, within the area of study of Natural Language Processing, is the identification and extraction of opinions issued in texts with the aim of classifying them, using computational processing, according to the polarity of the emotions they express about certain objects, situations or people. In other words, it is the study that determines the opinion of people in social networks on a specific topic, predicting the polarity of users (in favor, against, neutral, etc.), including topics ranging from products, films, services to sociocultural interests such as elections, wars, football, etc.

For this thesis project the following objectives are established:

1. The main objective of this work is to obtain an empirical evidence of the effect of such factors as: preprocessing of data, selection of features and use of emoticons in the opinion classification algorithms.

2. Provide information to identify the best performance of one classifier over the other, agree the data set used.

3. Provide a Tweets Classifier System on the Web, that a system developer can use when implementing or developing an opinion mining system.

Keywords: analysis of sentiment, text mining, selection of features, social networks, twitter, learning algorithms.

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 HiperTexto)
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)
KNN	**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)
MaxEnt	**M**aximum **E**ntropy **M**odeling (Modelo de Máxima Entropía)
ML	**M**aximum **L**ikelihood (Máxima Verosimilitud)
NB	**N**aive **B**ayes (Bayes Ingenuo)
NBM	**N**aive **B**ayes **M**ultinomial)
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)
SCTW	**S**istema **C**lasificador de **T**weets en la **W**eb
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

La minería textual es una de las tecnologías que, desde su formulación inicial a principios de la década de los noventa, ha tenido un gran impacto en las actividades relacionadas con la inteligencia militar. Si bien este impacto nunca ha alcanzado el nivel de generalización de la minería de datos, los desafortunados acontecimientos del 11 de septiembre de 2001 hicieron que distintos medios prestasen atención a las tecnologías empleadas por las organizaciones policiales encargadas de luchar contra el terrorismo. Así, a partir de esa fecha podemos encontrar un mayor número de referencias al uso de la minería textual y de datos con este propósito [Eíto Brun04].

Una aplicación de la lingüística computacional y del procesamiento de textos que pretende facilitar la identificación y extracción de nuevo conocimiento a partir de colecciones de documentos o corpus textuales, es por medio de la minería textual.

Minería de datos es el conjunto de técnicas y tecnologías que permiten explorar grandes bases de datos, de manera automática o semiautomática, con el objetivo de encontrar patrones repetitivos, tendencias o reglas que expliquen el comportamiento de los datos en un determinado contexto.

La diferencia entre minería de datos y minería textual es que en la última se pretende extraer conocimiento a partir de los patrones observables en grandes colecciones

de datos estructurados que se almacenan en bases de datos relacionales [Eíto Brun04].

Actualmente, las redes sociales se han convertido en una importante fuente de comunicación a nivel mundial. La gran cantidad de interacciones que se producen entre sus usuarios proporcionan una inmensa cantidad de información sobre sus preferencias, lo cual resulta verdaderamente útil para la detección de tendencias. Una de las redes sociales más relevantes y populares es Twitter, una red social de microblogging permite escribir mensajes de un máximo de 140 caracteres, de aquí el gran interés por estudiar los mensajes emitidos en esta red, por el desafío que involucra la clasificación automática de textos cortos [daCruz11], además de que se observan una gran cantidad de abreviaciones y se expresan mediante imágenes (emoticonos) algún sentimiento. Twitter proporciona una excelente fuente de información para hacer minería de opinión.

En la actualidad surge la necesidad de realizar estudios de cómo desarrollar e implementar sistemas de minería de opinión. En este trabajo minería de opinión a Twitter en determinar la opinión de los usuarios sobre una marca comercial, sobre el último producto presentado por una compañía, sobre la campaña electoral de un partido político, o estimar la intención de voto de un determinado partido político [Cámara11].

Además de documentar el efecto que tiene el preprocesamiento de los datos, los emoticonos al señalar un sentimiento en el texto y la selección de características, con la finalidad de observar el desempeño de los algoritmos de clasificación de texto y llevar de forma más rápida su implementación.

1.2. Antecedentes

La minería de opiniones *(Opinión Mining–OM)* o análisis de sentimientos está cobrando cada vez mayor importancia debido fundamentalmente a la gran cantidad de comentarios que se escriben en Internet por parte de millones de usuarios de todo el mundo a través de blogs, foros o redes sociales. En Internet proliferan las *webs* de opiniones, generalistas o específicas, sobre cualquier ítem como hoteles, productos electrónicos, coches, películas, temas políticos [Eugenio11].

Entre una de las múltiples tareas de las que se ocupa el Procesamiento del Lenguaje

Natural (PLN) se encuentra la clasificación de textos, que consiste en la asignación de un conjunto de categorías a una colección de documentos, resolviéndose de esta forma la clasificación objetiva de documentos.

Existe una gran cantidad de textos en el que el contenido subjetivo es lo mas relevante, y cuyo procesamiento no debería limitarse a aplicar únicamente las técnicas de la clasificación de documentos. Ante esta necesidad de clasificar la orientación, o la opinión que se expresan en los documentos, surge la área análisis de sentimientos (AS)[Cámara11].

El AS trata de clasificar los documentos en función de la polaridad de la opinión que expresa su autor.

En este trabajo la palabra *corpus* se denomina que es el conjunto de datos en un sentido general del término. El uso de corpus para el estudio de la lengua se considera una metodología empírica de trabajo, basada en el empleo de datos reales, de muestras de uso de la lengua [Llamazares08].

Existen muchos trabajos en el campo del análisis de sentimientos, habiéndose aplicado en multitud de dominios, pero la mayor parte de ellos han sido realizados sobre corpus de documentos en Inglés.

El trabajo se centra en obtener evidencia empírica de la aplicación de técnicas de AS en textos cuya información se encuentra en español y es obtenida de la red social Twitter. Además de implementar Sistema de Análisis de Opinión en la Web.

1.3. Objetivo

El objetivo principal de este trabajo es obtener una evidencia del efecto que tiene el preprocesamiento de los datos, la selección de características y el uso de emoticonos en los algoritmos de clasificación de texto, que se emplean en emitir la opinión de un documento. Proporcionar evidencia empírica que permita identificar las pruebas necesarias para decidir que Clasificador utilizar al momento de implementar o desarrollar un sistema de minería de opinión.

1.4. Justificación

Al ser un tema de investigación relativamente nuevo y por la gran cantidad de información que se genera en las redes sociales, se hace necesario documentar la problemática en la implementación de un sistema de minería de opinión. La complejidad del tema en textos cortos, crea un gran reto en la implementación de un sistema de MO en la red social Twitter en donde se interactúa con una colección de documentos.

Por último se señala la necesidad de incrementar los estudios de caso sobre el corpus en el idioma español ya que no hay un amplio trabajo como en el idioma ingles, por lo que es satisfactorio trabajar en el idioma español. Y la capacitación de profesionales en el área que impacten en las decisiones mediante el uso de información que no se ha descubierto y que no se encuentra transparente en las redes sociales.

1.4.1. Objetivos particulares

- Construcción de conjuntos de datos mediante API's establecidas para la prueba de algoritmos de clasificación de texto.
- Implementar diferentes tipos de preprocesamiento de datos en los conjuntos de datos construidos.
- Implementar el método de extracción de características χ^2 para identificar su impacto en los algoritmos de clasificación de texto en los conjuntos de datos construidos.
- Implementación y evaluación de algoritmos de clasificación de texto bajo los diferentes tipos de preprocesamiento de los datos.
- Realizar el análisis de los resultados y emitir las conclusiones.

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 principales y particulares de la tesis. Se señala además una descripción de cada capítulo.

- En el capítulo 2 se aborda el marco referencial donde se describe brevemente el concepto de minería de opinión, el modelo vectorial, las técnicas que se utilizan para la clasificación de texto y el método de selección de características.

- En el capitulo 3 se describe la implementación del sistema de Análisis de Opinión en la Web, describiendo un mini tutorial para utilizarla de forma correcta.

- En el capítulo 4 se realiza la parte experimental, sobre los algoritmos de clasificación y los que se les ha aplicado distintos tipos de preprocesado, el análisis y la interpretación de los resultados.

- En el capítulo 5 se presentan las conclusiones de los resultados obtenidos, así como aportaciones y trabajos futuros.

Capítulo 2

Marco Teórico

2.1. Definición de minería de texto

En primer lugar se delimita el alcance del término *"minería textual"* o *"minería de textos"*, que se utilizarán como sinónimos en esta Tesis.

La minería textual debe facilitar el análisis de corpus textual que a priori resultarían inmanejables debido a su tamaño. Tendrá como objetivo intermedio —y previo al descubrimiento de nuevo conocimiento— procesar y presentar la información disponible en grandes colecciones de documentos en un formato que facilite su comprensión y análisis[Eíto Brun04].

2.1.1. Minería de Opinión

En la actualidad la cantidad de datos producidos a nivel mundial es muy alta. Y toda esta información es atractiva para diferentes comerciales, industriales y académicos. La extracción de datos y su respectivo procesamiento, hace que estas tareas sean muy complejas y difíciles de hacer de forma manual.

La minería de opinión (MO) busca analizar las opiniones, sentimientos, valoraciones, actitudes y emociones de las personas hacia entidades como productos, servicios, organizaciones, individuos, problemas, sucesos, temas y sus atributos [Liu12]. Las empresas y organizaciones en general están interesadas en conocer cuál es la reputación que tienen

de sus usuarios en las redes sociales, blogs, wikis y otros sitios web [Vilares13].

2.2. El modelo vectorial

El modelo vectorial representa documento textual a través de vector de *término*, se considera cada documento como una colección de palabras (bolsa de palabras).

En este apartado se presenta 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 palabras es el elemento principal, y a partir de ellas clasificar documentos nuevos (tales como los textos de las redes sociales) en clases pre-existentes (disciplinas, opiniones o sentimiento).

Según Zazo et. al. [Zazo02], 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 (c_m). 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 [Venegas07].

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

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.) [Landauer02].

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, se debe 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 [Beresi04]. 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. El caso más simple es utilizar una aproximación binaria, de forma que si en el documento D_i aparece el término r, el valor c_{ir} sería 1, y en caso contrario sería 0.

Ahora bien, 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 [Salton86, Salton88, Harman92], pero en general, para estimarlos se parte de dos ideas en cierto sentido contrapuestas:

1. Si un término se consigna u ocurre mucho en un documento, es importante para

caracterizar el documento.

2. Pero si dicho término aparece en muchos documentos de la colección, este término no resulta beneficioso para distinguir un documento de los demás.

Se dice que cuando un término cae en la segunda idea, tiene un 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. Esto es lo que se conoce como 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 [Harman92], esto se define en la ecuación 2.2.

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

Al respecto, Salton y Buckley ([Salton88]) experimentaron con más de 200 sistemas de cálculo de pesos. Uno de los más utilizados esta representado 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. 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 [Manning99].

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, véase la ecuación 2.4, en donde Q es un documento de consulta que está representado por un vector de pesos $\vec{q}$. Así entonces, para una consulta $Q \rightarrow \vec{q}$, el índice de similitud con un documento D_i está dado por la ecuación (2.4).

$$simil(Q, D_i) = \frac{\sum_{j=1}^{m} \vec{q_j} \vec{d_{ij}}}{\sum_{j=1}^{m} \vec{q_j}^2 \cdot \sum_{j=1}^{m} \vec{d_{ij}}^2} \tag{2.4}$$

Otro índice de similitud que también es ampliamente utilizado por la comunidad de sistemas en recuperación de información es el coseno del ángulo formado por ambos vectores, que geométricamente indica que aquellos vectores que son cercanos al paralelismo entre ellos (ángulo cercano a 0 grados) son los más similares, mientras que si son perpendiculares (ángulos próximos a los 90 grados) son totalmente diferentes, este índice de similitud está dado por la ecuación 2.5.

$$\cos\theta = \frac{\sum_{j=1}^{m} \vec{q_j} \cdot \vec{d_{ij}}}{\sqrt{\sum_{j=1}^{m} \vec{q_i}^2} \cdot \sqrt{\sum_{j=1}^{m} \vec{d_{ij}}^2}} \tag{2.5}$$

Cabe señalar, que existen otros métodos propuestos para calcular la similitud, el inmediato es la distancia Euclidiana y algunos otros ejemplos son: el coeficiente de emparejamiento (*matching coefficient*) [Bader58], el coeficiente de Dice [Dice45], el coeficiente de Jaccard (o Tanimoto) [Rogers60] y el coeficiente de solapamiento (*overlap coefficient*). Una síntesis con la descripción de estas medidas de similitud puede encontrarse en Jurafsky y Martin [Jurafsky00], Manning y Schütze [Manning99], Tzoukermann, Klavans y Strzalkowski [Tzoukermann03].

2.2.1. Ejemplo de la representación vectorial

En esta subsección se presenta un ejemplo del modelo vectorial, que es el más utilizado en los sistemas de recuperación de información.

Este modelo trabaja los documentos expresados en función de 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 (el texto de cada uno de los Tweets), en donde las columnas representan los términos ó (palabras diferentes en el corpus) y cada renglón representa un documento del 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.

El conjunto de datos de texto contiene los siguientes documentos:

D_1*: el río Danubio pasa por Viena, su color es azul.*

D_2*: el caudal de un río asciende en Invierno.*

D_3*: el río Rhin y el río Danubio tienen mucho caudal.*

D_4*: si un río es navegable, es porque tiene mucho caudal.*

Su matriz correspondiente dentro del modelo ó representación vectorial, eliminando algunas palabras vacías *el, pasa, por, su, es, en, tienen, un*), estará representada con los valores de frecuencia de ocurrencia de las palabras, como se muestra en la tabla 2.1.

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

	río	**danubio**	**viena**	**color**	**azul**	**caudal**	**invierno**	**rhin**	**navegable**
D_1	1	1	1	1	1	0	0	0	0
D_2	1	0	0	0	0	1	1	0	0
D_3	2	1	0	0	0	1	0	1	0
D_4	1	0	0	0	0	1	0	0	1

En cuanto a las palabras vacías, se eliminan 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 *tf-idf*, 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**)

Los componentes del 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 se le asignan automáticamente 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 de 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 la combinación de la frecuencia de término (tf), y la frecuencia inversa del documento (idf)***. Para calcular el valor de la $j - ésima$ entrada del vector que corresponde al documento i, de la cuál se emplea la ecuación (2.2) en la que se utilizo 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 del ejemplo, en la tabla (2.2), se observan los resultados del cálculo de pesos de matriz de términos y documentos en el espacio vectorial.

Cálculo de frecuencias inversas:

$idf(rio) = lg(4/4) = lg(1) = 0$

$idf(danubio) = lg(4/2) = lg(2) = 0.301$

$idf(viena) = lg(4/1) = lg(4) = 0.602$

$idf(color) = lg(4/1) = lg(4) = 0.602$

$idf(azul) = lg(4/1) = lg(4) = 0.602$

$idf(caudal) = lg(4/3) = lg(1.33) = 0.124$

$idf(invierno) = lg(4/1) = lg(4) = 0.602$

$idf(rhin) = lg(4/1) = lg(4) = 0.602$

$idf(navegable) = lg(4/1) = lg(4) = 0.602$

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

	río	**danubio**	**viena**	**color**	**azul**	**caudal**	**invierno**	**rhin**	**navegable**
D_1	0	0.301	0.602	0.602	0.602	0	0	0	0
D_2	0	0	0	0	0	0.124	0.602	0	0
D_3	0	0.301	0	0	0	0.124	0	0.602	0
D_4	0	0	0	0	0	0.124	0	0	0.602

Una vez que se cuenta con la matriz de pesos (representación vectorial de los datos), para un nuevo documento Q (documento de consulta), se requiere que ese texto sea equiparable con los documentos D de la colección, respecto a la consulta Q del usuario y se procede a calcular la similaridad de Q con cada uno de los documentos en D, ello se realiza mediante el producto escalar (que se muestra en la ecuación (2.4)). De esta forma,

la similaridad de un documento y una consulta, es igual a la suma de los productos de sus pesos, donde cada peso representa a un término. En La tabla 2.3 se muestra este proceso.

Tabla 2.3: Producto escalar de pesos tf-idf.

	río	**danubio**	**viena**	**color**	**azul**	**caudal**	**invierno**	**rhin**	**navegable**
D_1	0	0.301	0.602	0.602	0.602	0	0	0	0
D_2	0	0	0	0	0	0.124	0.602	0	0
D_3	0	0.301	0	0	0	0.124	0	0.602	0
D_4	0	0	0	0	0	0.124	0	0	0.602
Q	0	0.301	0	0	0	0.124	0	0	0

El cálculo de la similaridad se aplica para 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, D_3 y D_4. 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.

Posteriormente se calculan las similitudes existentes entre los distintos documentos (D_1, D_2, D_3 y D_4) y el vector Q. 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.

A continuación se muestra el cálculo de similitudes:

simil(Q, D_1) = 0*0 + 0.301*0.301 + 0*0.602 + 0*0.602 + 0*0.602 + 0.124*0 + 0*0 + 0*0 + 0*0 = **0.09**

simil(Q, D_2) = 0*0 + 0.301*0 + 0*0 + 0*0 + 0*0 + 0.124*0.124 + 0*0.602 + 0*0 + 0*0 = **0.01**

simil(Q, D_3) = 0*0 + 0.301*0.301 + 0*0 + 0*0 + 0*0 + 0.124*0.124 + 0*0 + 0*0.602 + 0*0 = **0.10**

simil(Q, D_4) = 0*0 + 0.301*0 + 0*0 + 0*0 + 0*0 + 0.124*0.124 + 0*0 + 0*0 + 0*0.602 = **0.01**

Con estos valores de similitud, se obtiene la respuesta: D_3, D_2, D_1, D_4. En este modelo se observa un ejemplo de acierto y un ejemplo de fallo, el primero de los documentos recuperados sí responde a la pregunta D_3 y al mismo tiempo los demás no responden adecuadamente (la similitud es muy baja). Para éste ejemplo, justifican la presencia de

documentos no relevantes en la respuesta de los sistemas de recuperación de información, y se observa que este esquema básico de alineamiento ha sufrido muchos cambios. Por lo tanto el modelo vectorial necesita de la intersección de los términos de la consulta con los documentos, en caso contrario no se produce la recuperación de información. Pero la ventaja de utilizar este método tiene en cuenta los pesos tf-idf para determinar la representatividad de los documentos de la colección.

2.3. Algoritmos de clasificación

Existe una gran variedad de técnicas de clasificación de documentos en aprendizaje de máquina (ML, por sus siglas en Inglés de Machine Learning). En esta sección se describen cuatro métodos de clasificación de texto, basados en el modelo vectorial: Naive Bayes Multinomial (*NBM*), K Vecinos más Cercanos (*K-nearest neighbors* —KNN), Máquina de Vectores de Soporte (*Support Vector Machine* —SVM) y Modelo de Máxima Entropía (*Maximum Entropy Modeling* —MaxEnt). Se tiene como finalidad documentar evidencia del desempeño de dichos Clasificadores en la tarea de clasificación de textos cortos con datos coleccionados de la red social Twitter, en el idioma español y bajo diferentes tipos de preprocesamiento de los datos. Estos métodos, si bien son comunes en la clasificación de documentos, aún no han sido suficientemente probados en corpus textuales cortos y en español [Carlos00, Beresi04].

A continuación se explica brevemente las características principales de los métodos de clasificación NBM, KNN, SVM y MaxEnt. 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 estos métodos.

2.3.1. Clasificador Naive Bayes Multinomial

Los Clasificadores bayesianos [Duda73] 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 veloci-

dad cuando se han aplicado a grandes bases de datos textuales, particularmente en español [Molina04, Beresi04, Bordignon04].

Naive Bayes es uno de los modelos probabilistas mas simples y más usados en clasificación de texto porque produce resultados tan buenos como otros modelos más sofisticados.

En este trabajo se considerara un Clasificador de la familia Naive Bayes: Bayesiano Multinomial.

- **Clasificador Bayesiano Multinomial:** Considera el número de apariciones de cada término para evaluar la contribución de su probabilidad condicional dada la clase del documento.

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 se denota $P(D)$ como la probabilidad a *priori* de los datos y $P(D|h)$ como la probabilidad de los datos dada una hipótesis, se quiere estimar $P(h|D)$, 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 particular, el Clasificador bayesiano ingenuo se utiliza cuando se quiere clasificar un ejemplo descrito por un conjunto de atributos, características 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 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 [Jurafsky00, Molina04, Bordignon04].

2.3.2. Clasificador de los K vecinos más cercanos (KNN)

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 asignarle una categoría a un nuevo objeto. Los vecinos más cercanos a una instancia se obtienen, para el

caso de los atributos continuos, por lo general utilizando la distancia Euclidiana sobre los m posibles atributos. El resultado de la clasificación por medio de este algoritmo puede ser discreto o continuo. En ambos casos el resultado de la clasificación es la clase más común o con mayoría de los k vecinos [Morales09, Han11].

Los ejemplos de entrenamiento n son vectores en un espacio característico multi-dimensional, cada ejemplo está descrito en términos de m 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 *m-dimensional*.

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

- **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 se elige $k = 1$; 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 valor de k puede ser seleccionado mediante algún algoritmo de 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 KNN. 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 se puede ponderar el voto de cada vecino de acuerdo al cuadrado inverso de sus distancias, como se observa en 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

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

  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 al método base es muy efectiva en muchos problemas prácticos. Es robusto ante los ruidos de datos y suficientemente efectivo en conjuntos de datos grandes. Se

observa que al tomar promedios ponderados de los k vecinos más cercanos el algoritmo evita el impacto de ejemplos con ruido aislados.

- **Ejemplo del algoritmo** KNN

  En la figura 2.1 se muestra un círculo encerrando a una x, el cual corresponde al ejemplo u objeto que se desea clasificar, se representan 12 muestras pertenecientes a dos clases distintas: la Clase 1 está formada por 6 cuadrados y la Clase 2 formada por 6 círculos. En este ejemplo, se han seleccionado tres vecinos, es decir, (k=3), los cuales están encerrados junto al ejemplo a clasificar por una circunferencia. Al contar cuantos elementos de cada clase existen, tenemos que de la Clase 1 existe un elemento y de la Clase 2 existen 2 elementos; para decidir a que clase pertenece el ejemplo a clasificar se toma aquella que contiene mayor número de elementos, en el ejemplo corresponde a la Clase 2 que contiene 2 elementos.

