UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

MÁQUINAS DE SOPORTE VECTORIAL PARA LA ORIENTACIÓN SEMÁNTICA DE OPINIONES EN ESPAÑOL

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

Licenciado en Ciencias de la Computación

PRESENTA:

Alonso Palomino Garibay

TUTORA

Sofía N. Galicia-Haro

Datos del jurado

1. **Datos del alumno:**
   Palomino
   Garibay
   Alonso
   Universidad Nacional Autónoma de México
   Facultad de Ciencias
   Ciencias de la Computación

2. **Datos del tutor:**
   Dra.
   Galicia
   Haro
   Sofía Natalia

3. **Datos del sinodal 1:**
   Dr.
   Ivan Vladimir
   Meza
   Ruiz

4. **Datos del sinodal 2:**
   Dr.
   Gustavo
   De la Cruz
   Martínez

5. **Datos del sinodal 3:**
   Lic.
   Sergio
   Hernández
   López

6. **Datos del sinodal 4:**
   Dr.
   Gibran
   Fuentes
   Pineda

7. **Datos del trabajo escrito:**
   Máquinas de soporte vectorial para la orientación semántica de opiniones en español.
   111 pp.
   Ciudad Universitaria, 18 de Mayo del 2016

"I, myself, have had many failures and I've learned that if you are not failing a lot, you are probably not being as creative as you could be, you aren't stretching your imagination enough."

- John Warner Backus

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Resumen

Facultad de Ciencias

Departamento de Matemáticas

Licenciatura en Ciencias de la Computación

Máquinas de soporte vectorial para la orientación semántica de opiniones en español

por Alonso Palomino Garibay

Existe un auge de medios de comunicación sociales que ha permitido que en sitios como Twitter, Yelp, Facebook o sitios de ventas como Amazon e Ebay, los usuarios compartan sus comentarios acerca de muchos temas. Desde diversas perspectivas: comerciales, industriales, sociales, entre otras, se considera que obtener comentarios específicos de esos medios sociales y saber qué contienen ayudaría para distintos propósitos. En particular, desde una perspectiva comercial, es útil conocer qué tipo de opinión tienen los usuarios sobre un producto determinado. Por otro lado, desde un punto de vista social es importante conocer qué opinan los votantes de cierto candidato electoral.

Debido a esta necesidad, surgió el desarrollo de sistemas computacionales que procesan directamente el texto procedente de la gran cantidad de comentarios que abundan en internet, para intentar obtener información útil, como por ejemplo: determinar si una opinión con respecto a una entidad en particular es favorable o desfavorable.

El presente trabajo tiene como objetivo utilizar un algoritmo supervisado de aprendizaje automático: máquinas de soporte vectorial, para obtener un clasificador automático que estime un puntaje que va del uno al cinco, tarea que es muy similar a la que realizan los usuarios al evaluar productos comerciales en sitios de ventas por internet. La metodología propuesta utiliza patrones lingüísticos obtenidos a partir de una colección de textos opiniones sobre un producto electrodoméstico, en este caso lavadoras. Posteriormente, estos patrones lingüísticos, sirven para entrenar al algoritmo que a su vez aprenderá automáticamente. Finalmente, podrá categorizar en un rango de una a cinco estrellas nuevas instancias de opiniones nunca antes vistas.

Agradecimientos

Quiero expresar mi enorme gratitud a la Dra. Sofía N. Galicia-Haro por la gran paciencia que me tuvo a la hora de haber tomado este trabajo.

Al Dr. Iván Vladimir Meza-Ruiz le ofrezco mi especial agradecimiento por su apoyo, ser gran mentor y la valiosa oportunidad que me dio de participar en el workshop de Author Profiling en PAN 2015, qué es de las mejores experiencias que me llevo durante mi paso por la UNAM. Sin duda me ayudó a crecer intelectualmente y a plantearme nuevas preguntas.

Le agradezco al Dr. Gustavo de la Cruz Martínez por su apoyo y su excelente curso de Inteligencia Artificial. Sin duda otra buena experiencia que me ayudó a la realización de este trabajo mediante solidos fundamentos en el área.

Me gustaría agradecerle Dr. Gibran Fuentes Pineda por los comentarios y revisiones que sin duda me ayudaron a corregir y mejorar este trabajo.

Al Lic. Sergio Hernández López le agradezco por su curso de análisis numérico y de aprendizaje automático, que fue muy importante para mi formación. De la misma forma le agradezco sus comentarios y correcciones.

Finalmente a mis asesores Sofía N. Galicia-Haro, Iván V. Meza-Ruiz, Gustavo de la Cruz Martínez, Gibran Fuentes Pineda, Sergio Hernández López y María Guadalupe Elena Ibargüengoitia González les agradezco su apoyo durante mi difícil y largo proceso para iniciar estudios de posgrado.