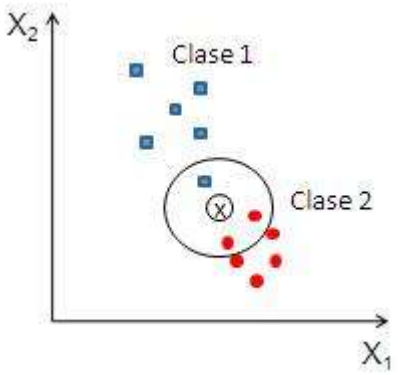


Figura 2.1: Ejemplo del algoritmo KNN, de los 3 vecinos más cercanos a la muestra x.

Por tanto, la regla 3-NN asignará la muestra x a la Clase 2. Es importante señalar que si se hubiese utilizado como regla de clasificación la 1-NN, la muestra x sería asignada a la Clase 1, pues el vecino más cercano de la muestra x pertenece a la Clase 1.

2.3.3. 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 novedoso, con el cual se ha conseguido

buen desempeño de generalización sobre una amplia variedad de problemas de clasificación, destacando recientemente en problemas de clasificación de textos. El método tiene la capacidad de minimizar el error de generalización, es decir, los errores del Clasificador sobre nuevos documentos [Cortes95, Hsu03, Baldi03, Téllez05].

Particularmente 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 [Cortes95].

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 SVM, es producir un modelo que permita predecir los valores de clasificación (identificar la clase) en la etapa de prueba conociendo solo los atributos [Baldi03].

En términos geométricos, el problema que resuelve SVM es identificar una frontera de decisión lineal entre dos grupos, a través de una línea que los separe, cuyo objetivo es maximizar el espacio a los elementos más cercanos de dicha línea, de modo muy similar a lo que se realiza utilizando el análisis discriminante [Sharma96, Hair99]. Sin embargo, SVM incluye una operación nueva llamada *cambio 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 2.2(a), observamos un segmento de línea que sirve de separación en un espacio bidimensional de un conjunto de ejemplos separables en dos clases, denominado *hiperplano de separación*. En la figura 2.2(b), se observa que un espacio multidimensional cuenta con una infinidad de posibilidades de separación de las clases por medio de múltiples hiperplanos. Lo que el procedimiento de SVM hace, es buscar el mejor hiperplano que maximice la distancia a los vectores más próximo al hiperplano.

Para desarrollar el concepto de SVM, se hace el planteamiento de clasificación binaria para ejemplos perfectamente separables mediante lo que se conoce en SVM como

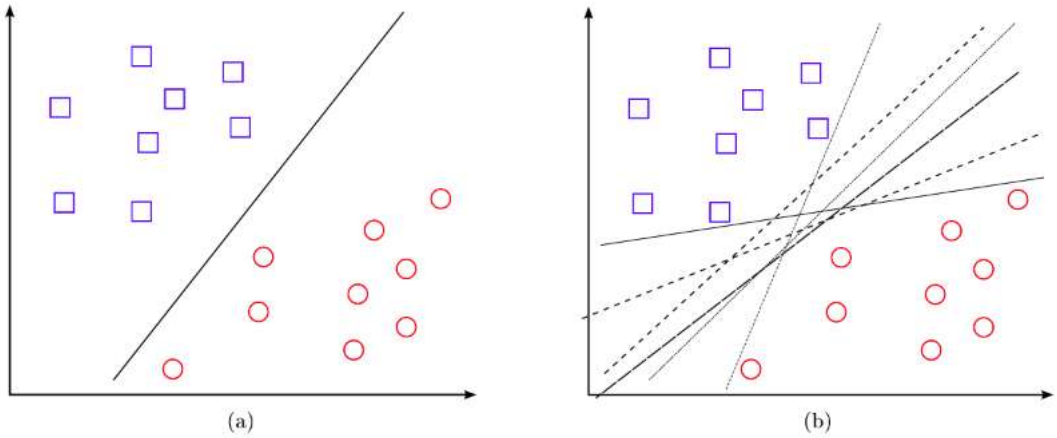


Figura 2.2: Hiperplanos de separación en un espacio bidimensional de un conjunto de ejemplos separables en dos clases: (a) ejemplos de hiperplano de separación (b) otros ejemplos de separación, de entre los infinitos posibles. [Suárez14].

"margen duro".

A continuación se describe el procedimiento de hiperplanos de separación en un espacio bidimensinal:

Dado un conjunto separable de ejemplos $S = (x_1, y_1), ..., (x_n, y_n)$ donde $x_i \in \mathbb{R}^d$ e $y_i \in +1, -1$ que corresponden a las etiquetas de cada clase, se puede definir un hiperplano de separación como el que se muestra en la figura 2.2(a). El hiperplano se puede representar como una función lineal que es capaz de separar S sin error, como se observa en la ecuación 2.17.

$$D(x) = (w_1 x_1 + ... + w_d x_d) + b = \langle w, x \rangle + b \tag{2.17}$$

donde w y b son coeficientes reales.

El hiperplano de separación cumplirá las siguientes restricciones para todo x_i del conjunto de ejemplos como se expresa en la ecuación 2.18.

$$x_i = \begin{cases} \langle w, x_i \rangle + b \geq 0 & \text{si } y_i = +1 \\ \langle w, x_i \rangle + b \leq 0 & \text{si } y_i = -1, i = 1, ..., n \end{cases} \tag{2.18}$$

o de forma mas compacta en la ecuación 2.19

$$y_i D(x_i) \geq 0 \qquad i = 1, ..., n \tag{2.19}$$

Para establecer si es posible obtener algún criterio adicional que permita definir un hiperplano de separación óptimo, primero se define el concepto *margen* de un hiperplano de separación, denotado por τ como la mínima distancia entre dicho hiperplano y el ejemplo más cercano de cualquiera de las dos clases, como se ve en la figura 2.3(a). A partir de esta definición, un hiperplano de separación se denominará *óptimo* si su margen es el tamaño máximo posible como se muestra en la figura 2.3(b).

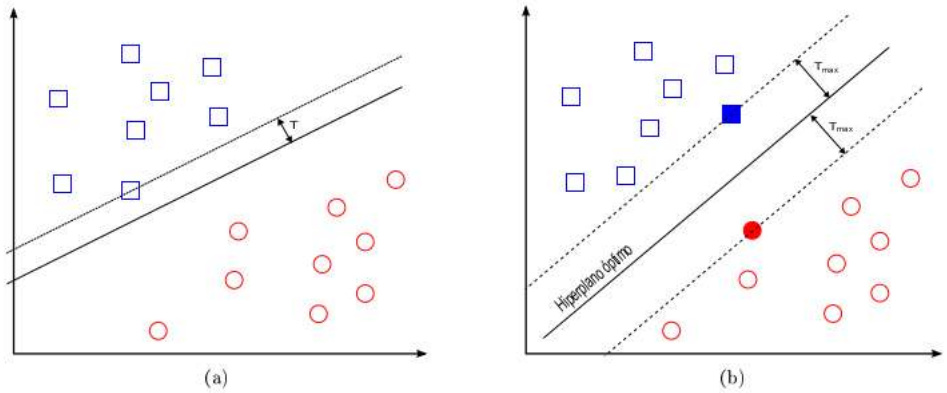


Figura 2.3: Margen de un hiperplano de separación (a) hiperplano de separación no-óptimo y su margen asociado (no máximo) (b) hiperplano de separación de óptimo y su margen asociado (máximo). [Suárez14].

Una propiedad inmediata de la definición de hiperplano de separación óptimo es que éste equidista del ejemplo más cercano de la clase. Supóngase que la distancia del hiperplano óptimo al ejemplo más cercano de la clase +1 fuese menor que la correspondencia al ejemplo más cercano de la clase -1. Esto significará que se puede alejar el hiperplano del ejemplo de la clase +1 una distancia tal que la distancia del hiperplano a dicho ejemplo se mayor que antes y a su vez, siga siendo menor que la distancia al ejemplo más cercano de la clase -1.

Por geometría se sabe que la distancia entre un hiperplano de separación $D(x)$ y un ejemplo x' viene dada, con la ecuación 2.20.

$$\frac{|D(x')|}{||w||} \tag{2.20}$$

siendo $|\cdot|$ el operador de valor absoluto, $||\cdot||$ el operador norma de un vector y w el vector que junto con el parámetro b, que define el hiperplano $D(x)$. $D(x)$ tiene la propiedad de ser perpendicular al hiperplano considerado. Haciendo uso de las expresiones 2.20 y 2.21, todos los ejemplos de entrenamiento se cumplirán.

$$y_i D(x_i) \geq 0 \qquad i = 1, ..., n \tag{2.21}$$

$$\frac{y_i D(x_i)}{||w||} \geq \tau, \qquad i = 1, ..., n \tag{2.22}$$

De la expresió 2.22, se deduce que el hiperplano óptimo es equivalente a encontrar el valor de w. Así por ejemplo, todas las funciones lineales $\lambda(\langle w, x\rangle + b)$, con $\lambda \in \mathbb{R}$ representan el mismo hiperplano. Para limitar el número de soluciones a una sola, la ecuación 2.22 se puede expresar también como la ecuación 2.23.

$$y_i D(x_i) \geq \tau||w||, \qquad i = 1, ..., n \tag{2.23}$$

la escala del producto de τ y la norma de w se fija, de forma arbitraria en la unidad, como se ve en la ecuación 2.24.

$$\tau||w|| = 1 \tag{2.24}$$

De 2.24 se concluye que aumentar el margen es equivalente a disminuir la norma de w, como lo expresa la ecuación 2.25.

$$\tau = \frac{1}{||w||} \tag{2.25}$$

De acuerdo a la definición un hiperplano de separación óptimo, como se muestra en la figura 2.4 será aquel que posee un margen máximo y por tanto un valor mínimo de $||w||$ y además esta sujeto a la restricción dada por la ecuación 2.23 y la ecuación 2.24, se obtiene la ecuación 2.26.

$$y_i D(x_i) \geq 1, \qquad i = 1, ..., n \tag{2.26}$$

o lo que es lo mismo en la ecuación 2.27

$$y_i D(\langle w, x_i \rangle + b) \geq 1, \qquad i = 1, ..., n \tag{2.27}$$

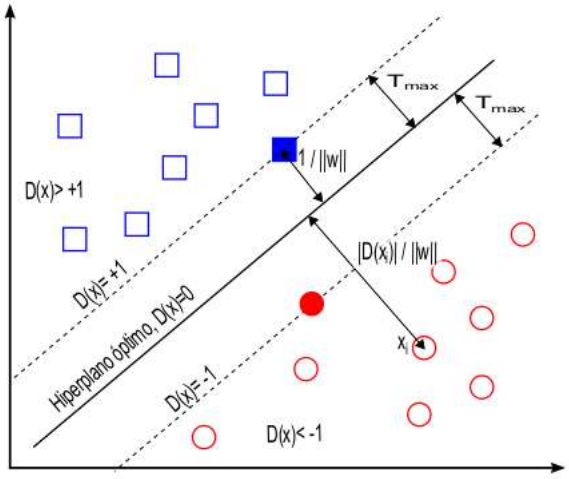


Figura 2.4: La distancia de cualquier ejemplo, x_i, al hiperplano de separación óptimo viene dada por $\frac{D(x_i)}{||w||}$. En particular, si dicho ejemplo pertenece al conjunto de vectores soporte (identificados pos siluetas sólidas), la distancia a dicho hiperplano será siempre $\frac{1}{||w||}$. Ademas, los vectores soporte aplicados a la función de decisión siempre cumplen que $|D(x)| = 1$. [Suárez14].

El concepto de margen máximo está relacionado directamente con la capacidad de generalización del hiperplano de separación, de tal forma que a mayor margen mayor distancia de separación existirá entre las dos clases. Los ejemplos que están situados a ambos lados del hiperplano óptimo y que definen el margen (o lo que es lo mismo aquellos para los que la restricción 2.27 es una igualdad) reciben el nombre de *vectores de soporte* y se observan también en la figura 2.4. Puesto que estos ejemplos son los más cercanos al hiperplano de separación serán los más difíciles de clasificar y por tanto deberían ser los únicos ejemplos a considerar a la hora de construir dicho hiperplano [Suárez14].

2.3.4. Modelo de Máxima Entropía (MaxEnt)

El modelo de Máxima Entropía, conocido también como *Multinomial Logistic Regression*, es un método de clasificación discriminativo y de aprendizaje supervisado donde los documentos del conjunto de datos son descritos a partir de una lista de características, siendo cada uno de estas características una restricción del modelo. Este método se basa en seleccionar la distribución de probabilidad que satisfaga todas las restricciones del modelo y maximice la entropía [Manning99].

En la clasificación por MaxEnt, la probabilidad que un texto corresponda a una determinada categoría debe maximizar la entropía de clasificación, de manera que no se introduzca un sesgo en el sistema. A diferencia de NB, este método no asume independencia entre las características o términos [Mehra02].

El método consiste en lo siguiente:

Cada característica f_i es una función binaria que puede ser utilizada para caracterizar una propiedad del par $(\vec{x}_j, c)$ donde $\vec{x}$ es un vector que representa un documento y c, la clase:

$$f_i(\vec{x}_j, c) = \begin{cases} 1 \ \textit{si la característica } \boldsymbol{i} \textit{ esta presente en el documento } \boldsymbol{j} \textit{ y } \boldsymbol{j} \textit{ pertenece a la clase } c \\ 0 \quad \textit{en otro caso} \end{cases} \tag{2.28}$$

Log-Linear Classifier

MaxEnt pertenece a la familia de Clasificadores conocidos como exponenciales o *log-linear* y se basa en extraer características de las observaciones y combinarlos linealmente ponderados con un determinado peso como se indica en la ecuación (2.29).

$$P(c|x) = \frac{1}{Z} exp(\sum_{i}^{\infty} w_i f_i) \tag{2.29}$$

Siendo w_i el peso de la característica f_i y Z un factor de normalización para que las probabilidades sumen 1.

Linear Regression

Los Clasificadores *log-linear* utilizan el concepto de *Linear Regression* (Regresión Lineal) para la combinación de características. Este método tiene el objetivo de predecir un valor, perteneciente a los números reales, a partir de la combinación lineal de los valores que toman determinados atributos en cada observación del modelo.

En el caso de la regresión lineal, la función $y = m * x + b$ se usa para describir la combinación lineal. Se debe elegir "b'' y "m'' de modo que entre todas las posibles rectas sean las mejores para el modelo.

Generalizando, la ecuación del método de *linear regression*:

$$y = \sum_{i=0}^{N} w_i \times f_i \tag{2.30}$$

Logistic Regression

Hasta ahora se ha tratado el método de regresión lineal donde el valor a predecir pertenecía a los números reales. Sin embargo, en la tareas de clasificación de textos, la predicción toma un valor discreto, por ejemplo para indicar que el texto pertenece a determinada clase. Para este caso la variable y tomará el valor 1 (verdadero) o 0 (falso) y este valor está dado con una probabilidad de acierto.

Por lo tanto, discretizando la ecuación de regresión lineal, se obtiene:

$$P(y = true|x) = \sum_{i=0}^{N} w_i \times f_i \tag{2.31}$$

Sin embargo, lo anterior no es un modelo probabilístico válido dado que no se asegura que la probabilidad sea un valor entre 0 y 1. Para resolver esto se utiliza lo que se conoce como ***logit-function***:

$$logit(p(x)) = ln\left(\frac{p(x)}{1 - p(x)}\right) \tag{2.32}$$

Aplicando la *función logit* al modelo lineal anterior, se obtiene:

$$ln\left(\frac{p(y=true|x)}{1-p(y=true|x)}\right) = w \cdot f_i \tag{2.33}$$

Escrito de otra forma:

$$p(y=true|x) = \frac{exp(\sum_{i=0}^{N} w_i f_i)}{1+exp(\sum_{i=0}^{N} w_i f_i)} \tag{2.34}$$

El modelo que utiliza una función lineal para estimar la probabilidad utilizando la función *logit* se conoce cómo **logistic regression** [Jurafsky00].

Generalizando el modelo de *logistic regression* se obtiene la ecuación (2.35) para modelos de máxima entropía:

$$P(c|x) = \frac{exp(\sum_{i=0}^{N} w_{ci} f_i)}{\sum_{c'} \in C^{exp(\sum_{i=0}^{N} w_{c'i} f_i)}} \tag{2.35}$$

Siendo el denominador el *factor de normalización Z* de la ecuación (2.29).

Será tarea del Clasificador seleccionar la clase que maximice la probabilidad anterior.

$$C_{MAXENT} = \frac{exp(\sum_{i=0}^{N} w_{ci} f_i)}{\sum_{c'} \in C^{exp(\sum_{i=0}^{N} w_{c'i} f_i)}} \tag{2.36}$$

2.4. Selección de características

La selección de características, también denominados atributos, componentes, variables, columnas, coordenadas o dimensiones, es un término usado habitualmente en minería de datos para describir las herramientas y las técnicas disponibles para reducir las entradas de los datos a un tamaño apropiado para su procesamiento y análisis [Pereira González15].

Para ello es necesario formular algunas preguntas sobre la selección de características.

- **¿Qué es selección de características?** Es el proceso de selección de un subconjunto de características relevantes para su uso en la construcción de modelos. La selección de características también se denomina selección de variables o selección de atributos.

- **¿Qué características debe utilizar para crear un modelo predictivo?** Es posible seleccionar automáticamente las características de los datos que sean más útiles o más relevantes para el problema en el que está trabajando. Este es un proceso llamado selección de características.

- **¿Para qué sirve la selección de características en los Clasificadores de texto?** Los métodos de selección de características le ayudan a los Clasificadores de textoa crear un modelo predictivo preciso con menos datos.

  Los métodos de selección de características pueden usarse para identificar y eliminar atributos innecesarios, irrelevantes y redundantes a partir de datos que no contribuyen a la precisión de un modelo predictivo, pueden disminuir la precisión del modelo.

- **¿Qué algoritmo de Selección de características usar?** Se eligió el **Método Filtrado** de entre otro, los métodos de selección de la característica de filtro aplican una medida estadística para asignar una puntuación a cada característica. Las características se clasifican por la puntuación y se seleccionan para mantenerse o eliminarse del conjunto de datos. Los métodos son a menudo univariados y consideran la característica independientemente, o con respecto a la variable dependiente.

  Para este trabajo se utiliza el método la prueba estadística χ^2 Chi al cuadrado.

2.4.1. Método de selección de características χ^2 (Chi cuadrado)

La prueba χ^2 se utiliza en la estadística, entre otras cosas, para probar la independencia de dos eventos, donde dos eventos A y B se definen como independientes si $P(AB) = P(A)P(B)$ o lo que es lo mismo, $P(A|B) = P(A)$ y $P(B|A) = P(B)$. En la selección características, los dos eventos son de un término t y una clase c, se usa para probar si la ocurrencia de un término específico y la ocurrencia de una clase específica son independientes. Así, por medio de la ecuación 2.37 se estima la relación t/c para cada término y se clasifican por su puntuación.

$$\chi^2(t|c) = \frac{N(AD - CB)^2}{(A + C)(B + D)(A + B)(C + D)} \tag{2.37}$$

Donde;

- A es el número de veces que t y c ocurren.
- B es el número de veces que t ocurre con la salida c.
- C es el número de veces que c ocurre sin t.
- D es el número de veces cuando ni c ni t se produce.
- N es el número total de documentos.

Capítulo 3

Sistema Clasificador de Tweets en la Web

En este capítulo se describe la interfaz de un sistema mínimo de Clasificación de Tweets en la WEB, que se desarrolló en esta tesis, al cual se le han integrado los diferentes Clasificadores que fueron previamente entrenados para clasificar texto en español (opinión o polaridad del texto en dos clases: POSITIVA, NEGATIVA).

3.1. Diagrama de flujo de procesamiento de predicción de Tweets

La figura 3.1 ilustra el diagrama de flujo que siguen los datos que se utilizan en el Sistema Clasificador de Tweets en la Web (SCTW) implementado. Primero el texto de entrada (tweet) pasa a la etapa de preprocesado (se describe en la sección 4.2), se eliminan los signos de puntuación y se convierten a minúsculas cada uno de los caracteres del texto. Posteriormente se pasa el texto resultante a la etapa del Clasificador (algoritmos descritos en la sección 2.3), el cual previamente fue entrenado y recibe el modelo que contiene lo aprendido en el entrenamiento (en el apéndice A se explica a detalle 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

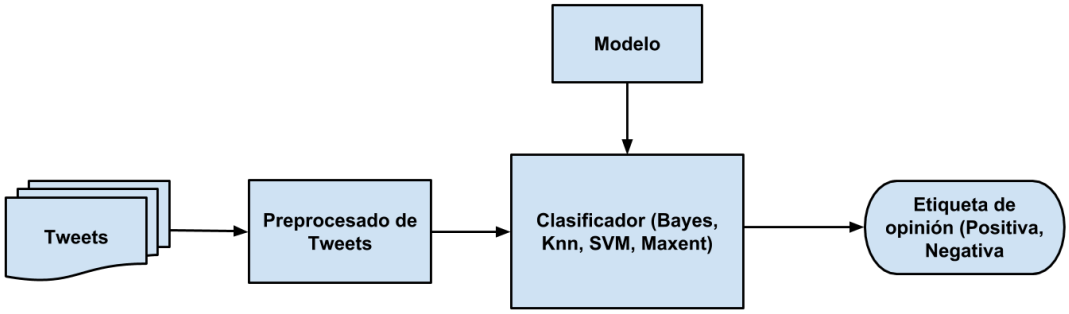


Figura 3.1: Diagrama de flujo del sistema de opinión o predicción de polaridad.

positiva para el texto que proporcionó el usuario; y la etiqueta *neg* que indica la predicción que corresponde a la polaridad negativa para el texto analizado.

3.2. Diseño de interfaz Web del sistema de opinión.

El sistema de predicción de texto se implementó en los lenguajes de programación PHP y Python, donde la implementación funcional principal se encuentra escrito en Python, mientras que para la interacción de la página Web con los sistemas de Clasificación se utilizó PHP. Para describir la apariencia o presentación de la página Web se utilizó el lenguaje de etiquetas HTML (*HyperText Markup Language*) y a las hojas de estilo CSS (*Cascading Style Sheets*). Los códigos fuente para la interconexión e implementación de la interfaz Web se presentan en el apéndice B.

3.3. Funcionamiento del Sistema

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

3.3.1. Vista principal del Sistema de Opinión en la Web

En la figura 3.2 se observa la vista principal del SCTW en español. Los elementos que lo integran 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 direcciona a la página oficial de la Universidad. 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. En seguida se encuentra un breve saludo de bienvenida en una etiqueta de texto.

Figura 3.2: 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 que contiene es por medio de una caja de texto. En la figura 3.3, se resalta el campo mediante un ovalo de color.

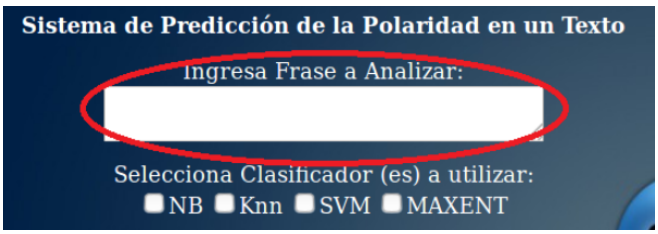


Figura 3.3: Área para ingresar texto a predecir.

Una vez ingresado el texto en el área definida, se analiza la polaridad del texto. La figura 3.4 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).

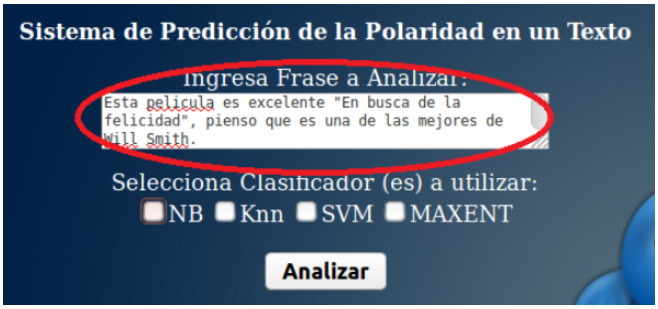


Figura 3.4: Ejemplo de entrada de texto al Sistema.

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 SCTW cuenta con cuatro tipos de Clasificadores o algoritmos: NBM, KNN, SVM y MaxEnt; los cuales tienen un mismo fin, asignar polaridad al texto ingresado al Sistema. En la figura 3.5 se señala con una marca de color la sección para elegir el Clasificador que se desea utilizar.

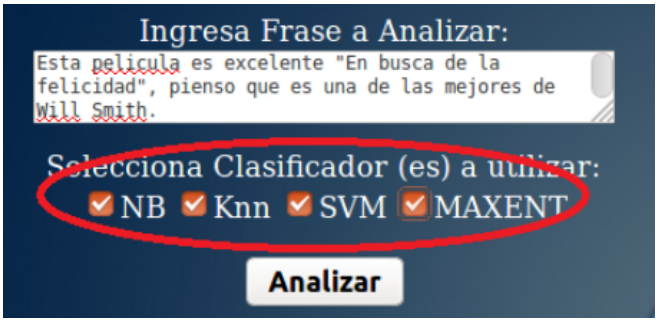


Figura 3.5: Selección de Clasificador en el Sistema Web.

Predicción del texto

Una vez que se seleccionó el Clasificador o Clasificadores deseados con el botón "Analizar", se aplica la acción que ejecuta el script que recibe los datos del usuario y realiza la Clasificación. En la figura 3.6 el elemento se encuentra encerrado por una marca de color para realizar la acción de Clasificación del texto ingresado.

Finalmente el sistema agrega sobre la misma vista la predicción de opinión o polaridad que se obtuvo con cada Clasificador seleccionado. La figura 3.7 indica con la

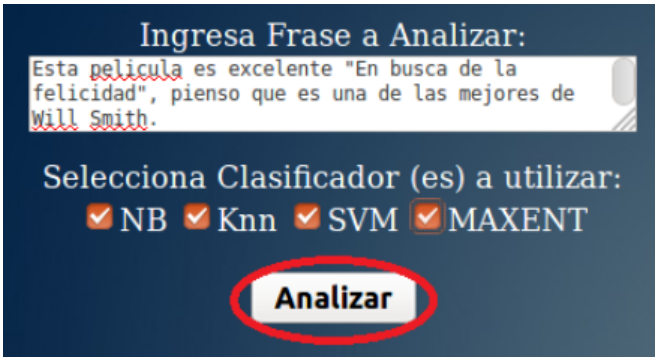


Figura 3.6: Botón para realizar la acción de Clasificación del texto ingresado

marca de color la salida del Sistema Web que el usuario obtendrá para el texto que fue introducido.

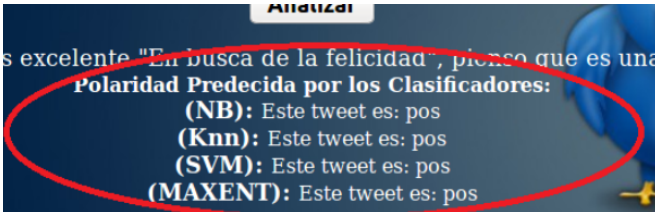


Figura 3.7: Vista final del Sistema de opinión, indicando la polaridad que los Clasificadores proporcionaron al texto de entrada.

Capítulo 4

Experimentos y resultados

Para responder a la pregunta ¿Existe una mejora en el desempeño al seleccionar características mediante el método χ^2 en los algoritmos de clasificación?, se presenta en este capítulo el proceso de evaluación bajo diferentes tipos de preprocesamiento de datos y diferentes cantidades de características seleccionadas para el entrenamiento y prueba de los algoritmos de clasificación que fueron descritos en la sección 2.3. Se detalla además el conjunto de datos que se construyo a partir de datos existentes en la red social Twitter, considerando el cambio de símbolo de los emoticonos por una palabra que representa el sentimiento u opinión. 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.

3. Cantidad de características seleccionadas en base a la técnica χ^2

4.1. Características de los datos utilizados

Los datos que se utilizaron en este trabajo corresponden a los empleados en el trabajo de [Esquivel16], con la diferencia de que fueron depurados algunos textos duplicados, la construcción del conjunto de datos: corpus, se pueden ver en el apéndice D. Consiste

en conjuntos de Tweets de diferentes temas comentados por diversos usuarios en dicha red social. Cada uno de los conjuntos de Tweets fueron divididos y clasificados en dos subconjuntos: Tweets Positivos y Tweets Negativos. En la tabla 4.1 se presenta la descripción de los dos conjuntos de datos para la experimentación.

Tabla 4.1: Corpus de Tweets, se cuenta con 2 conjuntos de datos en donde a los documentos les fue asignada una de las dos clases establecidas.

Conjuntos	***Tweets***		
	Positivos	**Negativos**	***Total de Tweets***
C1	537	504	*1, 041*
C2	2, 680	1, 959	*4, 639*

La tabla 4.2 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 4.2: Cantidad de palabras en los conjuntos de datos utilizados en los experimentos.

Conjuntos	***Cantidad de palabras***		
	Positivos	**Negativos**	***Tamaño Total***
C1	11, 432	10, 621	*22, 053*
C2	46, 193	33, 745	*79, 938*

En la tabla 4.3 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 se han de utilizar los algoritmos de clasificación.

Tabla 4.3: 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*

4.2. Preprocesamiento de los datos

Los tipos de preprocesamiento que se plantean y a los cuales se sometieron los Clasificadores son los siguientes:

Preprocesamiento 1. **Tweets Originales.** Del conjunto de Tweets que se obtuvieron originalmente, solamente se eliminaron todos los textos repetidos. En estos datos las palabras *excelente*, *Excelente* y *excelente!!!!* son tres características diferentes en el análisis.

Preprocesamiento 2. **Tweets Originales sin Emoticonos.** En este caso el preprocesamiento de los datos se refiere a reemplazar los emoticonos (Caritas) por texto con el significado correspondiente de acuerdo el tipo de emoticono, ver figura 4.1.

Figura 4.1: Se reemplaza el emoticono por el texto.

En la tabla 4.4 se muestra el nuevo tamaño del vocabulario una vez que se ha aplicado la sustitución de los emoticonos por texto.

Tabla 4.4: Tamaño del vocabulario en este tipo de preprocesamiento.

Conjuntos	***Tamaño del Vocabulario en Tweets***		
	Positivos	**Negativos**	***Tamaño Total***
C1	3, 194	2, 804	*5, 998*
C2	10, 204	7, 111	*17, 315*

Preprocesamiento 3. **Tweets con Texto sin Emoticonos sin Signos de Puntuación.** El preprocesamiento de los datos se refiere la sustitución de emoticonos por texto y eliminar todos los signos de puntuación de los datos originales, tales como: *: ; , . ? ¿ ! ¡*. Con este procesamiento de los datos las palabras *agradable!!!!!* y *agradable?* son una misma característica en la representación vectorial.

En la tabla 4.5 se muestra el tamaño del vocabulario resultante de sustitución y remoción de los signos de puntuación a los textos originales. Respecto al preprocesamiento 2, solamente en el conjunto de datos C2 se redujo 975 palabras en el tamaño del vocabulario.

Tabla 4.5: Tamaño del vocabulario al aplicar el preprocesamiento 3.

Conjunto	***Tamaño del Vocabulario***		
	Positivos	**Negativos**	***Tamaño Total***
C1	3, 183	2, 801	*5, 984*
C2	9, 669	6, 661	*16, 340*

Preprocesamiento 4. **Tweets con Texto en Minúsculas sin Emoticonos.** Cada carácter en mayúsculas es convertido a minúsculas. Ejemplos de palabras que se convierten en la misma característica son: *GRANDIOSO* y *grandioso*, también se ha sustituido emoticono (carita) por texto.

En la tabla 4.6 se muestra el tamaño del vocabulario resultante después de que los caracteres en los textos originales se sustituyen emoticonos por texto y se convirtieron a minúsculas.

Tabla 4.6: Tamaño del vocabulario después del preprocesamiento 4.

Conjuntos	***Tamaño del Vocabulario***		
	Positivos	**Negativos**	***Tamaño Total***
C1	2, 972	2, 622	*5, 594*
C2	9, 397	6, 598	*15, 995*

Preprocesamiento 5. **Tweets con Texto en Minúsculas sin Emoticonos sin Signos de Puntuación.** En este caso el preprocesamiento de los datos se refiere, cada carácter en mayúsculas es convertido a minúsculas, se ha sustituido emoticonos por texto, finalmente se ha eliminado 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 y sustituido emoticono por texto. Con este procesamiento de los datos las palabras *divertido!!!!!* y *divertido?* son una misma característica en la representación vectorial.

En la tabla 4.7 se muestra el nuevo tamaño del vocabulario después de que en los textos originales se convirtieran los caracteres a letras minúsculas, se han sustituido emoticono por texto y además se remo-

vieran los signos de puntuación.

Tabla 4.7: Tamaño del vocabulario de este tipo de preprocesamiento.

# Muestra	***Tamaño del Vocabulario***		
	Positivos	**Negativos**	***Tamaño Total***
C1	2, 961	2, 620	*5, 581*
C2	8,866	6, 159	*15, 025*

4.3. Descripción de la evaluación de los Clasificadores

Uno de los fines de la evaluación, es identificar la efectividad de los Clasificadores ante diferentes tipos de preprocesamiento de los datos. La figura 4.2 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 con ello tomar la mejor decisión en la implementación dentro de un sistema o aplicación comercial. El objetivo de la parte A) en la figura 4.2, que corresponde a la fase de entrenamiento o aprendizaje del sistema, es crear un modelo utilizando el subconjunto de datos para entrenar. Mientras que en la parte B), el modelo es utilizado para probar con datos que nunca se han visto por el Clasificador (conjunto de datos de prueba).

4.3.1. 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

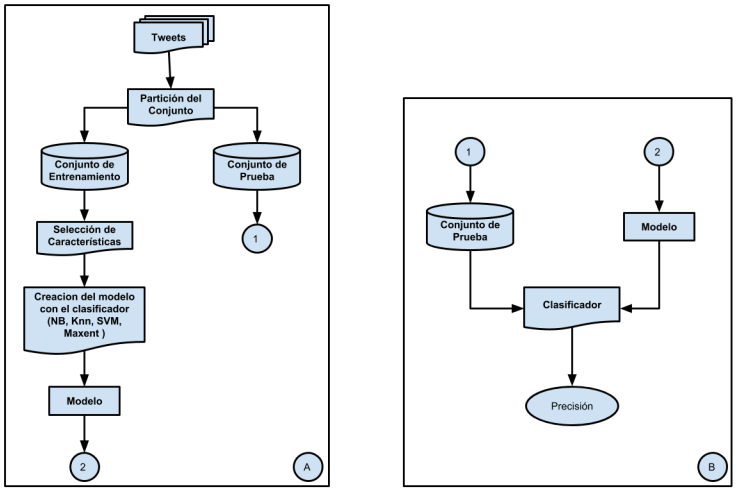


Figura 4.2: Diagrama de flujo del experimento implementado para obtener la evaluación de los Clasificadores. A) Entrenamiento del Clasificador (aprendizaje). B) Prueba del Clasificador (evaluación del aprendizaje)

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. Otra división de datos muy crucial para este trabajo consiste en tomar el porcentaje de selección de características, que va desde un 10 % hasta un 100 % de acuerdo al proceso de porcentaje que toma los datos de entrenamiento y de prueba, señalando que es de vital importancia la elección aleatoria de los datos para una mayor certeza de los resultados.

Las tablas 4.8 y 4.9 nos muestran las cantidades en que se dividieron los datos de los corpus utilizados en el presente trabajo.

Tabla 4.8: División de los datos indicando las cantidades de documentos utilizados para el entrenamiento y prueba de los Clasificadores (Conjunto C1).

Corpus Conjunto C1		
División de Datos en %	**# Documentos a Entrenar**	**# Documentos de Prueba**
10 % - 90 %	103	938
20 % - 80 %	207	834
30 % - 70 %	312	729
40 % - 60 %	415	626
50 % - 50 %	520	521
60 % - 40 %	624	417
70 % - 30 %	727	314
80 % - 20 %	832	209
90 % - 10 %	936	105

Tabla 4.9: División de los datos indicando las cantidades de documentos utilizados para el entrenamiento y prueba de los Clasificadores (Conjunto C2).

Corpus Conjunto C2		
División de Datos en %	**# Documentos a Entrenar**	**# Documentos de Prueba**
10 % - 90 %	463	4176
20 % - 80 %	927	3712
30 % - 70 %	1391	3248
40 % - 60 %	1855	2784
50 % - 50 %	2319	2320
60 % - 40 %	2783	1856
70 % - 30 %	3247	1392
80 % - 20 %	3711	928
90 % - 10 %	4175	464

4.4. Experimentos con los diferentes Clasificadores

En los experimentos se contemplaron los cuatro algoritmos descritos en el capítulo 2 en la sección del Marco Teórico. Dichos algoritmos fueron implementados en el lenguaje de programación Python, los detalles de la implementación se pueden ver en el apéndice A.

Para cada división de datos por Clasificador y por conjunto de datos, la métrica de evaluación que se reporta es la exactitud, la cual se promedia de las 50 ejecuciones que se realizó con cada algoritmo. La ecuación 4.1 define la métrica de evaluación reportada.

$$E = \frac{T_a}{T_a + T_e} \tag{4.1}$$

En donde T_a se refiere a los documentos (Tweets) que el Clasificador identificó como aciertos, es decir, les asigno la etiqueta de la clase en forma correcta; mientras que T_e se refiere a los documentos (Tweets) que el sistema clasificó erróneamente.

Los experimentos realizados permiten tener evidencia del comportamiento de los diferentes algoritmos de clasificación principalmente respecto a:

- Tipo de preprocesamiento de los datos
- Cantidad de documentos utilizados para el entrenamiento
- Cantidad de características seleccionadas en cada división de datos

4.5. Resultados de los Experimentos

4.5.1. Clasificador Naive Bayes Multinomial

En este primer experimento se hacen explícitos los resultados considerando la cantidad de características y la cantidad de documentos para entrenar los sistemas.

Preprocesamiento 1: Tweets Originales

La tabla 4.10 nos muestra los resultados de la exactitud promedio que obtuvo el Clasificador NB Multinomial por las 50 veces que se realizó el experimento, considerando un porcentaje de la cantidad de documentos para entrenar y un porcentaje de selección de características en la clasificación, y para cada una de las divisiones o particionamiento que se realizó con los datos para el entrenamiento y prueba del Clasificador.

El mejor resultado se obtuvo en 80 % de documentos en el entrenamiento y el 100 % de selección de características, la mayor precisión obtenida en este experimento fue de 0.781 en la escala de 0 a 1.

Considerando el conjunto de datos C2 para el mismo Clasificador NB Multinomial, los resultados de precisión promedio de 50 ejecuciones en las diferentes divisiones de datos

Tabla 4.10: Precisión del Clasificador Naive Bayes Multinomial promedio de 50 ejecuciones con los Tweets originales (C1).

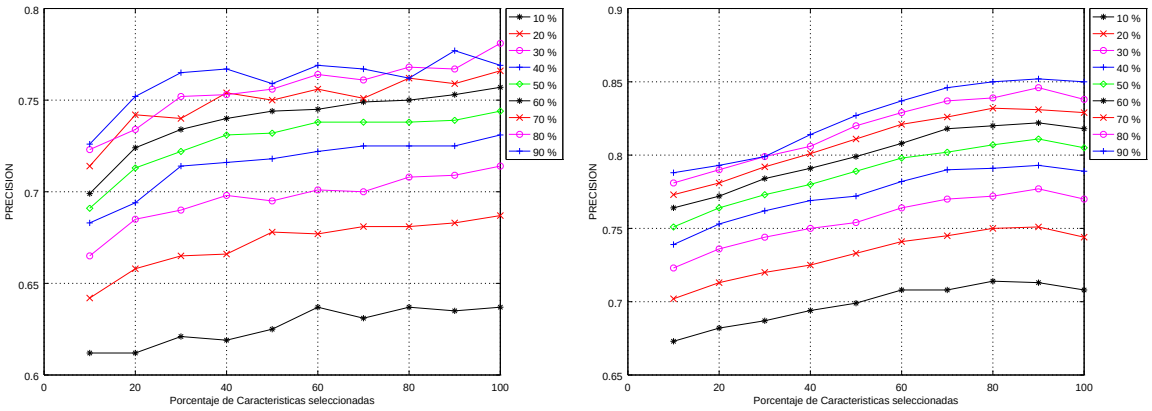
% Selección de características	***Cantidad en % de Documentos en el entrenamiento***								
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
10	0.612	0.642	0.665	0.683	0.691	0.699	0.714	0.723	0.726
20	0.612	0.658	0.685	0.694	0.713	0.724	0.742	0.734	0.752
30	0.621	0.665	0.690	0.714	0.722	0.734	0.740	0.752	0.765
40	0.619	0.666	0.698	0.716	0.731	0.740	0.754	0.753	0.767
50	0.625	0.678	0.695	0.718	0.732	0.744	0.750	0.756	0.759
60	0.637	0.677	0.701	0.722	0.738	0.745	0.756	0.764	0.769
70	0.631	0.681	0.700	0.725	0.738	0.749	0.751	0.761	0.767
80	0.637	0.681	0.708	0.725	0.738	0.750	0.762	0.768	0.762
90	0.635	0.683	0.709	0.725	0.739	0.753	0.759	0.767	0.777
100	0.637	0.687	0.714	0.731	0.744	0.757	0.766	**0.781**	0.769

los cuales fueron seleccionados en forma aleatoria, se muestran en la tabla 4.11. En este experimento el mejor resultado se observó en 90 % de documentos de entrenamiento y 90 % de selección de características, obteniendo la mayor precisión en 0.852. Para este experimento no fue necesario utilizar todas las características seleccionadas o el 100 % para obtener la mayor precisión como fue el resultado en el conjunto C1.

Tabla 4.11: Precisión del Clasificador Multinomial Naive Bayes promedio de 50 ejecuciones con los Tweets originales (C2).

% Selección de características	***Cantidad en % de Documentos en el entrenamiento***								
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
10	0.673	0.702	0.723	0.739	0.751	0.764	0.773	0.781	0.788
20	0.682	0.713	0.736	0.753	0.764	0.772	0.781	0.790	0.793
30	0.687	0.720	0.744	0.762	0.773	0.784	0.792	0.799	0.799
40	0.694	0.725	0.750	0.769	0.780	0.791	0.801	0.806	0.814
50	0.699	0.733	0.754	0.772	0.789	0.799	0.811	0.820	0.827
60	0.708	0.741	0.764	0.782	0.798	0.808	0.821	0.829	0.837
70	0.708	0.745	0.770	0.790	0.802	0.818	0.826	0.837	0.846
80	0.714	0.750	0.772	0.791	0.807	0.820	0.832	0.839	0.850
90	0.713	0.751	0.777	0.793	0.811	0.822	0.831	0.846	**0.852**
100	0.708	0.744	0.770	0.789	0.805	0.818	0.829	0.838	0.850

Para observar el comportamiento del desempeño del Clasificador en forma gráfica, es decir, ilustrar los resultados de las tablas 4.10 y 4.11, se presentan la figura 4.3. Para



(a) ***Conjunto C1*** **(b)** ***Conjunto C2***

Figura 4.3: Comportamiento del Clasificador NB Multinomial en relación a la precisión que se obtiene respecto de la cantidad de documentos utilizados para el entrenamiento en el el tipo de preprocesamiento 1.

el conjunto de datos C1 (Figura 4.3(a)), se observa que los resultados de precisión del Clasificador fue variable cuando se tuvo la cantidad de documentos del 70 al 90 %. Para el conjunto C2, el comportamiento fue creciente hasta el 90 % de características seleccionadas, mientras que respecto de la cantidad de documentos siempre se mejoró la precisión. En el 90 % de los datos se sufre de un pequeño sobre entrenamiento del sistema, ver la figura 4.3(b).

Preprocesamiento 2: Tweets Originales sin Emoticonos

Con el tipo 2 de preprocesamiento de datos el Clasificador Naive Bayes Multinomial requirió del total de características y del 90 % de documentos para obtener la mejor precisión promedio en las 50 ejecuciones de cada experimento realizado, con el conjunto de datos C1, ver los resultados en la tabla 4.12. Para este experimento en el conjunto C1 se obtuvo la mayor precisión con 0.778 en 90 % de cantidad de documentos para el entrenamiento y el total de características.

Al proceder con el conjunto de datos C2 con las mismas condiciones de desarrollo del experimento para el algoritmo, se observó que los mejores resultados se obtuvieron en la

Tabla 4.12: Precisión del Clasificador Naive Bayes Multinomial promedio de 50 ejecuciones con los Tweets sin emoticonos (C1).