Le agradezco a Francisco por ser un excelente padre y por apoyarme cuando lo he necesitado, también a mis abuelos y a Patricia por cuidar de mí.

A mis amigos Adolfo Camacho-Gonzáles, Alejandro Soriano, José Luis Álvarez y José Galindo por hacer mi paso por la universidad más agradable.

Índice general

Índice de figuras

Índice de tablas

Gracias Francisco por creer en mí.

1 Introducción

El análisis de sentimiento o también llamado minería de opiniones, es la tarea de encontrar y analizar computacionalmente opiniones sobre entidades específicas; por ejemplo, productos comerciales (teléfonos, electrodomésticos, entre otros), candidatos electorales, servicios (por ejemplo: hoteles, restaurantes, entre otros). Por lo que recientemente en las ciencias de la computación, el análisis de sentimiento es una de las áreas en las que se ha venido desarrollando una gran cantidad de investigación [Feldman, 2013]. Feldman indica que una gran cantidad de empresas de software desarrollan soluciones a ese problema y que software estadístico como SAS[1] y SPSS[2] incluyen módulos para el análisis de sentimiento.

En [Liu, 2012], se reporta que existen distintos nombres y tareas con diferencias sutiles en este campo de estudio (por ejemplo: análisis de sentimiento, minería de opiniones, extracción de opiniones, minería de sentimientos, análisis de subjetividad, análisis de afecto, análisis de emociones, etc.) debido a que distintas comunidades han llamado y abordado al problema que se plantea en este trabajo de diversas formas. El investigador Liu Bin plantea que todos ellos se encuentran bajo el campo del análisis de sentimiento o minería de opiniones, es por eso que para este trabajo usaremos el termino análisis de sentimiento y minería de opiniones de forma indistinta. Cabe mencionar que en el ámbito comercial, el nombre de análisis de sentimiento es más común mientras que en la academia, tanto minería de opiniones como análisis de sentimiento se usan indistintamente.

Las redes sociales como Twitter y Facebook son un punto focal de muchas aplicaciones del análisis de sentimiento. La aplicación más común es monitorear la reputación de

[1] SAS Sentiment Analysis
[2] SPSS Text Analytics for Surveys

una marca específica, naturalmente esto puede extenderse a otros dominios, por ejemplo campañas políticas, cuotas de pantalla para una emisora o de un programa televisivo. En [Feldman, 2013], el autor menciona que una aplicación que realizaba análisis de sentimiento en tiempo real a partir de tuits fue Tweetfeel[3]. Otro ejemplo que se indica en [Feldman, 2013] es que la minería de opiniones puede ofrecer un gran valor sustancial a los candidatos a diversos cargos políticos, permitiéndole a los administradores u organizadores de campañas políticas realizar un seguimiento de cómo los votantes se sienten acerca de diferentes temas y cómo se relacionan con los discursos y acciones de algún candidato político. Otro dominio importante para el análisis de sentimiento son los mercados financieros. Feldman indica que hay una gran cantidad de noticias, artículos, blogs y tuits sobre distintas empresas o instituciones gubernamentales y que un sistema de minería de opiniones puede utilizar distintas fuentes de información para saber la percepción y lo que se escribe o dice sobre estas entidades así como agregarles el sentimiento en forma de una puntuación que pueda utilizarse posteriormente.

Respecto a las opiniones sobre productos, en [Liu, 2012] se consideró estudiar reseñas porque están focalizadas y llenas de opiniones. Liu menciona que conceptualmente no hay diferencias con las redes sociales, noticias y foros de discusión pero que hay diferencias en el grado de dificultad para tratar con unos u otros. Por ejemplo, los tuits son muy cortos e informales, con emoticones, jerga, ligas a otros sitios web y abreviaturas. Otra característica que hace difícil el problema de derivar la orientación semántica en opiniones es el dominio en el que se quiere aplicar. Es decir, no es lo mismo procesar reseñas de películas, que procesar reseñas de opiniones de electrodomésticos también contribuye en la dificultad de su tratamiento. Los foros de discusión con temas sociales y políticos son más difíciles por sus temas complejos y expresiones de sentimiento, sarcasmo e ironías. Las opiniones sobre productos y servicios son generalmente más fáciles de analizar, aunque sigue siendo un reto su análisis.

Por el volumen de las opiniones sobre productos y servicios, resulta difícil su manejo para un análisis manual, ya que requiere mucho tiempo escoger y comparar opiniones sobre entidades específicas. Adicionalmente, dentro de las opiniones, puede haber oraciones que expresen tanto juicios positivos como negativos sobre características del producto, lo que hace más complicada su utilización.