% Selección de características	***Cantidad en % de Documentos en el entrenamiento***								
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
10	0.609	0.642	0.667	0.679	0.694	0.706	0.708	0.722	0.731
20	0.613	0.656	0.677	0.692	0.709	0.723	0.73	0.744	0.747
30	0.621	0.662	0.689	0.712	0.721	0.732	0.741	0.747	0.767
40	0.623	0.666	0.695	0.711	0.726	0.732	0.744	0.758	0.754
50	0.627	0.676	0.702	0.716	0.73	0.749	0.749	0.753	0.759
60	0.631	0.671	0.703	0.723	0.736	0.742	0.752	0.76	0.758
70	0.634	0.675	0.706	0.723	0.733	0.745	0.756	0.76	0.764
80	0.638	0.683	0.712	0.725	0.738	0.747	0.76	0.765	0.764
90	0.634	0.681	0.707	0.725	0.737	0.747	0.761	0.764	0.768
100	0.642	0.686	0.713	0.727	0.747	0.755	0.765	0.773	**0.778**

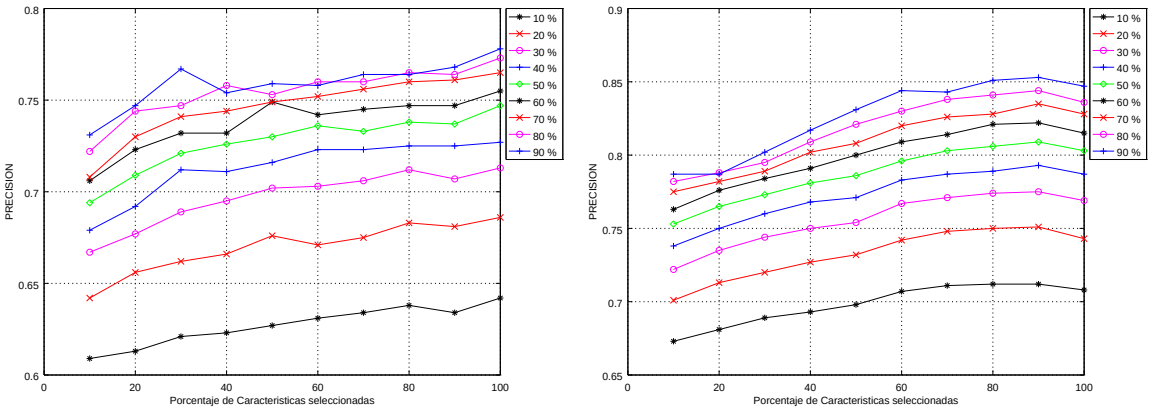
división de los datos que corresponde al 90 % de documentos para el entrenamiento y 90 % en el aspecto de selección de características, en donde el valor de precisión fue de 0.853, ver la tabla 4.13.

Respecto al tipo de preprocesamiento 1, los resultados para el conjunto C1 disminuyeron en el tipo de preprocesamiento 2 un 0.3 %, mientras que para el conjunto C2 se incremento un 0.1 %.

Tabla 4.13: Precisión del Clasificador Naive Bayes Multinomial promedio de 50 ejecuciones con los Tweets originales sin emoticonos (C2).

% Selección de características	***Cantidad en % de Documentos en el entrenamiento***								
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
10	0.673	0.701	0.722	0.738	0.753	0.763	0.775	0.782	0.787
20	0.681	0.713	0.735	0.75	0.765	0.776	0.782	0.788	0.787
30	0.689	0.72	0.744	0.76	0.773	0.784	0.789	0.795	0.802
40	0.693	0.727	0.75	0.768	0.781	0.791	0.802	0.809	0.817
50	0.698	0.732	0.754	0.771	0.786	0.8	0.808	0.821	0.831
60	0.707	0.742	0.767	0.783	0.796	0.809	0.82	0.83	0.844
70	0.711	0.748	0.771	0.787	0.803	0.814	0.826	0.838	0.843
80	0.712	0.75	0.774	0.789	0.806	0.821	0.828	0.841	0.851
90	0.712	0.751	0.775	0.793	0.809	0.822	0.835	0.844	**0.853**
100	0.708	0.743	0.769	0.787	0.803	0.815	0.828	0.836	0.847

La figura 4.4 nos muestra las gráficas del comportamiento del Clasificador respec-



(a) ***Conjunto C1*** **(b)** ***Conjunto C2***

Figura 4.4: Comportamiento del Clasificador NB Multinomial en relación a la precisión que se obtiene respecto de la cantidad de documentos utilizados para el entrenamiento en el tipo de preprocesamiento 2.

to al porcentaje de características seleccionadas en el tipo de preprocesamiento 2. Para el conjunto de datos C1, podemos ver en la figura 4.4(a) que el comportamiento es creciente hasta el 70 % tanto en la cantidad de características seleccionadas como de la cantidad de documentos utilizados para el entrenamiento. En la figura 4.4(b) los resultados del comportamiento son similares que los observados en el preprocesamiento 1 para el conjunto C2, se obtiene la mejor precisión utilizando solamente el 90 % de selección de características y 90 % de documentos para el entrenamiento, observándose un sobre entrenamiento del Clasificador en el 90 % de características seleccionadas.

Preprocesamiento 3: Tweets con Texto sin Emoticonos y sin Signos de Puntuación

Los resultados del Clasificador NB Multinomial con el tipo de preprocesamiento 3 de los datos para el conjunto C1 se muestran en la tabla 4.14. La mejor precisión se encuentra en 90 % documentos de entrenamiento y 100 % de selección de características, obteniendo la mayor precisión con 0.782. Respecto a los tipos de preprocesamiento 1 y 2, hasta el momento observados, se tiene una diferencia del 0.4 % respecto del tipo de preprocesamiento 1 y de

un 0.1 % del tipo de preprocesamiento 2.

Tabla 4.14: Precisión del Clasificador Naive Bayes Multinomial promedio de 50 ejecuciones con los Tweets sin emoticonos sin SP(C1).

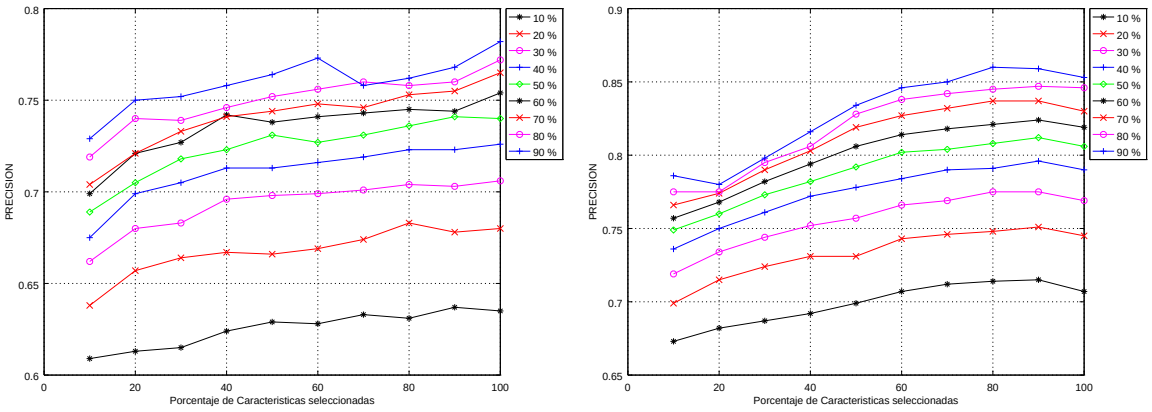
% Selección de características	***Cantidad en % de Documentos en el entrenamiento***								
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
10	0.609	0.638	0.662	0.675	0.689	0.699	0.704	0.719	0.729
20	0.613	0.657	0.68	0.699	0.705	0.721	0.721	0.74	0.750
30	0.615	0.664	0.683	0.705	0.718	0.727	0.733	0.739	0.752
40	0.624	0.667	0.696	0.713	0.723	0.742	0.741	0.746	0.758
50	0.629	0.666	0.698	0.713	0.731	0.738	0.744	0.752	0.764
60	0.628	0.669	0.699	0.716	0.727	0.741	0.748	0.756	0.773
70	0.633	0.674	0.701	0.719	0.731	0.743	0.746	0.76	0.758
80	0.631	0.683	0.704	0.723	0.736	0.745	0.753	0.758	0.762
90	0.637	0.678	0.703	0.723	0.741	0.744	0.755	0.76	0.768
100	0.635	0.68	0.706	0.726	0.74	0.754	0.765	0.772	**0.782**

Al aplicar el tipo de preprocesamiento 3 a los datos del conjunto C2, los resultados de precisión también se incrementaron respecto de los tipos de preprocesamiento anteriores en las siguientes proporciones: 0.8 % y 0.7 % para los tipos de preprocesamiento 1 y 2 respectivamente. Se observó además que con el 80 % de características seleccionadas se logra obtener el mejor resultado de precisión, véase la tabla 4.15 que indica con negritas el mejor resultado en el cruce de la fila que corresponde al 80 % de selección de características y en la columna que corresponde al 90 % de cantidad de documentos para el entrenamiento de sistema Clasificador.

En la figura 4.5(a) se observa gráficamente el comportamiento del Clasificador en relación con el tipo de preprocesamiento 3, del mismo modo podemos ver que utilizando el 100 % de selección de características se obtiene la mejor precisión y tomando el 90 % de documentos de entrenamiento y el resto de prueba, esto es en el conjunto C1. Se observa además que la figura 4.5(b), la cual corresponde al resultado del conjunto de datos C2, permite ver que en el 80 % de selección de características y con el 90 % de documentos para el entrenamiento se obtiene la mejor precisión. Debe señalarse que en este experimento se hace notar que no es necesario utilizar el 90 % o el 100 % de selección de características para obtener la mayor efectividad.

Tabla 4.15: Precisión del Clasificador Naive Bayes Multinomial promedio de 50 ejecuciones con los Tweets sin emoticonos sin SP(C2).

% Selección de características	***Cantidad en % de Documentos en el entrenamiento***								
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
10	0.673	0.699	0.719	0.736	0.749	0.757	0.766	0.775	0.786
20	0.682	0.715	0.734	0.75	0.76	0.768	0.774	0.775	0.780
30	0.687	0.724	0.744	0.761	0.773	0.782	0.79	0.795	0.798
40	0.692	0.731	0.752	0.772	0.782	0.794	0.803	0.806	0.816
50	0.699	0.731	0.757	0.778	0.792	0.806	0.819	0.828	0.834
60	0.707	0.743	0.766	0.784	0.802	0.814	0.827	0.838	0.846
70	0.712	0.746	0.769	0.79	0.804	0.818	0.832	0.842	0.850
80	0.714	0.748	0.775	0.791	0.808	0.821	0.837	0.845	**0.860**
90	0.715	0.751	0.775	0.796	0.812	0.824	0.837	0.847	0.859
100	0.707	0.745	0.769	0.79	0.806	0.819	0.83	0.846	0.853



(a) ***Conjunto C1*** **(b)** ***Conjunto C2***

Figura 4.5: Comportamiento del Clasificador NB Multinomial en relación a la precisión que se obtiene respecto de la cantidad de documentos utilizados para el entrenamiento en el tipo de preprocesamiento 3.

Preprocesamiento 4: Tweets con Texto en Minúsculas sin Emoticonos

Los resultados del conjunto de datos C1 con el tipo 4 de preprocesamiento de los datos para el Clasificador NB Multinomial se muestran en la tabla 4.16. La mayor precisión es de 0.775 con el 90 % tanto de cantidad de documentos para el entrenamiento, como cantidad de mejores características seleccionadas para la construcción de los vectores.

Comparando este mejor valor de precisión obtenido en este experimento, respecto a los valores obtenidos en los tipos de preprocesamientos anteriores, es el menor valor, es decir el peor caso hasta este momento. La diferencia con el mejor valor obtenido en el tipo de preprocesamiento 3, es de 0.007 %.

Tabla 4.16: Precisión del Clasificador Naive Bayes Multinomial promedio de 50 ejecuciones con los Tweets minúsculas sin emoticonos (C1).

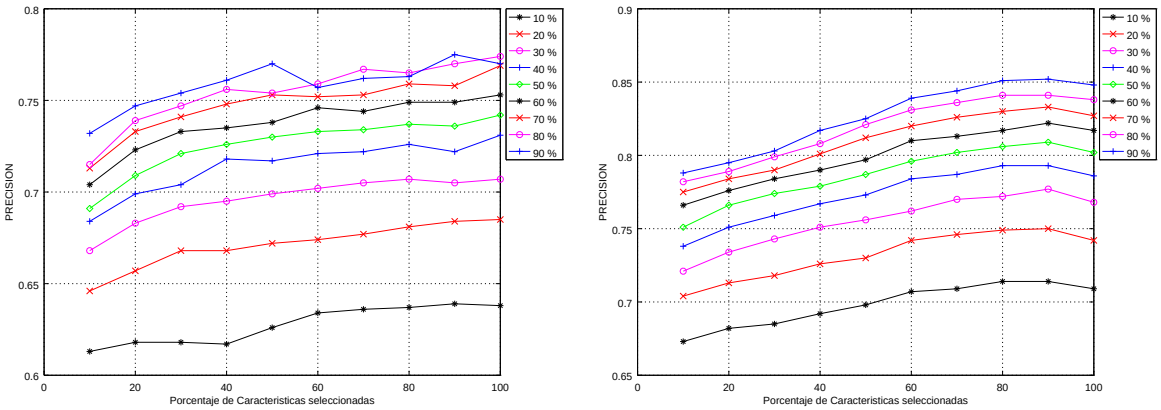
% Selección de características	***Cantidad en % de Documentos en el entrenamiento***								
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
10	0.613	0.646	0.668	0.684	0.691	0.704	0.713	0.715	0.732
20	0.618	0.657	0.683	0.699	0.709	0.723	0.733	0.739	0.747
30	0.618	0.668	0.692	0.704	0.721	0.733	0.741	0.747	0.754
40	0.617	0.668	0.695	0.718	0.726	0.735	0.748	0.756	0.761
50	0.626	0.672	0.699	0.717	0.73	0.738	0.753	0.754	0.770
60	0.634	0.674	0.702	0.721	0.733	0.746	0.752	0.759	0.757
70	0.636	0.677	0.705	0.722	0.734	0.744	0.753	0.767	0.762
80	0.637	0.681	0.707	0.726	0.737	0.749	0.759	0.765	0.763
90	0.639	0.684	0.705	0.722	0.736	0.749	0.758	0.77	**0.775**
100	0.638	0.685	0.707	0.731	0.742	0.753	0.769	0.774	0.770

Para el conjunto de datos C2, en este experimento el mejor resultado es el equivalente al tipo de preprocesamiento 1 de 0.852 en el 90 % de cantidad de documentos para el entrenamiento y el 90 % de selección de características, véase la tabla 4.17.

Tabla 4.17: Precisión del Clasificador Naive Bayes Multinomial promedio de 50 ejecuciones con los Tweets minúsculas sin emoticonos (C2).

% Selección de características	***Cantidad en % de Documentos en el entrenamiento***								
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
10	0.673	0.704	0.721	0.738	0.751	0.766	0.775	0.782	0.788
20	0.682	0.713	0.734	0.751	0.766	0.776	0.784	0.789	0.795
30	0.685	0.718	0.743	0.759	0.774	0.784	0.79	0.799	0.803
40	0.692	0.726	0.751	0.767	0.779	0.79	0.801	0.808	0.817
50	0.698	0.73	0.756	0.773	0.787	0.797	0.812	0.821	0.825
60	0.707	0.742	0.762	0.784	0.796	0.81	0.82	0.831	0.839
70	0.709	0.746	0.77	0.787	0.802	0.813	0.826	0.836	0.844
80	0.714	0.749	0.772	0.793	0.806	0.817	0.83	0.841	0.851
90	0.714	0.75	0.777	0.793	0.809	0.822	0.833	0.841	**0.852**
100	0.709	0.742	0.768	0.786	0.802	0.817	0.827	0.838	0.848

La figura 4.6 muestra el comportamiento de desempeño del Clasificador cuando se aplica el tipo de preprocesamiento 4 a ambos conjuntos de datos. La gráfica para el conjunto C1 (Figura 4.6(a)) nos muestra que el desempeño del Clasificador no se mantiene creciente a lo largo del porcentaje de características seleccionadas, principalmente en la cantidad de documentos para entrenar el sistema que corresponde al 90 %. Por el contrario vemos en la figura 4.6(b), la cual se refiere al conjunto de datos C2, que en todos los casos de cantidad de documentos para entrenar el sistema es creciente hasta el 90 % de características seleccionadas.



(a) ***Conjunto C1*** **(b)** ***Conjunto C2***

Figura 4.6: Comportamiento del Clasificador NB Multinomial en relación a la precisión que se obtiene respecto de la cantidad de documentos utilizados para el entrenamiento en el el tipo de preprocesamiento 4.

Preprocesamiento 5: Tweets con Texto en Minúsculas sin Emoticonos sin Signos de Puntuación

En este ultimo tipo de preprocesamiento de los datos, el Clasificador muestra el mejor resultado de precisión en la división de datos del 90 % de cantidad de documentos para el entrenamiento y el 100 % de características seleccionadas, véase la tabla 4.18. Con este tipo de preprocesamiento se logró mejorar todos los anteriores en este conjunto de datos, respecto al valor mínimo de precisión obtenida por los otros tipos de preprocesamiento, se

tiene un incremento del 1.4 % y respecto del segundo mejor resultado es del 0.7 %.

Tabla 4.18: Precisión del Clasificador Naive Bayes Multinomial promedio de 50 ejecuciones con los Tweets en minúsculas sin emoticonos sin signos de puntuación(C1).

% Selección de características	***Cantidad en % de Documentos en el entrenamiento***								
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
10	0.601	0.642	0.656	0.677	0.687	0.699	0.7	0.709	0.728
20	0.617	0.652	0.68	0.696	0.704	0.718	0.728	0.728	0.746
30	0.617	0.659	0.69	0.71	0.717	0.727	0.741	0.748	0.751
40	0.617	0.662	0.693	0.712	0.726	0.736	0.74	0.751	0.742
50	0.622	0.669	0.696	0.716	0.728	0.742	0.744	0.75	0.758
60	0.624	0.669	0.702	0.715	0.73	0.742	0.755	0.75	0.755
70	0.628	0.675	0.702	0.718	0.733	0.747	0.749	0.756	0.759
80	0.629	0.68	0.705	0.723	0.731	0.743	0.754	0.772	0.766
90	0.634	0.677	0.704	0.722	0.734	0.746	0.751	0.769	0.766
100	0.632	0.681	0.707	0.722	0.742	0.752	0.763	0.775	**0.789**

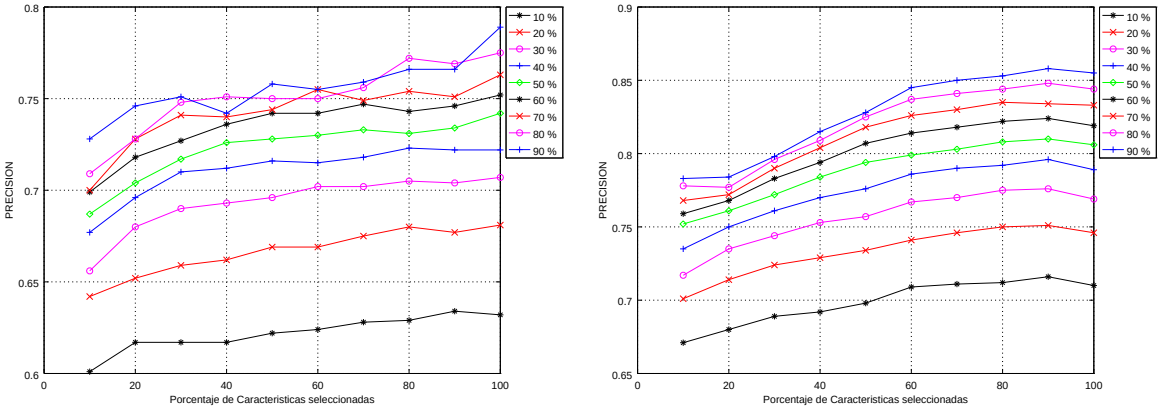
Para el conjunto de datos C2 y el tipo de preprocesamiento 5, el mejor resultado obtenido se encuentra con una diferencia inferior al tipo de preprocesamiento 3 del 0.2 %. La mayor precisión fue de 0.858, la cual se obtuvo en el 90 % de documentos para el entrenamiento y el 10 % de documentos de prueba. No se requiere el 100 % de características para lograr un mejor desempeño del Clasificador.

Tabla 4.19: Precisión del Clasificador Naive Bayes Multinomial promedio de 50 ejecuciones con los Tweets en minúsculas sin emoticonos sin signos de puntuación(C2).

% Selección de características	***Cantidad en % de Documentos en el entrenamiento***								
	10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
10	0.671	0.701	0.717	0.735	0.752	0.759	0.768	0.778	0.783
20	0.68	0.714	0.735	0.75	0.761	0.768	0.772	0.777	0.784
30	0.689	0.724	0.744	0.761	0.772	0.783	0.79	0.796	0.798
40	0.692	0.729	0.753	0.77	0.784	0.794	0.804	0.809	0.815
50	0.698	0.734	0.757	0.776	0.794	0.807	0.818	0.825	0.828
60	0.709	0.741	0.767	0.786	0.799	0.814	0.826	0.837	0.845
70	0.711	0.746	0.77	0.79	0.803	0.818	0.83	0.841	0.85
80	0.712	0.75	0.775	0.792	0.808	0.822	0.835	0.844	0.853
90	0.716	0.751	0.776	0.796	0.81	0.824	0.834	0.848	**0.858**
100	0.71	0.746	0.769	0.789	0.806	0.819	0.833	0.844	0.855

Finalmente en relación con el preprocesamiento 5, podemos observar en la figura

4.7(a) que nos muestra el comportamiento del Clasificador para cada uno de los conjuntos de datos, y permite observar los valores máximos en una forma más sencilla. El comportamiento del Clasificador es similar en ambos conjuntos a los comportamientos en los preprocesamientos anteriores. El mejor valor de precisión para C1 se observa en el 100 % de características seleccionadas y en C2 en el 90 %.



(a) ***Conjunto C1*** **(b)** ***Conjunto C2***

Figura 4.7: Comportamiento del Clasificador NB Multinomial en relación a la precisión que se obtiene respecto de la cantidad de documentos utilizados para el entrenamiento en el tipo de preprocesamiento 5.