Una de las principales tareas en la minería de opiniones sobre productos comerciales consiste en determinar la polaridad de las opiniones tomando en cuenta el documento completo, oraciones, o características específicas. Para posteriormente clasificarlas en

[3] TweetFeel: Real-Time Sentiment Search

positivas o negativas, aunque otros rangos de valoración se han considerado, por ejemplo: muy positivas, positivas, neutras, negativas y muy negativas.

La orientación semántica se refiere a si un documento completo es positivo o negativo. Una palabra que transmite un sentimiento puede tener orientaciones semánticas contrarias en diferentes dominios. Por ejemplo, *"una cámara súper lenta"* puede ser útil para reproducir vídeos sin perder detalles, por lo tanto el enunciado anterior puede tener una orientación semántica positiva, en cambio una *"lavadora súper lenta"* podría tener una falla y así tener una orientación semántica negativa.

1.1. Objetivo

Este trabajo presenta una aproximación para resolver el problema de derivar la orientación semántica de opiniones sobre productos comerciales, escritas en español, mediante un método automático de clasificación supervisado.

1.2. Definición del problema

Para resolver el problema de derivar la orientación semántica en opiniones en primera instancia se necesita contar con comentarios específicos, por lo que primero habría que buscarlos en las diversas fuentes o plataformas de medios sociales en internet. En segunda instancia tendríamos que conocer el contenido de dicha información, para saber qué contienen sería necesario analizar e interpretar las opiniones y almacenar esta información en formatos adecuados. Finalmente, se aplicaría un método que aprenda de esta información previamente almacenada y evaluarla de la misma forma en que lo haría un usuario.

En este trabajo utilizamos una colección de opiniones acerca de lavadoras, anteriormente extraída de Internet que cuenta con errores gramaticales, ortográficos y de puntuación, más detalles pueden encontrarse en la sección 3.3.1. Por lo que nos concentramos en saber de alguna manera qué contienen los comentarios y que es lo que caracteriza a cada comentario. Seguimos los métodos típicamente empleados para minar opiniones, utilizamos palabras, secuencias de palabras y oraciones de los documentos que portan una opinión o un sentimiento. Para clasificar las opiniones de acuerdo a los niveles de valoración que asignan los usuarios, los métodos que actualmente se han empleado

corresponden a los métodos de aprendizaje automático en inteligencia artificial. Aplicamos el método de máquinas de soporte vectorial que permite separar las instancias de opiniones a clasificar mediante hiperplanos, con la posibilidad de extenderse a cambios de espacios de mayor dimensión mediante funciones. En este método la selección de características es el primer problema a resolver, que corresponde a la selección de palabras, secuencias de palabras u oraciones de los textos.

1.3. Objetivos específicos

Los objetivos particulares que delimitamos para este trabajo fueron:

a) Determinación de características que modelan sentimiento en opiniones, de acuerdo a los principales trabajos desarrollados en el área de análisis de sentimiento o minería de opiniones.

b) Exploración y experimentación de una biblioteca de aprendizaje automático que implemente el modelo de máquinas de soporte vectorial.

c) Definición de características de negación para mejorar la clasificación inicial.

d) Evaluación del rendimiento del modelo y de las características propuestas para determinar la orientación semántica de las opiniones.

Para lograr estos objetivos seguimos los siguientes pasos:

a) Revisión bibliográfica sobre minería de opiniones, sobre derivación de la orientación semántica en inglés y en español, y sobre la negación en español

b) Obtención de características para el método de aprendizaje.

c) Selección de una herramienta de aprendizaje automático que incluyera las máquinas de soporte vectorial.

d) Experimentación con los parámetros de la herramienta para clasificación multiclase y para diferentes funciones kernel.

e) Obtención de características de negación para el método de aprendizaje.

f) Experimentación con las características de negación y evaluación de los resultados.

1.4. Estructura de este trabajo

En el capítulo dos presentamos los antecedentes teóricos para este trabajo. Primero damos una introducción al problema de minar o analizar opiniones computacionalmente, e indicamos los trabajos ya realizados para el idioma español. En segundo lugar presentamos las distintas subtareas relacionadas a dicho problema y proveemos algunas definiciones generales usadas en este contexto. Continuamos presentando la teoría básica de los métodos de aprendizaje automático y su clasificación con la finalidad de dar una intuición. Finalmente, introducimos algunos fundamentos teóricos de las máquinas de soporte vectorial.

En el capítulo tres presentamos el trabajo relacionado a derivar la orientación semántica en el cual nos basamos para una propuesta inicial. En particular citamos trabajos que hacen uso de algoritmos con métodos no supervisados y supervisados. En seguida especificamos los recursos necesarios para derivar la orientación semántica en este trabajo. Finalmente presentamos las ideas que seguimos para obtener características de negación en español.