4.5.2. Clasificador KNN

En los experimentos realizados con el Clasificador KNN, los valores de k que se utilizaron fueron: $k = \{7, 27, 105\}$. Se procedió de forma similar que con el Clasificador NB Multinomial, es decir, se obtuvo el promedio de 50 ejecuciones al elegir cierto porcentaje de características seleccionadas y en forma aleatoria un porcentaje de documentos para el entrenamiento. 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

La tabla 4.20 nos muestra los resultados de precisión que obtuvo el Clasificador KNN, considerando el porcentaje de selección de características indicado en las columnas, los renglones corresponden a la cantidad de documentos para entrenar el sistema. El mejor resultado se obtuvo cuando k es igual a 7 y el porcentaje de selección de característica es del 100 % y el 90 % de documentos para el entrenamiento, el cual fue de 0.723.

Tabla 4.20: Precisión del Clasificador KNN para valores de $\{k = 7, 27, 105\}$, con los Tweets originales del conjunto C1, considerando el porcentaje de selección de características utilizado.

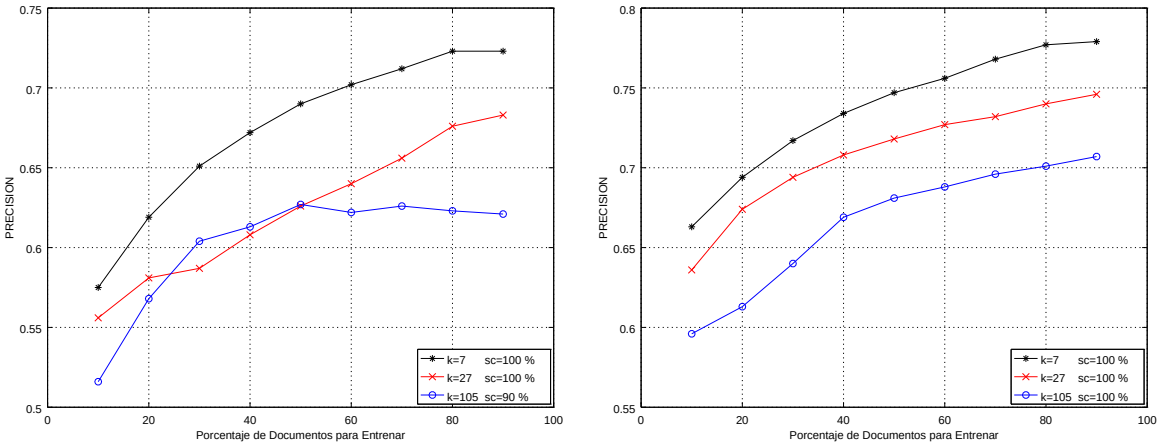
% Documentos en el entrenamiento	***% de Selección de características***		
	k=7 sc=100	**k=27 sc=100**	**k=105 sc=90**
10	0.575	0.556	0.516
20	0.619	0.581	0.568
30	0.651	0.587	0.604
40	0.672	0.608	0.613
50	0.69	0.626	0.627
60	0.702	0.64	0.622
70	0.712	0.656	0.626
80	0.723	0.676	0.623
90	**0.723**	0.683	0.621

Para el conjunto de datos C2, la tabla 4.21 nos muestra el mejor valor de precisión obtenido, el cual corresponde a 0.779, y se obtuvo con el valor de $k = 7$ con el 100 % de características seleccionadas y utilizando el 90 % de documentos para el entrenamiento.

El comportamiento del Clasificador KNN para los tres valores de k utilizados y ambos conjuntos de datos se muestran en la figura 4.8. En la gráfica 4.8(a), que representa los datos con el conjunto C1, se observa que con los valores de $k = \{7, 27\}$ el comportamiento del Clasificador es creciente, no así para cuando $k = 105$ que empieza a disminuir a partir del 50 % de documentos utilizados para entrenar, lo que señala un sobre entrenamiento del Clasificador. Para el conjunto de datos C2, ver la gráfica 4.8(b), el comportamiento del Clasificador es siempre creciente, obteniendo los mejores resultados en el 90 % de documentos utilizados en el entrenamiento. En ambos casos la mejor precisión se obtienen cuando k es igual a 7.

Tabla 4.21: Precisión del Clasificador KNN para el tipo de preprocesamiento 1 con el conjunto de datos C2 y considerando el 100 % de selección de características.

% Documentos en el entrenamiento	***% de Selección de características***		
	k=7 sc=100	**k=27 sc=100**	**k=105 sc=100**
10	0.663	0.636	0.596
20	0.694	0.674	0.613
30	0.717	0.694	0.64
40	0.734	0.708	0.669
50	0.747	0.718	0.681
60	0.756	0.727	0.688
70	0.768	0.732	0.696
80	0.777	0.74	0.701
90	**0.779**	0.746	0.707



(a) ***Conjunto C1*** **(b)** ***Conjunto C2***

Figura 4.8: Comportamiento de desempeño del Clasificador KNN, para $k = \{7, 27, 105\}$, considerando el tipo de preprocesamiento 1 para ambos conjuntos.

Preprocesamiento 2: Tweets Originales sin Emoticonos

Con el segundo preprocesamiento de los datos y someter al Clasificador KNN a las mismas condiciones que el preprocesamiento anterior, se obtuvieron los resultados promedios de precisión en la clasificación para el conjunto C1 que se muestran en la tabla 4.22. El mejor resultado se observa con el 90 % de documentos para el entrenamiento y el 100 % de características ($sc = 100$) en el vocabulario de datos. Con este tipo de preprocesamiento se

obtuvó un 0.3 % de mejora respecto del tipo de preprocesamiento 1 en el mejor resultado que se observa cuando el valor de k es 7.

Tabla 4.22: Precisión del Clasificador KNN con el tipo de preprocesamiento 2 con los datos del conjunto C1, *sc* indica el porcentaje de selección de características en donde se obtuvieron los mejores resultados.

% Documentos en el entrenamiento	***% de Selección de características***		
	k=7 sc=100	**k=27 sc=100**	**k=105 sc=20**
10	0.577	0.547	0.516
20	0.617	0.58	0.539
30	0.646	0.584	0.557
40	0.668	0.603	0.576
50	0.682	0.621	0.583
60	0.697	0.631	0.613
70	0.701	0.647	0.605
80	0.72	0.674	0.633
90	**0.726**	**0.693**	**0.657**

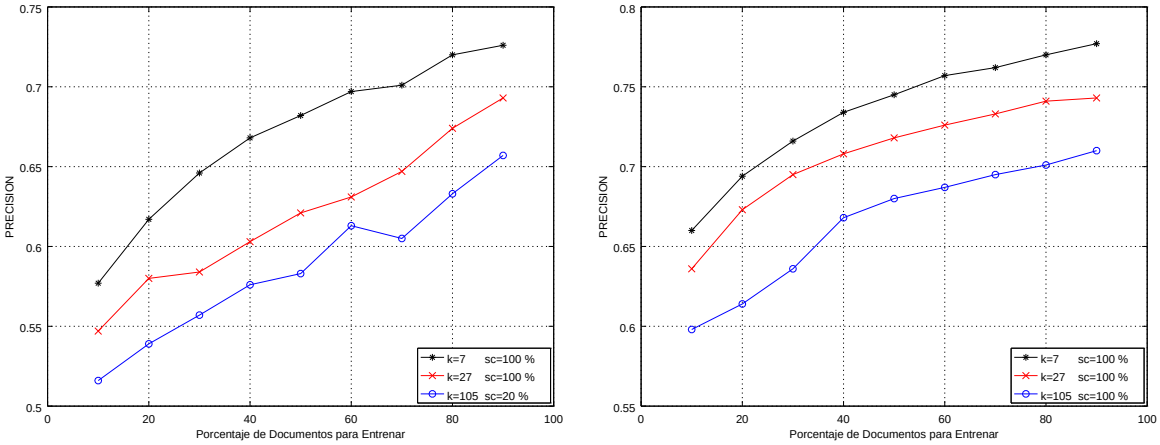
De forma similar en la tabla tabla 4.23 se muestran los resultados que se obtuvieron para el conjunto C2. El valor de mejor precisión fue de 0.777, con valor de k igual a 7 utilizando el total de características en el vocabulario y el 90 % de documentos para el entrenamiento. Se tiene una disminución del 0.2 % y de 0.3 % con $k = \{7, 27\}$ respectivamente en relación a lo obtenido en el tipo de preprocesamiento 1. Mientras que con $k = 105$ este tipo de preprocesamiento permitió un incremento del 0.3 %.

Tabla 4.23: Precisión del Clasificador KNN con el tipo de preprocesamiento 2 en el conjunto de datos C2.

% Documentos en el entrenamiento	***% de Selección de características***		
	k=7 sc=100	**k=27 sc=100**	**k=105 sc=100**
10	0.66	0.636	0.598
20	0.694	0.673	0.614
30	0.716	0.695	0.636
40	0.734	0.708	0.668
50	0.745	0.718	0.68
60	0.757	0.726	0.687
70	0.762	0.733	0.695
80	0.77	0.741	0.701
90	**0.777**	**0.743**	**0.71**

El comportamiento del Clasificador con el tipo de preprocesamiento 2 se observa en la figura 4.9. La gráfica 4.9(a), que representa los resultados utilizando el conjunto C1, indica que incrementar el valor de k el aprendizaje del Clasificador se vuelve en una generalización de los datos muy pobre. En los casos de la $k = \{7, 27\}$ el comportamiento del Clasificador es creciente.

Por otro lado, con el conjunto C2, ver la gráfica 4.9(b), podemos observar que en todos los casos la función es creciente, el valor de desempeño es mejor cuando se trabaja con el valor de k igual a 7, en todos los casos se requiere el total de características para obtener los mejores resultados de clasificación.



(a) ***Conjunto C1*** **(b)** ***Conjunto C2***

Figura 4.9: Comportamiento de desempeño del Clasificador KNN, para $k = \{7, 27, 105\}$, considerando el tipo de preprocesamiento 2 para ambos conjuntos.

Preprocesamiento 3: Tweets con Texto sin Emoticonos sin Signos de Puntuación

El tipo de preprocesamiento 3 aplicado al conjunto de datos C1 para el Clasificador KNN, seleccionando los mejores resultados obtenidos por la cantidad de características seleccionadas, produjo los valores de precisión que se muestran en la tabla 4.24. Con los valores de $k = 7$ se obtiene la mejor precisión con 0.719 en la cantidad de documentos utilizados para el entrenamiento del 90 %. El Clasificador respondió en forma negativa nue-

vamente a este tipo de preprocesamiento de los datos, debido a que disminuyo su eficiencia en un 0.4 % y 0.7 % respecto a los tipos de preprocesamiento 1 y 2 respectivamente.

Tabla 4.24: Precisión del Clasificador KNN con el tipo de preprocesamiento 3 en el conjunto de datos C1.

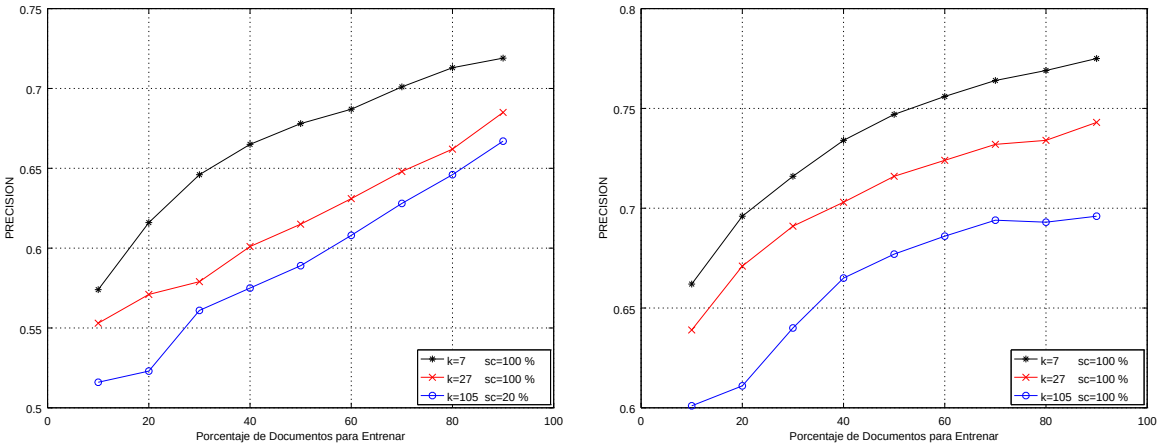
% Documentos en el entrenamiento	***% de Selección de características***		
	k=7 sc=100	**k=27 sc=100**	**k=105 sc=20**
10	0.574	0.553	0.516
20	0.616	0.571	0.523
30	0.646	0.579	0.561
40	0.665	0.601	0.575
50	0.678	0.615	0.589
60	0.687	0.631	0.608
70	0.701	0.648	0.628
80	0.713	0.662	0.646
90	**0.719**	**0.685**	**0.667**

En los resultados que se obtuvieron para el conjunto de datos C2, véase la tabla 4.25, se observa nuevamente una disminución de los resultados respecto a los tipos de prerprocesamiento 1 y 2 en un 0.4 % y 0.2 % respectivamente, esto en el mejor resultado que se obtuvo de 0.775 con el valor de $k = 7$, utilizando la división de datos de 90 % de documentos de entrenamiento y el 10 % de documentos para prueba, requiriendo el total de características en el vocabulario.

Tabla 4.25: Precisión del Clasificador KNN con el tipo de preprocesamiento 3 en el conjunto de datos C2.

% Documentos en el entrenamiento	***% de Selección de características***		
	k=7 sc=100	**k=27 sc=100**	**k=105 sc=100**
10	0.662	0.639	0.601
20	0.696	0.671	0.611
30	0.716	0.691	0.64
40	0.734	0.703	0.665
50	0.747	0.716	0.677
60	0.756	0.724	0.686
70	0.764	0.732	0.694
80	0.769	0.734	0.693
90	**0.775**	**0.743**	**0.696**

En la figura 4.10 se observan las gráficas del comportamiento del Clasificador para ambos conjuntos de datos. Tanto la gráfica 4.10(a) y 4.10(b) ilustran que en estas condiciones el Clasificador mantiene un comportamiento creciente en la precisión obtenida al clasificar los datos de prueba.



(a) ***Conjunto C1*** **(b)** ***Conjunto C2***

Figura 4.10: Comportamiento de desempeño del Clasificador KNN, para $k = \{7, 27, 105\}$, considerando el tipo de preprocesamiento 3 para ambos conjuntos.

Preprocesamiento 4: Tweets con Texto en Minúsculas sin Emoticonos

Al aplicar el cuarto preprocesamiento a los datos, el Clasificador KNN tuvo un comportamiento de desempeño creciente respecto a la cantidad de datos que se utilizaron para el entrenamiento y la cantidad de características seleccionadas, en ambos conjuntos solamente para los valores de $k = \{7, 27\}$. Los resultados del promedio de la precisión obtenida en las 50 ejecuciones por cada particionamiento de los datos para el conjunto de datos C1 se muestran en la tabla 4.26. Además, para este mismo conjunto de datos, se obtuvo la mejor precisión en la división de datos del 90 % de documentos de entrenamiento y el 100 % de las características en el vocabulario.

La tabla 4.27 muestra los resultados para el conjunto de datos C2, en la que se observa que el valor 0.779, el cual corresponde al 90 % de documentos para el entrenamiento

Tabla 4.26: Precisión del Clasificador KNN con el tipo de preprocesamiento 4 en el conjunto de datos C1.

% Documentos en el entrenamiento	***% de Selección de características***		
	k=7 sc=100	**k=27 sc=100**	**k=105 sc=20**
10	0.569	0.548	0.516
20	0.619	0.579	0.525
30	0.648	0.581	0.555
40	0.668	0.604	0.569
50	0.683	0.622	0.587
60	0.696	0.634	0.619
70	0.706	0.641	0.602
80	0.72	0.667	0.629
90	**0.719**	**0.693**	**0.656**

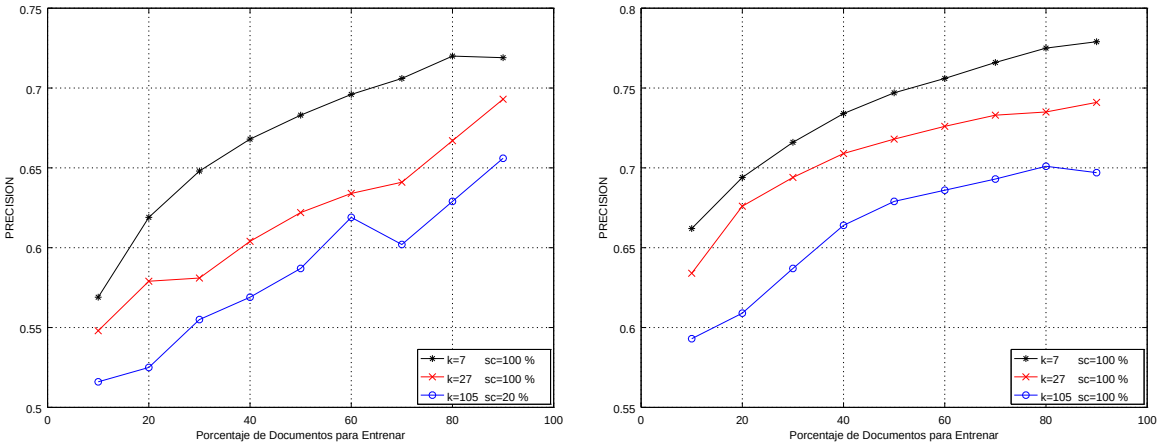
y con el total de características del vocabulario, es el mejor resultado que proporciona el Clasificador, equivalente al desempeño obtenido con el tipo de preprocesamiento 1, que al momento es el mejor resultado que se ha observado para este Clasificador.

Tabla 4.27: Precisión del Clasificador KNN con el tipo de preprocesamiento 4 en el conjunto de datos C2.

% Documentos en el entrenamiento	***% de Selección de características***		
	k=7 sc=100	**k=27 sc=100**	**k=105 sc=100**
10	0.662	0.634	0.593
20	0.694	0.676	0.609
30	0.716	0.694	0.637
40	0.734	0.709	0.664
50	0.747	0.718	0.679
60	0.756	0.726	0.686
70	0.766	0.733	0.693
80	0.775	0.735	0.701
90	**0.779**	**0.741**	**0.697**

La figura 4.11(a) nos muestra el comportamiento del desempeño del Clasificador KNN para el conjunto C1 de datos en las tres variantes del valor de k, se observa que se obtiene el mejor resultado cuando k es igual a 7, utilizando el 90 % de documentos de entrenamiento y el 100 % de características. Con $k = 105$ no siempre es creciente su comportamiento en ambos conjuntos de datos, como se puede observar en las gráficas 4.11(a)

y 4.11(b), en la primera se observa una disminución del valor de precisión en el 70 % de documentos para entrenar, mientras que en la segunda es hasta el 90 %.



(a) ***Conjunto C1*** **(b)** ***Conjunto C2***

Figura 4.11: Comportamiento de desempeño del Clasificador KNN, para $k = \{7, 27, 105\}$, considerando el tipo de preprocesamiento 4 para ambos conjuntos.

Preprocesamiento 5: Tweets con Texto en Minúsculas sin Emoticonos sin Signos de Puntuación

Los resultados del Clasificador KNN con el tipo de preprocesamiento 5 para el conjunto de datos C1 se muestra en la tabla 4.28. El mejor valor de precisión se observa en el valor de $k = 7$ en el 90 % de la cantidad de documentos utilizados para el entrenamiento y con el total de características en el vocabulario. En este tipo de preprocesamiento se obtuvo el menor valor de precisión (0.717) de los mejores registrados en los diferentes tipos de preprocesamiento. Diferencias por debajo del 0.2 %, 0.6 % y 0.9 % respecto a los tipos de preprocesamiento 3, 1 y 2 respectivamente.

El efecto que se manifestó con los tipos de preprocesamiento de los datos y con la selección de características para el conjunto de datos C1, en el Clasificador KNN, señala que es suficiente con cambiar los símbolos de los emoticonos y mantener los datos con la cantidad de características mayores.

Tabla 4.28: Precisión del Clasificador KNN con el tipo de preprocesamiento 5 en el conjunto de datos C1.

% Documentos en el entrenamiento	***% de Selección de características***		
	k=7 sc=100	**k=27 sc=100**	**k=105 sc=20**
10	0.571	0.548	0.516
20	0.613	0.577	0.526
30	0.647	0.584	0.557
40	0.663	0.601	0.581
50	0.681	0.612	0.599
60	0.692	0.634	0.6
70	0.695	0.651	0.605
80	0.704	0.664	0.644
90	**0.717**	**0.668**	**0.677**

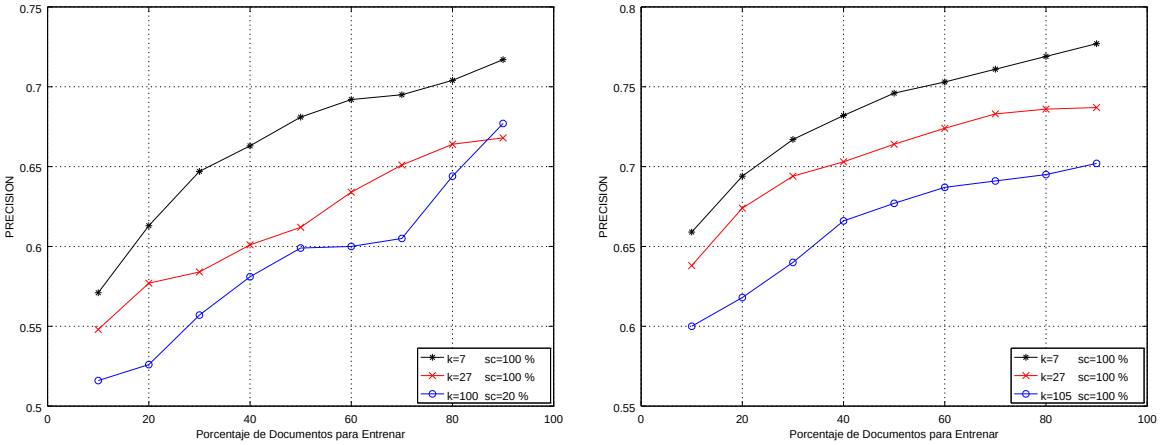
Aplicando el mismo procedimiento a los datos del conjunto de datos C2, se obtuvieron los resultados que se muestran en la tabla 4.29, en donde el mejor desempeño de precisión se obtiene con $k = 7$, utilizando el 90 % de documentos para el entrenamiento y el total de características en el vocabulario. Respecto a los tipos de preprocesamiento anterior, se observa que el valor de mejor precisión, el cual es de 0.777, es igual en el tipo de preprocesamiento 2, es menor en los tipos de preprocesamiento 1 y 4 con un 0.2 %, y es mayor en el tipo de preprocesamiento 3 con 0.2 %.

Tabla 4.29: Precisión del Clasificador KNN con el tipo de preprocesamiento 5 en el conjunto de datos C2.

% Documentos en el entrenamiento	***% de Selección de características***		
	k=7 sc=100	**k=27 sc=100**	**k=105 sc=100**
10	0.659	0.638	0.6
20	0.694	0.674	0.618
30	0.717	0.694	0.64
40	0.732	0.703	0.666
50	0.746	0.714	0.677
60	0.753	0.724	0.687
70	0.761	0.733	0.691
80	0.769	0.736	0.695
90	**0.777**	**0.737**	**0.702**

Finalmente, en la figura 4.12 se muestra el comportamiento de desempeño del

Clasificador KNN con el tipo de preprocesamiento 5 para ambos conjuntos de datos. Un comportamiento creciente en todos los casos, para ambos conjuntos de datos se observa en las gráficas 4.12(a) y 4.12(b).



(a) ***Conjunto C1*** **(b)** ***Conjunto C2***

Figura 4.12: Comportamiento de desempeño del Clasificador KNN, para $k = \{7, 27, 105\}$, considerando el tipo de preprocesamiento 5 para ambos conjuntos.

4.5.3. 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. Las condiciones de evaluación realizada son las mismas que los dos Clasificadores anteriores.

Preprocesamiento 1: Tweets Originales

Primero se obtuvieron los resultados de precisión que arrojó el Clasificador SVM en las 50 ejecuciones que se realizó el experimento para cada una de las divisiones o particionamiento de los datos para el entrenamiento, y de los cuales se consideró la selección de características. Dichos resultados para el conjunto C1 y conjunto C2 se muestran en la tabla 4.30. Se observan que los mejores resultados para el conjunto C1 se obtuvieron en la división de datos del 90 % de documentos de entrenamiento, así mismo utilizando el 90 % de

selección de características obteniendo el mejor desempeño de precisión con 0.755. Para el conjunto C2 como resultado se obtuvo la mejor precisión con 0.870 en la división de datos utilizando el 90 % de documentos para el entrenamiento y el 10 % de prueba, utilizando el 100 % de características en el vocabulario.

Tabla 4.30: Precisión promedio del Clasificador SVM con el tipo de preprocesamiento 1, para ambos conjuntos de datos C1 y C2.

% Documentos en el entrenamiento	***% de Selección de características***	
	Conjunto C1 sc=90	**Conjunto C2 sc=100**
10	0.635	0.715
20	0.679	0.758
30	0.704	0.787
40	0.723	0.807
50	0.74	0.826
60	0.748	0.84
70	0.766	0.85
80	0.769	0.862
90	**0.775**	**0.87**

Preprocesamiento 2: Tweets Originales sin Emoticonos

Al proceder con el tipo de preprocesamiento 2, considerando el mismo procedimiento que el preprocesamiento anterior, se obtuvieron los resultados que se presentan en la tabla 4.31. En ambos conjuntos de datos se obtuvo el mejor valor promedio de precisión, cuando la división de datos fue del 90 % de documentos para el entrenamiento y con el 90 % de selección de características. Los resultados muestran un comportamiento que al incrementar la cantidad de datos para el entrenamiento, incrementa el desempeño del Clasificador, lo que de alguna forma establece que la generalización de los datos tiene el comportamiento de un modelo ideal. Los mejores valores observados son superiores en un 0.2 % respecto a los obtenidos en el tipo de preprocesamiento 1 para ambos conjuntos de datos. Se indica además la diferencia en la cantidad de características utilizadas en el conjunto de datos C2 para obtener ese mismo resultado, la cual disminuye en un 10 %.

Tabla 4.31: Precisión promedio del Clasificador SVM con el tipo de preprocesamiento 2, para ambos conjuntos de datos C1 y C2.

% Documentos en el entrenamiento	***% de Selección de características***	
	Conjunto C1 sc=90	**Conjunto C2 sc=90**
10	0.63	0.717
20	0.68	0.759
30	0.708	0.786
40	0.725	0.808
50	0.74	0.823
60	0.749	0.836
70	0.757	0.849
80	0.766	0.854
90	**0.777**	**0.872**

Preprocesamiento 3: Tweets con Texto sin Emoticonos sin Signos de Puntuación

Al proceder con el tipo de preprocesamiento 3 en el conjunto de datos C1, los resultados del Clasificador SVM indican que el desempeño tiene un comportamiento creciente, en donde la mejor precisión es de 0.777 para el conjunto de datos C1 y de 0.872 para el conjunto de datos C2, en ambos casos los mejores resultados se obtienen en el 90 % de mejores características seleccionadas y en el 90 % de la cantidad de documentos utilizados para el entrenamiento, véase la tabla 4.32 que muestran los resultados que se describen.

Tabla 4.32: Precisión promedio del Clasificador SVM con el tipo de preprocesamiento 3, para ambos conjuntos de datos C1 y C2.

% Documentos en el entrenamiento	***% de Selección de características***	
	Conjunto C1 sc=90	**Conjunto C2 sc=90**
10	0.631	0.719
20	0.671	0.762
30	0.7	0.787
40	0.72	0.808
50	0.734	0.825
60	0.747	0.84
70	0.754	0.851
80	0.756	0.862
90	0.777	0.872

Comparando los resultados respecto a los tipos de preprocesamiento anteriores,

se observa que los mejores resultados son iguales al tipo de preprocesamiento 2 y con una diferencia de 0.2 % por arriba del resultado en el tipo de preprocesamiento 1.

Preprocesamiento 4: Tweets con Texto en Minúsculas sin Emoticonos

Los resultados obtenidos del Clasificador SVM con el tipo de preprocesamiento 4 aplicado a ambos conjuntos de datos C1 y C2 se muestran en la tabla 4.33. El comportamiento de los resultados es creciente, a mayor cantidad de datos para el entrenamiento, mayor el valor de la precisión obtenida. Los mejores resultados se observaron con el total de características en el vocabulario (sc=100) y en el 90 % de cantidad de documentos para el entrenamiento. Respecto a los resultados obtenidos en los tipos de preprocesamiento anteriores, con el conjunto de datos C1 se mantienen el resultado de 0.777 que se tienen en los tipos de preprocesamiento 2 y 3. Con el conjunto de datos C2, el mejor resultado se ve disminuido en un 0.1 % y un 0.3 % respecto a los tipos de preprocesamiento 1 y 2 respectivamente, el mejor valor obtenido en el tipo de preprocesamiento 2 es igual al tipo de preprocesamiento 3.

Tabla 4.33: Precisión promedio del Clasificador SVM con el tipo de preprocesamiento 4, para ambos conjuntos de datos C1 y C2.

% Documentos en el entrenamiento	***% de Selección de características***	
	Conjunto C1 sc=100	**Conjunto C2 sc=100**
10	0.634	0.716
20	0.675	0.758
30	0.699	0.786
40	0.725	0.805
50	0.737	0.824
60	0.751	0.836
70	0.748	0.85
80	0.764	0.862
90	0.777	0.869

Preprocesamiento 5: Tweets con Texto en Minúsculas sin Emoticonos sin Signos de Puntuación

Finalmente se realizó el experimento con el tipo de preprocesamiento 5 con el Clasificador SVM. Los resultados promedio de precisión en la clasificación para cada uno de los conjuntos de datos se muestran en la tabla 4.34. Los mejores resultados se observaron con el total de características en el vocabulario y en el 90 % de cantidad de documentos utilizados para el entrenamiento en ambos conjuntos de datos. La mejor precisión en el conjunto de datos C1 es igual al tipo de preprocesamiento 1 e inferior en un 0.2 % respecto del resto de los tipos de preprocesamiento de los datos. En el conjunto de datos C2, el valor de precisión 0.877 es el mejor valor de precisión observado en el experimento, por arriba en un 0.7 % del tipo de preprocesamiento 1, 0.5 % respecto del tipo de preprocesamiento 2 y 3, y un 0.8 % respecto del tipo de preprocesamiento 4.

Tabla 4.34: Precisión promedio del Clasificador SVM con el tipo de preprocesamiento 5, para ambos conjuntos de datos C1 y C2.

% Documentos en el entrenamiento	***% de Selección de características***	
	Conjunto C1 sc=100	**Conjunto C2 sc=100**
10	0.626	0.717
20	0.679	0.759
30	0.703	0.785
40	0.717	0.806
50	0.736	0.823
60	0.749	0.839
70	0.756	0.849
80	0.756	0.865
90	**0.775**	**0.877**

Al observar en forma gráfica los resultados de los diferentes preprocesamientos en forma individual para cada conjunto de datos, los cuales corresponden al comportamiento de desempeño del Clasificador, ver las figuras 4.13 y 4.14, se hacen las observaciones que se describen a continuación. Para el conjunto de datos C1, se observa que el tipo de preprocesamiento 4 no siempre su comportamiento es creciente, debido a que en el 70 % de documentos para entrenar sufre de un decremento en el valor de precisión promedio obtenida, sin embargo para el resto de experimentos, la gráfica indica que el comportamiento es

creciente en general. Además se observa que no existe un comportamiento de los resultados que tenga una diferencia en donde se supere siempre a alguno de los otros tipos de preprocesamiento. A partir del 70 % de documentos para entrenar, la gráfica en 4.13 sugiere que el tipo de preproceamiento 1 y 2 son las mejores opciones para que el Clasificador obtenga su mejor desempeño.

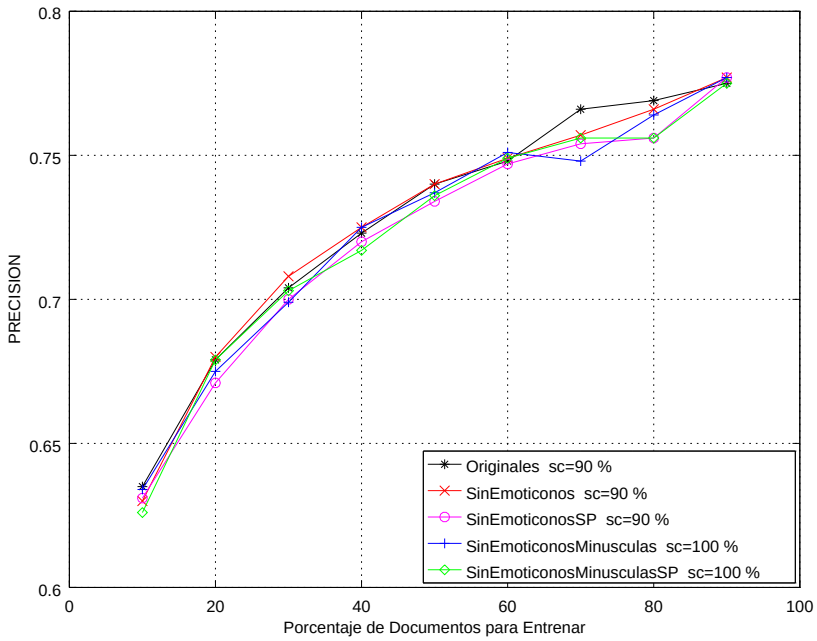


Figura 4.13: Comportamiento de desempeño del Clasificador SVM en los diferentes tipos de preprocesamiento de los datos sobre el conjunto C1.

Para el conjunto de datos C2, en la figura 4.14, se observa que el comportamiento de desempeño del Clasificador es siempre en forma creciente. La gráfica sugiere que a partir del 80 % de documentos utilizados para el entrenamiento, el tipo de preprocesamiento 5 permite obtener los mejores resultados de precisión con el total de características en el vocabulario. En general no existe una diferencia clara en los resultados obtenidos para cada tipo de preprocesamiento de los datos que identifique a alguno como opción o sugerencia. Por último, se observa que con el 90 % de características seleccionadas y el tipo de preprocesamiento 3 se obtiene el mejor resultado en diferentes puntos de la gráfica hasta el 70 % de documentos utilizados para entrenar.

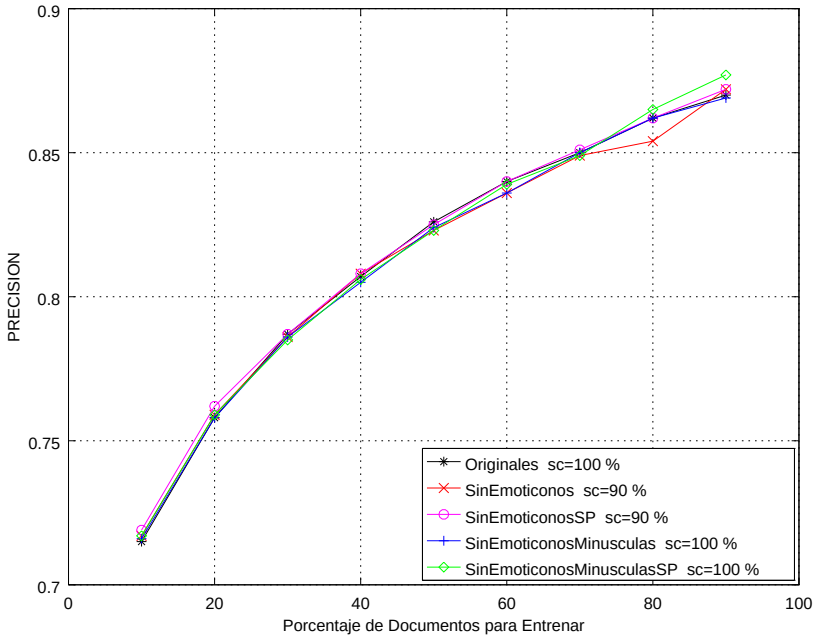


Figura 4.14: Comportamiento de desempeño del Clasificador SVM en los diferentes tipos de preprocesamiento de los datos sobre el conjunto C2.

4.5.4. Clasificador Máxima Entropía (MaxEnt)

En este último experimento, el Clasificador de Máxima Entropía (MaxEnt), se procede de forma similar que los tres Clasificadores anteriores utilizando las mismas técnicas para determinar el promedio de 50 ejecuciones, al aplicar al conjunto de datos los diferentes tipos de preprocesamientos.

Preprocesamiento 1: Tweets Originales

Con el primer preprocesamiento de los datos en ambos conjuntos C1 y C2, se tienen los resultados que se muestran en la tabla 4.35. En el caso del conjunto C1, con el 90 % de características seleccionadas se obtuvieron los mejores resultados, mientras que para el conjunto C2 fueron necesarias el total de características en el vocabulario. En ambos conjuntos de datos se observa un comportamiento creciente en el valor de precisión obtenido respecto a la cantidad de documentos utilizados para el entrenamiento, lográndose los mejores resultados en el 90 % de documentos para entrenar.

Tabla 4.35: Precisión del Clasificador MaxEnt, promedio de 50 ejecuciones con el tipo de preprocesamiento 1 en los conjuntos de datos C1 y C2.

% Documentos en el entrenamiento	***% de Selección de características***	
	Conjunto C1 sc=90	**Conjunto C2 sc=100**
10	0.629	0.702
20	0.677	0.733
30	0.699	0.759
40	0.718	0.777
50	0.726	0.789
60	0.741	0.8
70	0.75	0.811
80	0.753	0.82
90	**0.763**	**0.828**

Preprocesamiento 2: Tweets Originales sin Emoticonos

Con el tipo de preprocesamiento 2 en ambos conjuntos de datos, se obtienen los resultados de la tabla 4.36. En el conjunto C1 nuevamente se obtienen los mejores resultados con el 90 % de características seleccionadas, mientras que con el conjunto C2 se requiere el total de palabras en el vocabulario. En el conjunto de datos C1 se obtiene la mejor valor de precisión en la división de datos 90 % de documentos para el entrenamiento de 0.755, inferior respecto del tipo de preprocesamiento 1 en un 0.8 %, lo que nos indica que para este conjunto de datos con este Clasificador no existe un beneficio el hecho de cambiar los símbolos de los emoticonos por texto. Para el conjunto de datos C2 se obtiene la mejor precisión de 0.828 utilizando el 90 % de documentos de entrenamiento, mismo valor que el mejor resultado en el preprocesamiento 1.

Preprocesamiento 3: Tweets con Texto sin Emoticonos sin Signos de Puntuación

A continuación procedemos con el tercer tipo de preprocesamiento sobre los conjuntos de datos C1 y C2, obteniendo los resultados que se muestran en la tabla 4.37. Los mejores resultados se obtuvieron con el total de características en el vocabulario para ambos conjuntos de datos. El mejor desempeño del Clasificador en este experimento se observó con el 90 % de documentos utilizados para el entrenamiento. En ambos conjuntos el comportamiento de desempeño es en forma creciente, para el conjunto C1 se obtuvo el mejor valor de

Tabla 4.36: Precisión del Clasificador MaxEnt, promedio de 50 ejecuciones con el tipo de preprocesamiento 2 en los conjuntos de datos C1 y C2.

% Documentos en el entrenamiento	***% de Selección de características***	
	Conjunto C1 sc=90	**Conjunto C2 sc=100**
10	0.629	0.698
20	0.676	0.735
30	0.697	0.757
40	0.712	0.777
50	0.728	0.788
60	0.735	0.801
70	0.74	0.81
80	0.755	0.821
90	**0.755**	**0.828**

precisión de 0.767 mientras que para C2 fue de 0.834. Con este tipo de preprocesamiento, en el conjunto de datos C1 se obtuvo una mejora del 0.3 % y de 1.2 % respecto de los tipos de preprocesamiento 1 y 2 respectivamente. Mientras que con el conjunto C2 se obtuvo una mejora de 0.6 % respecto de los dos preprocesamientos anteriores.

Tabla 4.37: Precisión del Clasificador MaxEnt, promedio de 50 ejecuciones con el tipo de preprocesamiento 3 en los conjuntos de datos C1 y C2.

% Documentos en el entrenamiento	***% de Selección de características***	
	Conjunto C1 sc=100	**Conjunto C2 sc=100**
10	0.629	0.7
20	0.672	0.737
30	0.694	0.761
40	0.711	0.778
50	0.727	0.793
60	0.738	0.803
70	0.747	0.817
80	0.747	0.822
90	**0.767**	**0.834**

Preprocesamiento 4: Tweets con Texto en Minúsculas sin Emoticonos

Al someter los datos de los conjuntos C1 y C2 al tipo de preprocesamiento 4 y observar el efecto sobre el comportamiento del Clasificador MaxEnt, se tiene que los resultados

para el conjunto C1 son superiores en un 0.3 % y 0.6 % para los tipos de preprocesamiento 1 y 2 respectivamente e inferior en un 0.7 % respecto del tipo de preprocesamiento 3. Para el conjunto C2 se obtuvo el mismo valor de precisión que el obtenido en los tipos de preprocesamiento 1 y 2 y se es inferior en un 0.6 % respecto del tipo de preprocesamiento 3. Los mejores resultados se obtuvieron con el total de características en el vocabulario y en el 90 % de cantidad de documentos utilizados para el entrenamiento.

Tabla 4.38: Precisión del Clasificador MaxEnt, promedio de 50 ejecuciones con el tipo de preprocesamiento 4 en los conjuntos de datos C1 y C2.

% Documentos en el entrenamiento	***% de Selección de características***	
	Conjunto C1 sc=100	**Conjunto C2 sc=100**
10	0.628	0.7
20	0.667	0.736
30	0.697	0.76
40	0.72	0.777
50	0.731	0.788
60	0.741	0.799
70	0.745	0.809
80	0.755	0.823
90	**0.76**	**0.828**

Preprocesamiento 5: Tweets con Texto en Minúsculas sin Emoticonos sin Signos de Puntuación

Por ultimo se realiza el experimento con el tipo de preprocesamiento 5 en los dos conjuntos de datos, en la tabla 4.39 se muestran los mejores valores de precisión que se observaron según la cantidad de características seleccionadas, siendo el total de características en el vocabulario en donde el Clasificador MaxEnt tuvo el mejor desempeño. Para el conjunto C1 se tiene el mejor valor de precisión de 0.761 y para el conjunto C2 se obtuvo el valor de 0.835, en ambos conjuntos el comportamiento es creciente y por lo tanto se tienen que los mejores resultados son en el 90 % de documentos utilizados para el entrenamiento. Respecto al desempeño con los tipos de preprocesamiento anteriores, en el conjunto C1 se incremento el mejor resultado en un 0.6 % y 0.1 % en los tipos de preprocesamiento 2 y 4 respectivamente, se disminuyo en un 0.2 % y 0.6 % en los tipos de preprocesamiento 1

y 3 respectivamente; para el caso del conjunto C2 se incremento en un 0.7 % en los tipos de preprocesamiento 1, 2 y 4 y en un 0.1 % respecto del tipo de preprocesamiento 5, es decir, para el conjunto C2 el mejor desempeño del Clasificador se observa con este tipo de preprocesamiento.

Tabla 4.39: Precisión del Clasificador MaxEnt, promedio de 50 ejecuciones con el tipo de preprocesamiento 5 en los conjuntos de datos C1 y C2.

% Documentos en el entrenamiento	***% de Selección de características***	
	Conjunto C1 sc=100	**Conjunto C2 sc=100**
10	0.623	0.702
20	0.671	0.737
30	0.695	0.761
40	0.71	0.776
50	0.724	0.792
60	0.736	0.804
70	0.742	0.812
80	0.751	0.824
90	**0.761**	**0.835**

En la figura 4.15 se muestra el comportamiento del Clasificador para cada uno de los 5 tipos de preprocesamiento de datos sobre el conjunto C1 que se implementaron en este experimento, considerando los mejores resultados observados por el procedimiento de selección de características en cada uno de ellos, predominando el total de características en el vocabulario (sc=100 %). En general el comportamiento del Clasificador es creciente en los 5 tipos de preprocesamiento de datos y el mejor desempeño se observa en la cantidad mayor de documentos utilizados para el entrenamiento. El tipo de preprocesamiento 3 tiene el mejor valor de precisión. No existe una diferencia notable de mejora con algún tipo de preprocesamiento de datos con este Clasificador y para el conjunto de datos C1.

En la figura 4.16 se muestran los resultados del comportamiento de desempeño del Clasificador en los diferentes tipos de preprocesamiento de los datos sobre el conjunto C2. El comportamiento es creciente en todos los casos, sobresaliendo el tipo de preprocesamiento 3 ligeramente sobre los otros hasta del 20 % al 70 % de documentos utilizados para el entrenamiento. El mejor resultado de precisión promedio se observó en el tipo de preprocesamiento 5 cuando son considerados el 90 % de documentos de entrenamiento.

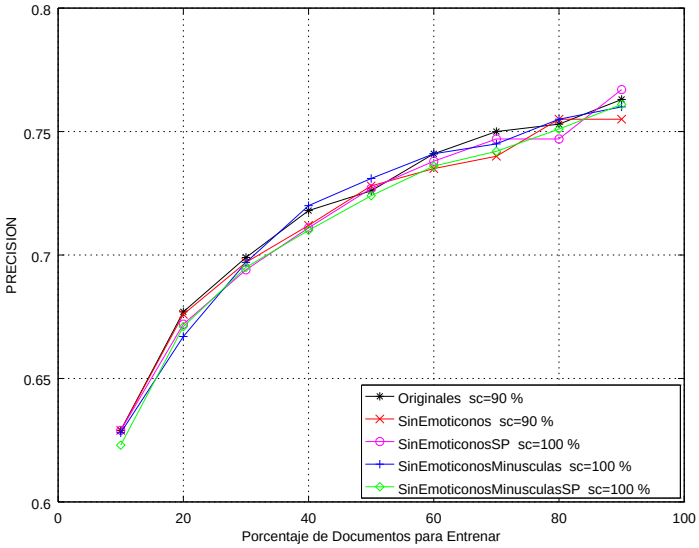


Figura 4.15: Comportamiento de desempeño del Clasificador MaxEnt para cada uno de los 5 tipos de preprocesamiento de datos en el conjunto C1. *sc* se refiere a la cantidad de características seleccionadas.

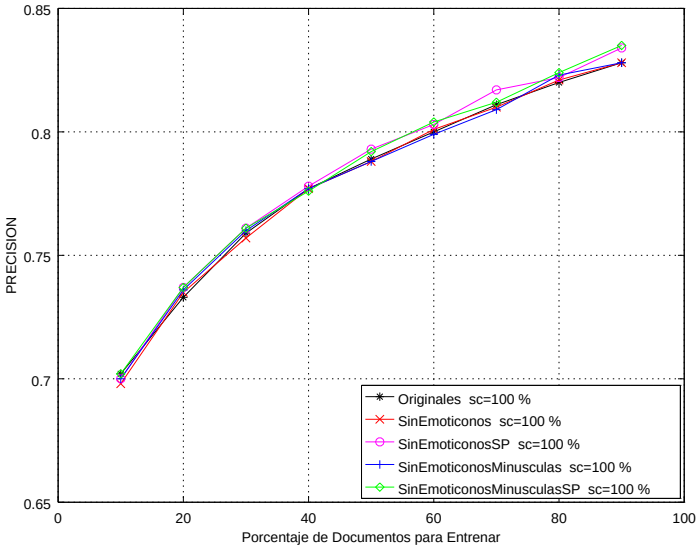


Figura 4.16: Comportamiento de desempeño del Clasificador MaxEnt para cada uno de los 5 tipos de preprocesamiento de datos en el conjunto C2. *sc* se refiere a la cantidad de características seleccionadas.

4.6. Resumen General

Los resultados de los experimentos realizados con los cuatro 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 Emoticonos; 3) Tweets con Texto sin Emoticonos sin Signos de Puntuación; 4) Tweets con Texto en Minúsculas sin Emoticonos; y 5) Tweets con Texto en Minúsculas sin Emoticonos sin Signos de Puntuación.* La tabla 4.40 nos muestra el resumen general para el conjunto de datos C1, los mejores resultados se observan en los tipos de preprocesamiento de los datos 2, 3 y 5. Respecto a la cantidad de características seleccionadas, sólo en algunos casos se observó que con el 90 % de las mismas se puede obtener el mejor resultados, pero en su mayoría (14 de 20 de los mejores resultados) se requiere del total de características en el vocabulario para alcanzar los mejores resultados.

Tabla 4.40: Resumen general en cuanto a la efectividad de los diferentes Clasificadores con respecto a los 5 preprocesamientos realizados (Conjunto C1).

Clasificador	**Tipo de Preprocesamiento**									
	1		**2**		**3**		**4**		**5**	
	E	**sc**	*E*	**sc**	*E*	**sc**	*E*	**sc**	*E*	**sc**
NB	0.781	100	0.778	100	0.782	100	0.775	90	**0.789**	100
KNN	0.723	100	**0.726**	100	0.719	100	0.719	100	0.717	100
SVM	0.775	90	**0.777**	90	**0.777**	90	0.777	100	0.775	100
MaxEnt	0.763	90	0.755	90	**0.767**	100	0.760	100	0.761	100

EL resumen general para el conjunto C2 se muestra en la tabla 4.41, en ella podemos ver que los mejores resultados se encuentran en los tipos de preprocesamiento 3, 4 y 5. Respecto a la selección de características se tiene que se logró un mejor resultado con el Clasificador NB con el 80 % de características, la menor cantidad de características en donde se observó un valor máximo en la precisión de un Clasificador. EL resto de mejores resultados requirieron en total de características en el vocabulario y en lo general 13 de 20 mejores resultados se lograron con el total de características.

Con al menos tres Clasificadores que obtuvieron sus mejores resultados en el tipo

Tabla 4.41: Resumen general en cuanto a la efectividad de los diferentes Clasificadores con respecto a los 5 preprocesamientos realizados (Conjunto C2).

Clasificadores	**Tipo de Preprocesamiento**									
	1		**2**		**3**		**4**		**5**	
	E	**sc**	*E*	**sc**	*E*	**sc**	*E*	**sc**	*E*	**sc**
NB	0.852	90	0.853	90	**0.860**	80	0.852	90	0.858	90
KNN	0.779	100	0.777	100	0.775	100	**0.779**	100	0.777	100
SVM	0.870	100	0.872	90	0.872	90	0.869	100	**0.877**	100
MaxEnt	0.828	100	0.828	100	0.834	100	0.828	100	**0.835**	100

de preprocesamiento 5 de los datos, se puede concluir con esta evidencia que si se tiene una mejora en la exactitud del Clasificador al aplicarle dicho tipo de preprocesamiento.

Otro de los resultados de los experimentos es observar las diferencias de desempeño de los cuatro Clasificadores, para con ello verificar el que obtiene la mejor eficiencia respecto de los otros.

Tabla 4.42: Diferencias de la mejor efectividad de los Clasificadores en los 5 tipos de preprocesamiento de los datos en el Conjunto C1.

Clasificadores	**Tipo de Preprocesamiento**					
	1	**2**	**3**	**4**	**5**	**Promedio (%)**
NB-KNN	5.8	5.2	6.3	5.6	7.2	6.02
NB-SVM	0.6	0.1	0.5	2	3.4	1.32
NB-MaxEnt	1.8	2.3	1.5	1.5	2.8	1.98
SVM-KNN	5.2	5.1	5.8	5.8	3.8	5.14
SVM-MaxEnt	1.2	2.2	1	1.7	0.6	1.34
MaxEnt-KNN	4	2.9	4.8	4.1	4.4	4.04

La tabla 4.42 nos muestra las diferencias de los mejores resultados obtenidos en cada uno de los tipos de preprocesamiento de los datos en el conjunto C1, tomando un Clasificador respecto del resto. Por ejemplo, si tomamos el NB Multinomial vs KNN para el preprocesamiento 1, se puede ver que NB Multinomial supera en eficiencia a KNN a un 5.8 %, mientras que NB Multinomial vs SVM en este mismo caso, se puede ver que NB Multinomial supera en eficiencia un 0.6 % y a MaxEnt en un 1.8 %. Así de esta forma podemos ver las comparaciones de todos los algoritmos. Finalmente se obtiene el valor promedio de la diferencia de mejores exactitudes entre los algoritmos para los 5 tipos de

preprocesamiento. Para el conjunto C1 el mejor desempeño es para NB Multinomial, seguido por SVM, MaxEnt y al final KNN.

Tabla 4.43: Diferencias de la mejor efectividad de los Clasificadores en los 5 tipos de preprocesamiento de los datos en el Conjunto C2. El valor negativo en NB-SVM indica que es mejor SVM que NB.

Clasificadores	**Tipo de Preprocesamiento**					
	1	**2**	**3**	**4**	**5**	**Promedio (%)**
NB-KNN	7.3	7.6	8.5	7.3	8.1	7.76
NB-SVM	-1.8	-1.9	-1.2	-1.7	-1.9	-1.7
NB-MaxEnt	2.4	2.5	2.6	2.4	2.3	2.44
SVM-KNN	9.1	9.5	9.7	9	10	9.46
SVM-MaxEnt	4.2	4.4	3.8	4.1	4.2	4.14
MaxEnt-KNN	4.9	5.1	5.9	4.9	4.9	5.14

En la tabla 4.43 se muestran las diferencias de los mejores resultados observados para los diferentes Clasificadores en los 5 tipos de preprocesamiento utilizando el conjunto de datos C2. El valor negativo en NB-SVM se refiere a que SVM obtuvo un mejor desempeño que el NB en este conjunto de datos. Con estas diferencias, SVM es el que mejor desempeño mostró bajo los diferentes tipos de preprocesamiento de los datos, seguido del Clasificador SVM, MaxEnt y finalmente KNN.

Con las evidencias presentadas, decidir que Clasificador utilizar en un sistema de clasificación de texto, nos lleva a proponer los Clasificadores NB y SVM por su mejor desempeño mostrado en los conjuntos de datos utilizados en el presente trabajo.

Capítulo 5

Conclusiones y trabajos futuros

5.1. Conclusiones

En la creación de sistemas de minería de opinión, se tiene la necesidad de decidir el tipo de algoritmo de clasificación de texto a utilizar. En este trabajo se presenta un conjunto de experimentos que muestran evidencia empírica del desempeño, que bajo diferentes tipos de preprocesamiento fueron sometidos dos conjuntos de datos, para cuatro de los Clasificadores más populares del área de aprendizaje de máquina.

La evidencia empírica nos permite concluir en que NB y SVM son los dos Clasificadores que se recomiendan utilizar en algún sistema de clasificación de texto, por que presentaron el mejor comportamiento en la tarea de clasificación de documentos obtenidos del Twitter.

Una mejora significativa en el desempeño de los Clasificadores se observó con el preprocesamiento 5 de los datos, en los Clasificadores de mejor desempeño (NB y SVM). El tipo de preprocesamiento se refiere a utilizar solamente letras minúsculas, cambiar los símbolos de los emoticonos por palabras de texto representativas del sentimiento y eliminar los signos de puntuación.

En relación a la selección de características, el método χ^2 no influyó en incrementar el desempeño de los Clasificadores, debido a que la evidencia nos mostró que con el total de características en el vocabulario, se obtuvieron los mejores resultados de desempeño de

los Clasificadores.

5.2. Trabajos futuros

Como trabajo futuro se propone el uso de la representación vectorial de las palabras en la minería de opinión. Por su reciente auge, utilizar técnicas de Procesamiento del Lenguaje Natural, que permitan involucrar técnicas que nos lleven a obtener información del análisis sintáctico, semántico u otra índole en los conjuntos de datos.

Proponer técnicas o implementar alguna que permita identificar el sarcasmo en las frases contenidas en los documentos de texto.

Comparar con técnicas que involucren redes neuronales artificiales y que se han aplicado a la minería de texto.

De acuerdo a la experimentación y la utilización de estos contenidos para la creación de un corpus en español u recursos es un punto fundamental para el avance de la investigación en procesamiento automático de opiniones.

Realizar una evaluación mas exhaustiva con otros algoritmos de minería de opinión como BBR (Regresión Logística Bayesiana) entre otros.

Apéndice A

Código fuente de los clasificadores

1. **Código fuente del algoritmo que guarda y lee el Modelo del Clasificar Naive Bayes Multinomial, además se obtiene la efectividad y selección de características.**

```
#!/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 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 import metrics
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.naive_bayes import MultinomialNB
posFeatures = []
negFeatures = []
posTrain = []
negTrain = []
posTest = []
negTest = []

#Extraigo los Tweets positivos y negativos de los archivos
archivoPos = open('Positivos2Sort.txt','r')
leeLineaPos = archivoPos.readlines()
archivoPos.close()

for listaPos in leeLineaPos:
```

```
    posFeatures.append(listaPos)

archivoNeg = open('Negativos2Sort.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
entero=int(sys.argv[1])  # el primer argumento indica el porcentaje de datos para entrenamiento

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

#se hace la selecci n de caracter sticas
dimensiones=X_train.shape
numCarac=int(sys.argv[2]) # el segundo argumento indica el porcentaje de caracter sticas a considerar
if numCarac < 0:
    ch2 = SelectKBest(chi2,k='all')    # k indica el n mero de caracter sticas seleccionadas
else:
    numCarac2 = int(math.floor(dimensiones[1]*float(numCarac/10.0)))
    ch2 = SelectKBest(chi2,numCarac2)    # k indica el n mero de caracter sticas seleccionadas

Xnew_train = ch2.fit_transform(X_train, Y_train)

#Se crea el vector X_test para los datos de prueba
```

```
X_test = vectorizer.transform(f for f in testFeatures)
Xnew_test = ch2.transform(X_test)

#Para el clasificador Miltinomial NB
nb = MultinomialNB()
#Se guarda los mmodelos para poder utilizarlos posteriormente
with open('modelotweetsNBM.pkl','wb') as fid:
                cPickle.dump(nb,fid)
with open('mtweetsNBMVectorizerTFIDF.pkl','wb') as fid:
                cPickle.dump(vectorizer,fid)
#Se entrena el Modelo
nb.fit(Xnew_train, Y_train)
#Se hace la predicci n con el nuevo vector de prueba X_new_test
pred = nb.predict(Xnew_test)
#Se calcula la precici n
score = metrics.accuracy_score(Y_test, pred)
print  numCarac2,score
```

2. **Código fuente del algoritmo que guarda y lee el Modelo del Clasificar KNN a si mismo obtiene la efectividad y selección de características.**

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
# -*- coding: 850 -*-
import re, math, collections, itertools, os, sys
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 sklearn.feature_selection import SelectKBest, chi2

#Se crean los arreglos
posFeatures = []
negFeatures = []
posTrain = []
negTrain = []
posTest = []
negTest = []


#Extraigo los Tweets positivos y negativos de los archivos
archivoPos = open('Positivos2Sort.txt','r')
leeLineaPos = archivoPos.readlines()
archivoPos.close()

for listaPos in leeLineaPos:
    posFeatures.append(listaPos)
```

```
archivoNeg = open('Negativos2Sort.txt','r')
leeLineaNeg = archivoNeg.readlines()
archivoNeg.close()

for listaNeg in leeLineaNeg:
    negFeatures.append(listaNeg)

random.shuffle(posFeatures)
random.shuffle(negFeatures)

#Asigno un porcentaje de los Tweets para entrenamiento y el resto para prueba
entero=int(sys.argv[1])
posCutoff = int(math.floor(len(posFeatures)*entero/10))
negCutoff = int(math.floor(len(negFeatures)*entero/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_train para los datos de entrenamiento
Y_train = posTrain + negTrain
#Se crea el vector Y_test para los datos de prueba
Y_test = posTest + negTest

#se hace la selecci n de caracter sticas
dimensiones=X_train.shape
numCarac=int(sys.argv[2]) # el segundo argumento indica cuantas caracter sticas a considerar
if numCarac < 0:
        ch2 = SelectKBest(chi2,k='all')    # k indica el n mero de caracter sticas seleccionadas
else:
        numCarac2 = int(math.floor(dimensiones[1]*float(numCarac/10.0)))
        ch2 = SelectKBest(chi2,numCarac2)    # k indica el n mero de caracter sticas seleccionadas
Xnew_train = ch2.fit_transform(X_train, Y_train)

# Para el clasificador Knn
neigh = KNeighborsClassifier(n_neighbors=7)

neigh.fit(Xnew_train, Y_train)
```

```
# Se guarda los modelos aprendidos para poder utilizarlos posteriormente
with open('modelotweetsKNN.pkl','wb') as fid:
    cPickle.dump(neigh,fid)
with open('mtweetsKNNVectorizerTFIDF.pkl','wb') as fid:
    cPickle.dump(vectorizer,fid)


#Se crea el vector X_test para los datos de Prueba
X_test = vectorizer.transform(f for f in testFeatures)
Xnew_test = ch2.transform(X_test)
pred = neigh.predict(Xnew_test)

#Se calcula la Precicion
score = metrics.accuracy_score(Y_test, pred)
print  numCarac2, score
```

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

```
#!/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 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
from sklearn.feature_selection import SelectKBest, chi2

posFeatures = []
negFeatures = []
posTrain = []
negTrain = []
posTest = []
negTest = []
#num=sys.argv[1]
#numInt=int(num)

#Extraigo los Tweets positivos y negativos de los archivos
archivoPos = open('Positivos2Sort.txt','r')
leeLineaPos = archivoPos.readlines()
archivoPos.close()

for listaPos in leeLineaPos:
    posFeatures.append(listaPos)
```

```
archivoNeg = open('Negativos2Sort.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
entero=int(sys.argv[1])  # el primer argumento indica el porcentaje de datos para entrenamiento

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

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

Y_test = posTest + negTest

#se hace la selecci n de caracter sticas
numCarac=int(sys.argv[2]) # el segundo argumento indica el porcentaje de caracter sticas a considerar
if numCarac < 0:
    ch2 = SelectKBest(chi2,k='all')     # k indica el n mero de caracter sticas seleccionadas
else:
    numCarac2 = int(math.floor(dimensiones[1]*float(numCarac/10.0)))
    ch2 = SelectKBest(chi2,numCarac2)     # k indica el n mero de caracter sticas seleccionadas
Xnew_train = ch2.fit_transform(X_train, Y_train)
```

```
X_test = vectorizer.transform(f for f in testFeatures)
Xnew_test = ch2.transform(X_test)
#Para el clasificador SVM
SVM = SVC()

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

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

pred = SVM.predict(Xnew_test)

#Se calcula la efectividad
score = metrics.accuracy_score(Y_test, pred)
print  numCarac2,score
```

4. **Código fuente del algoritmo que guarda y lee el Modelo del Clasificar MAXENT a si mismo obtiene la efectividad y selección de características.**

```
#!/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 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 import metrics
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.linear_model import LogisticRegression

posFeatures = []
negFeatures = []
posTrain = []
negTrain = []
posTest = []
negTest = []

#Extraigo los Tweets positivos y negativos de los archivos
archivoPos = open('Positivos2Sort.txt','r')
leeLineaPos = archivoPos.readlines()
archivoPos.close()

for listaPos in leeLineaPos:
    posFeatures.append(listaPos)
```

```
archivoNeg = open('Negativos2Sort.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
entero=int(sys.argv[1])  # el primer argumento indica el porcentaje de datos para entrenamiento

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

#se hace la selecci n de caracter sticas
dimensiones=X_train.shape
numCarac=int(sys.argv[2]) # el segundo argumento indica el porcentaje de caracter sticas a considerar
if numCarac < 0:
    ch2 = SelectKBest(chi2,k='all')    # k indica el n mero de caracter sticas seleccionadas
else:
    numCarac2 = int(math.floor(dimensiones[1]*float(numCarac/10.0)))
    ch2 = SelectKBest(chi2,numCarac2)    # k indica el n mero de caracter sticas seleccionadas
Xnew_train = ch2.fit_transform(X_train, Y_train)

#Se crea el vector X_test para los datos de prueba
X_test = vectorizer.transform(f for f in testFeatures)
Xnew_test = ch2.transform(X_test)
```

```
#Para el clasificador Logistica Regresion
logreg = LogisticRegression()
# Se guarda los modelos aprendidos para poder utilizarlos posteriormente
with open('modelotweetsMAXENT.pkl','wb') as fid:
    cPickle.dump(logreg,fid)
with open('mtweetsMAXENTVectorizerTFIDF.pkl','wb') as fid:
    cPickle.dump(vectorizer,fid)
#Se entrena el Modelo
logreg.fit(Xnew_train, Y_train)
#Se hace la predicci n con el nuevo vector de prueba X_new_test
pred = logreg.predict(Xnew_test)

#Se calcula la precici n
score = metrics.accuracy_score(Y_test, pred)
print  numCarac2,score
```

5. **Ejemplo de algunos Tweets que contiene el archivo Positivos2Sort.txt y el archivo Negativos2Sort.txt, que son utilizados con los cuatro Algoritmos de Clasificación de texto NBM, KNN, SVM y MaxEnt.**

    a) Positivos2Sort.txt

```
27 pelicula romantica favorita
28 pelicula de accion que mas te guste
30 agosto lasexta estrena la pelicula     el     chico de filadelfia     httpstco6hyr1dfr65
30 agosto lasexta estrena la pelicula     el     chico de filadelfia     httpstco6klfwcz6tv
5 de los animales m s astutos del mundo mundo httpstcorpgbjp8rni
5 de los animales m s astutos del mundo mundo httpstcorpgbjp8rni
6 productos 57 premios muchas gracias lomejornosedetiene betterneverstops httpstcolfiixrvxl8
```

    b) Negativos2Sort.txt

```
con la neblina que hay en el pasamayo queda preciso para filmar una pelicula de terror
con tanto papeleo de las narices para la universidad estoy medio desaparecida pero sin el medio
copa am rica si messi no llega gait n ser  su reemplazante en tn m vil httpstco9ijtaqmd0b
corran pinches politicos mal gobierno
corrupto y ladr n httpstco49brtsm68r
corrupto y ladr n httpstco49brtsm68r
```

Apéndice B

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>

<title>Login</title>
</head>

<body BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#0000FF" VLINK="#800080" ALINK="#FF0000"
BACKGROUND="Wallpaper.jpg" LINK=>

<center><b><a HREF="http://www.umich.mx/"><img src="Umich.png"
alt="visita aulaclic" width="140" height="150" border="0"></a>
<font size=5.9 color="Olive"> Universidad Michoacana De San Nicol&aacute;s
De Hidalgo</font><a HREF="http://www.fie.umich.mx/" ><img src="Fie.gif"
alt="visita aulaclic" width="170" height="130" border="0"></a> </b></center>


<center><b> <font size=5 color="Olive">Facultad De Ingenier&iacute;
a El&eacute;ctrica </font></b></center>

<center><b><a HREF="http://www.umich.mx/licenciatura-ingenieria-computacion.html">
<font size=5 color="Olive">Ingenier&iacute;a En Computaci&oacute;n
</font></a></b></center>
<br></br>
<center><b><font size=4.7 color=White>Sistema de Predicci&oacute;n
de la Polaridad en un Texto </font></b></center>


<center>
        <form action="conexion.php" method="POST" >
```

```
        <p><font size=4 color=WHITE> Ingresa Frase a Analizar:
        </font><br><textarea name="analizar" rows="2" cols="50">
        </textarea> </p>
        <p><font size=4 color=WHITE> Selecciona Clasificador (es) a utilizar:</font><br>
        <input name="bayes" type="checkbox" /><font size=4 color=WHITE >NB</font>
        <input name="knn" type="checkbox" /><font size=4 color=WHITE >Knn</font>
        <input name="svm" type="checkbox" /><font size=4 color=WHITE >SVM</font>
        <input name="maxent" type="checkbox" /><font size=4 color=WHITE >MAXENT</font>
        <br><br>
        <button>
                <b><font size=4 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();
        $salidaMAXENT = array();
        $predice = array();
        $predice1 = NULL;
        $predice = NULL;
        $predice =$_POST['analizar'];
        $predice1 =$_POST['analizar'];

        echo "<b><font size=4 color=WHITE> Texto Ingresado: </font></b>";
        #echo "<br>";
        echo "<font size=4 color=WHITE>$predice</font>";
        echo "<br>";
        echo "<b><font size=3 color=WHITE> Polaridad Predecida por los Clasificadores: </font></b>";

        if(isset($_POST['bayes'])){
                $comando="python cBayesPredic.py \"$predice\"";
                #$output = exec("python cBayesPredic.py '".$predice."'",$salidaNB);
                $output=exec($comando,$salidaNB);
                echo "<br>";
                echo "<b><font size=4 color=WHITE>(NB): </font></b>";
                echo "<font size=3 color=WHITE>$salidaNB[0]</font>";
        }

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

```
                $comando="python KnnLoadPag.py \"$predice\"";
                #$output = exec("python KnnClasificadorLoadModel.py '".$predice."'",$salidaKnn);
                $output=exec($comando,$salidaKnn);
                echo "<br>";
                echo "<b><font size=4 color=WHITE>(Knn): </font></b>";
                echo "<font size=3 color=WHITE>$salidaKnn[0]</font>";
        }

        if(isset($_POST['svm'])){
                $comando="python svmLoadPag.py \"$predice\"";
                #$output = exec("python svmLoadPag.py '".$predice1."'",$salidaSVM);
                $output=exec($comando,$salidaSVM);
                echo "<br>";
                echo "<b><font size=4 color=WHITE>(SVM): </font></b>";
                echo "<font size=3 color=WHITE>$salidaSVM[0]</font>";
        }

        if(isset($_POST['maxent'])){
                $comando="python maxentLoadPag.py \"$predice\"";
                #$output = exec("python svmLoadPag.py '".$predice1."'",$salidaSVM);
                $output=exec($comando,$salidaMAXENT);
                echo "<br>";
                echo "<b><font size=4 color=WHITE>(MAXENT): </font></b>";
                echo "<font size=3 color=WHITE>$salidaMAXENT[0]</font>";
        }

?>
</center>
</body>
</html>
```

Apéndice C

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 recibe 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()
```

7. Código fuente para reemplazar los emoticonos por texto.

```
#-*-coding:utf-8-*-

nuevoArchivo=open("PositivosSC2.txt","w")
archivo=open("Positivos2Sort.txt","r")

reemplaza={'      ':'felizsonriente','      ':'aprobacion','      ':'burla',
'      ':'lagrimas de alegria','      ':'sonriente','      ':'ojos sonrientes',
'      ':'feliz','      ':'sonriente sudor frio','      ':'feliz sonriente',
'      ':'feliz sonriente','      ':'gui o','      ':'feliz sonriente',
'      ':'ligeramente sonriente','      ':'boca arriba','      ':'soriente',
'      ':'saboreando comida deliciosa','      ':'aliviada','      ':'enamorado','      ':'beso',
'      ':'besando','      ':'besando','      ':'besando con ojos cerrados','      ':'gui o',
'      ':'lengua fuera y ojos cerrados','      ':'lengua fuera','      ':'boca suelta',
'      ':'nerd','      ':'feliz sonriente','      ':'sombrero de vaquero','      ':'abrazo',
'      ':'payaso','      ':'burlona','      ':'sinboca','      ':'neutra','      ':'sin expresion',
'      ':'aburrida','      ':'ojos girando','      ':'pensativo','      ':'mentiroso',
'      ':'sonrojada','      ':'decepcionada','      ':'preocupacion','      ':'enfadada',
'      ':'enojada','      ':'pensativa','      ':'confundida','      ':'fruncido',
'      ':'fruncido','      ':'perseverancia','      ':'confundida','      ':'cansada',
'      ':'cansada','      ':'triunfal','      ':'sorprendido','      ':'gritando de miedo',
'      ':'temerosa','      ':'sudor frio','      ':'silenciosa','      ':'fruncido',
'      ':'angustiada','      ':'llorando','      ':'decepcion','      ':'somnolienta',
'      ':'babaen la boca','      ':'sudor frio','      ':'llorando a gritos',
```

```
'      ':'mareada','      ':'atonita','      ':'cirre en la boca','      ':'mascara medica',
'      ':'termometro','      ':'cabeza vendada','      ':'nausea','      ':'estornudos',
'      ':'durmiendo','      ':'dormir','      ':'celebrando','      ':'aplaudiendo',
'      ':'saludando','      ':'todo bien','      ':'mal','      ':'se aldepu o',
'   ':'pu oalzado','      ':'pu ohacialaizquierda','      ':'pu o hacia la derecha',
'      ':'victoria','      ':'dedos cruzados','      ':'ok','   ':'mano alzada',
'      ':'mano alzada','      ':'manos abiertas','      ':'apreton de manos',
'      ':'biceps flexionado','      ':'rezando','      ':'entrar en calor',
'      ':'silueta llenamano','      ':'opinion negativa',
'      ':'se al hacia la izquierda','      ':'se al hacia la derecha',
'      ':'provocacion','      ':'espera','      ':'llamame','      ':'cuernos',
'      ':'extra ar','#':''}

forlineainarchivo.readlines():
    forx,yinreemplaza.items():
            linea=linea.replace(x,y)
    nuevoArchivo.write(linea)
nuevoArchivo.close()
```

En la siguiente imagen se muestra el código para reemplazar los emoticonos (caritas) por texto.

```
#-*-coding:utf-8-*-

    nuevoArchivo=open("PositivosSC2.txt","w")
    archivo=open("Positivos2Sort.txt","r")

    reemplaza={'😀':'😃','😁':'aprobacion','😂':'burla',
    '😭':'lagrimas de alegria','😄':'sonriente','😆':'ojos sonrientes',
    '😅':'feliz','😓':'sonriente sudor frio','😇':'feliz sonriente',
    '☺':'feliz sonriente','😉':'guiño','😊':'feliz sonriente',
    '😌':'ligeramente sonriente','🙃':'boca arriba','😍':'soriente',
    '😋':'saboreando comida deliciosa','😏':'aliviada','😘':'enamorado','😚':'beso',
    '😗':'besando','😗':'besando','😙':'besando con ojos cerrados','😎':'guiño',
    '😜':'lengua fuera y ojos cerrados','😝':'lengua fuera','😛':'boca suelta',
    '🤓':'nerd','🤗':'feliz sonriente','🤠':'sombrero de vaquero','🤗':'abrazo',
    '😐':'payaso','😒':'burlona','😶':'sinboca','😦':'neutra','😑':'sin expresion',
    '😒':'aburrida','🙄':'ojos girando','🤔':'pensativo','😯':'mentiroso',
    '😳':'sonrojada','😵':'decepcionada','😕':'preocupacion','😠':'enfadada',
    '😡':'enojada','😧':'pensativa','😖':'confundida','😣':'fruncido',
    '😫':'fruncido','😩':'perseverancia','😕':'confundida','😴':'cansada',
    '😲':'extrañar','#':''}

    forlineainarchivo.readlines():
        forx,yinreemplaza.items():
            linea=linea.replace(x,y)
        nuevoArchivo.write(linea)
    nuevoArchivo.close()
```

Figura C.1: Código para reemplazar los emoticonos por texto.

Apéndice D

Construcción del Conjunto de Datos: Corpus

D.1. Introducción

Los corpus son hoy en día uno de los recursos básicos para el estudio y la descripción de las lenguas. En particular, en el ámbito de la *Lingüística Computacional*, se han erigido como punto de partida imprescindible para la elaboración de léxicos y gramáticas, y representan una línea de investigación transversal en lo que al tratamiento con medios informáticos del lenguaje se refiere, al ser indispensables para el desarrollo de aplicaciones basadas tanto en el texto como en el habla [Llisterri96].

A un conjunto de datos se le denomina *corpus* en un sentido general del término. Pero ha sido el empleo de las computadoras para reunir, organizar y procesar esos datos el que ha dotado de modernidad a esta tarea, hasta el punto de propiciar el despegue de toda una forma de hacer lingüística, la llamada *lingüística de corpus* [Llamazares08].

Un área básica de actuación de la lingüística computacional y cuyo desarrollo es requisito básico para que las demás aplicaciones computacionales que se diseñen para el procesamiento del lenguaje natural sean útiles, es precisamente la del estudio del uso y la estructura lingüística, lo que denomina [Guinovart98] *lingüística informática* y cuyo máximo exponente es la *lingüística de corpus* entendida como el estudio empírico de la

lengua a partir de los datos que proporciona el análisis de ejemplos reales de producciones lingüísticas (orales o escritas) almacenadas en una computadora [Pérez Hernández09].

D.1.1. Conjunto de datos en un sistema de aprendizaje

En los sistemas de aprendizaje de maquina se hacen necesarios los corpus en dos sentidos: 1) para que el sistema generalice o aprenda los datos; y 2) para probar la eficacia con que se aprendieron los datos. Esto lleva a establecer al menos dos tipos de conjuntos de datos: ***1) el conjunto de entrenamiento; y 2) 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 la figura D.1).

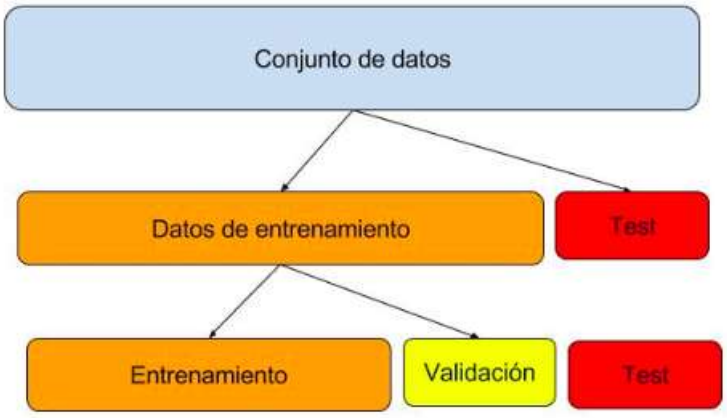


Figura D.1: Particionamiento típico del conjunto de datos [Cambronero06].

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 D.2).

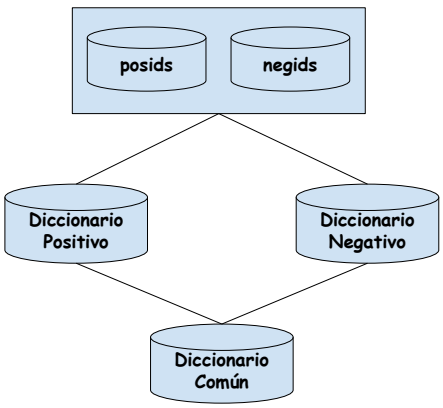


Figura D.2: Del conjunto de Tweets obtenidos, se deben etiquetar en dos clases: positivos y negativos. Se obtiene un diccionario común para representar vectorialmente los documentos o en forma individual, dependiendo del sistema de clasificación.

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.

D.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.

D.2.1. API de Twitter: Tweepy

Python es un gran lenguaje para todo tipo de cosas. Existe una comunidad activa de desarrolladores que crean 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 D.3 muestra un ejemplo de cómo acceder a la API de Twitter usando Tweepy con OAuth:

```
1  import tweepy
2  from tweepy.auth import OAuthHandler
3  #from twitter import Twitter, OAuth
4
5  # Consumer keys and acces tokens, used for OAuth
6  consumer_key = 'z7yCVwYHgdIGFHTuiEwEzsTDs'
7  consumer_secret = 'q6C583TD252UfLvbDGjkkLhVfcqvZwx0gCEL1uCk3H4aTL6mXQ'
8  access_token = '720650116059631616Z4y9E2cjAre9mOnORwOSqynbSOkrNLJ'
9  access_token_secret = 'arkW9HfsUhcVIUAHYiggA5LOdZB6nubZvsICgOLNikwS7'
10
11 # OAuth process, using the keys and tokens
12 auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
13 auth.set_access_token(access_token, access_token_secret)
14
15 #Creation of the actual interface, using authentication
16 api = tweepy.API(auth)
17 api.update_status("Hola Twitter desde mi Terminal!!")
```

Figura D.3: Ejemplo de cómo acceder a la API de Twitter usando Tweepy con OAuth.

El resultado del código de la imagen D.3 se muestra en la imagen D.4:

Figura D.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.

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 D.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 D.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 D.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”).

D.3. Análisis de Sentimientos con *Meaning Mloud*

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

```
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 D.6: Clases y funciones que se utilizaron.

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.

D.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.

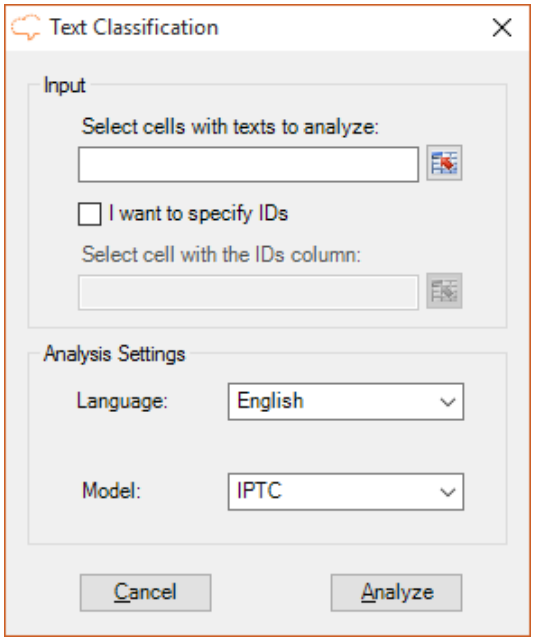


Figura D.7: Esta es la interfaz que aparece cuando se hace clic en el botón de clasificación de textos.

Como se puede ver en la figura D.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 D.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 D.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 D.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 defecto, como se muestra en la figura D.9.

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.

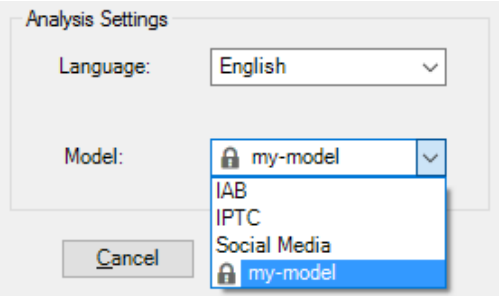


Figura D.8: Configuración de análisis para seleccionar el Idioma y el Modelo para clasificar los textos.

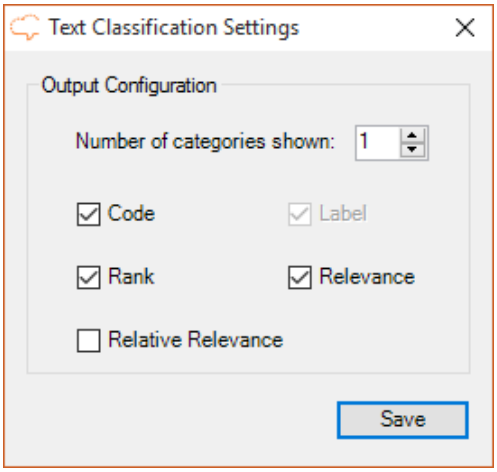


Figura D.9: Ajustes de clasificación de texto.

- **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 D.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 D.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.

Referencias

[Bader58] Bader, R. S. Similarity and recency of common ancestry. *Systematic Biology*, 7(4):184–187, 1958.

[Baldi03] Baldi, P., Frasconi, P., y Smyth, P. Modeling and understanding human behavior on the web. *Modeling the Internet and the Web: Probabilistic Methods and Algorithms*, págs. 171–209, 2003.

[Beresi04] Beresi, U. C., Adeva, J. G., Calvo, R., y Ceccatto, A. Automatic classification of news articles in spanish. *En Actas del Congreso Argentino de Ciencias de Computación*. 2004.

[Bordignon04] Bordignon, F., Peri, J., Tolosa, G., Villa, D., y Paoletti, L. Experimentos en clasificación automática de noticias en español utilizando el modelo bayesiano [en línea], 2004.

[Cámara11] Cámara, E. M., Valdivia, M. M., y Urena, L. A. Análisis de sentimientos. *IV Jornadas TIMM Tratamiento de la Información Multilingüe y Multimodal 7 y 8 de abril de 2011*, pág. 61, 2011.

[Cambronero06] Cambronero, C. G. y Moreno, I. G. Algoritmos de aprendizaje: knn & kmeans. *Intelgencia en Redes de Comunicación, Universidad Carlos III de Madrid*, 2006.

[Carlos00] Carlos, G. F. L. La investigación sobre recuperación de la información en español. 2000.

[Cortes95] Cortes, C. y Vapnik, V. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[daCruz11] da Cruz, G., Velozo, T., y Elvas Falcão Soares, A. Twitter, youtube e innovación en la promoción turística online: Análisis de las estrategias del ministerio de turismo de brasil. *Estudios y perspectivas en turismo*, 20(3):627–642, 2011.

[Dice45] Dice, L. R. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[Duda73] Duda, R. O., Hart, P. E., et al. *Pattern classification and scene analysis*, tomo 3. Wiley New York, 1973.

[Eíto Brun04] Eíto Brun, R. y Senso, J. A. Minería textual. *El profesional de la información*, 13(1), 2004.

[Esquivel16] Esquivel, R. B. S. Análisis de sentimiento en textos de twitter en español: Evaluación de clasificadores de texto, apr 2016.

[Eugenio11] Eugenio, M. C., Valdivia, M., Teresa, M., Perea Ortega, J. M., y Ureña López, L. A. Técnicas de clasificación de opiniones aplicadas a un corpus en español. 2011.

[Guinovart98] Guinovart, J. G. Fundamentos de lingüística computacional: bases teóricas, líneas de investigación y aplicaciones. *Bibliodoc: anuari de biblioteconomia, documentació i informació*, págs. 135–146, 1998.

[Hair99] Hair, J. F. y Suárez, M. G. *Análisis multivariante*, tomo 491. Prentice Hall Madrid, 1999.

[Han11] Han, J., Kamber, M., y Pei, J. *Data mining: concepts and techniques*. Elsevier, 2011.

[Harman92] Harman, D. Relevance feedback and other query modification techniques., 1992.

[Hsu03] Hsu, C.-W., Chang, C.-C., Lin, C.-J., et al. A practical guide to support vector classification. 2003.

[Jurafsky00] Jurafsky, D. y Martin, J. H. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition. 2000.

[Landauer02] Landauer, T. K. On the computational basis of learning and cognition: Arguments from lsa. *Psychology of learning and motivation*, 41:43–84, 2002.

[Liu12] Liu, B. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[Llamazares08] Llamazares, M. V. Lingüística con corpus (i). *Estudios Humanísticos. Filología*, (30):329–349, 2008.

[Llisterri96] Llisterri, J. y Moure, T. Lenguaje y nuevas tecnologías: el campo de la lingüística computacional. págs. 147–228, 1996.

[Manning99] Manning, C. D. y Schütze, H. *Foundations of statistical natural language processing*, tomo 999. MIT Press, 1999.

[Mehra02] Mehra, N., Khandelwal, S., y Patel, P. Sentiment identification using maximum entropy analysis of movie reviews. Inf. téc., Working paper, 2002.

[Molina04] Molina, J. y García, J. Técnicas de análisis de datos en aplicaciones prácticas utilizando microsoft excel y weka [en línea], 2004.

[Morales09] Morales, E. Descubrimiento de conocimiento en bases de datos. *Obtenido el día*, 11, 2009.

[Pereira González15] Pereira González, A. Selección de características para el reconocimiento de patrones con datos de alta dimensionalidad en fusión nuclear. 2015.

[Pérez Hernández09] Pérez Hernández, C. y Ortiz, A. M. Lingüística computacional y lingüística de corpus. potencialidades para la investigación textual. *Teoría y literatura artística en la sociedad digital: construcción y aplicabilidad de colecciones textuales informatizadas*, págs. 67–96, 2009.

[Rogers60] Rogers, D. J., Tanimoto, T. T., et al. A computer program for classifying plants. *Science (Washington)*, 132:1115–18, 1960.

[Salton86] Salton, G. y McGill, M. J. Introduction to modern information retrieval. 1986.

[Salton88] Salton, G. y Buckley, C. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[Sharma96] Sharma, S. Applied multivariate techniques, new york, john willey & sons, 1996.

[Suárez14] Suárez, E. J. C. Tutorial sobre máquinas de vectores soporte (svm). *Tutorial sobre Máquinas de Vectores Soporte (SVM)*, 2014.

[Téllez05] Téllez, A. Extracción de información con algoritmos de clasificación. *Extracción de información con algoritmos de clasificación*, 2005.

[Tzoukermann03] Tzoukermann, E., Klavans, J. L., y Strzalkowski, T. Information retrieval. *The Oxford handbook of computational linguistics*, págs. 530–544, 2003.

[Venegas07] Venegas, R. Clasificación de textos académicos en función de su contenido léxico-semántico. *Revista signos*, 40(63):239–271, 2007.

[Vilares13] Vilares, D., Alonso, M. A., y Gómez-Rodríguez, C. Clasificación de polaridad en textos con opiniones en espanol mediante análisis sintáctico de dependencias. *Procesamiento del lenguaje natural*, 50:13–20, 2013.

[Zazo02] Zazo, A., Figuerola, C., Alonso, J., y Gómez, R. Recuperación de información utilizando el modelo vectorial. *Recuperación de información utilizando el modelo vectorial*, 2002.