En el capítulo cuatro presentamos la herramienta utilizada para realizar los experimentos. En particular mencionamos las características más importantes de las herramientas que utilizamos, como pandas, NumPy y scikit-learn. Especificamos distintas características que hacen al proyecto scikit-learn una alternativa interesante para hacer tareas de aprendizaje automático. Incluimos las facilidades de la herramienta para realizar la evaluación.

En el capítulo cinco, presentamos los resultados obtenidos. Especificamos de forma detallada cada uno de los experimentos realizados, también se describe el proceso de como se fueron añadiendo las características definidas para generar distintos corpus que solo conservaban las características consideradas. De la misma forma mencionamos los parámetros utilizados y presentamos un análisis de los resultados, contrastamos nuestros resultados con otros trabajos desarrollados para el idioma español.

En el capitulo 6, presentamos un resumen general del trabajo, conclusiones y direcciones investigación futuras.

2 La minería de opiniones o el análisis de sentimiento

A continuación presentamos trabajo que consideramos relevante en el contexto del análisis de sentimiento o también llamado minería de opiniones por algunas otras comunidades para el idioma ingles, así como el estado del arte en esta área. Revisamos algunos de los antecedentes más importantes para el idioma español, damos algunos ejemplos y definiciones que sirven para construir la metodología que usamos para derivar la orientación semántica del corpus usado en este trabajo.

2.1. Contexto

En el trabajo de [Liu, 2012], el autor indica que el término análisis de sentimiento aparece probablemente por primera vez en [Nasukawa & Yi, 2003], en donde se propone una metodología para el análisis de sentimiento asociado con polaridades positivas o negativas, de temas específicos en un documento. En lugar de clasificar todo el documento como positivo o negativo. En el trabajo de Nakusawa y Yi se hace énfasis en que, para mejorar la precisión del análisis de sentimiento se necesitan identificar de forma adecuada las relaciones semánticas entre las expresiones que denotan sentimiento y el tema. En dicho trabajo, Nakusawa y Yi realizaron análisis semántico con un un analizador semántico y un lexicón de sentimiento, es decir, es una lista de palabras o frases que expresan sentimiento positivo o negativo. El sistema propuesto por Nakusawa y Yi alcanza una precisión que va del 75 % al 95 % en encontrar el sentimiento a partir de paginas de internet y artículos de noticias.

Liu indica que la expresión: *"minería de opiniones"* aparece por primera vez en [Dave *et al.*, 2003]. En este trabajo los autores identificaron las particularidades del problema

de minar opiniones y desarrollaron técnicas basadas en recuperación de información para la extracción de las características más relevantes en opiniones. A éstas se les asoció una calificación o puntaje, con base a los resultados de varios tipos de métricas y heurísticas los autores concluyeron que los datos tenían variaciones según la situación específica cada muestra, asimismo concluyeron que los mejores métodos funcionaron igual o mejor a los métodos tradicionales de aprendizaje automático, ya que al trabajar con enunciados individuales recolectados de distintas fuentes de internet, el rendimiento se ve limitado debido al ruido y la ambigüedad, inherentes al lenguaje natural.

A partir del año 2000 comenzó el auge en la investigación de minería de opiniones y sentimientos con el trabajo de [Hatzivassiloglou & Wiebe, 2000], en el cual se destacó que la subjetividad es habitual en el lenguaje humano y que las características a nivel enunciado son importantes para el desarrollo aplicaciones relacionadas al procesamiento de textos, por ejemplo, en áreas como recuperación de información y extracción de información. Los autores, estudiaron los efectos de adjetivos dinámicos, adjetivos semánticamente orientados y adjetivos graduables en con un clasificador de subjetividad simple. Concluyeron que este tipo de adjetivos son de gran utilidad para predecir la subjetividad. Un novedoso método que estadísticamente combina dos indicadores de graduabilidad fue presentado y evaluado.

Más adelante [Turney, 2002] presenta un algoritmo de entrenamiento no supervisado para la clasificación de reseñas recomendadas y no recomendadas. En donde la clasificación de una reseña se estima por el promedio de la orientación semántica de las frases de una reseña, compuesta por adjetivos o adverbios. Ejemplo de esto es que *diferencias muy sutiles* tienen una orientación semántica positiva, debido a que claramente cuenta con palabras que tienen una orientación semantica positiva. Mientras que *muy arrogante* al tener malas asociaciones o palabras negativas presenta una orientación semántica negativa. Su algoritmo logró una precisión del 74 % cuando se evaluó en 410 reseñas de portal del internet *Epinions*, cabe mencionar que el autor tomó opiniones en el contexto de cuatro dominios diferentes: reseñas de automóviles, bancos, películas y destinos turísticos.

En [Morinaga *et al.*, 2002] se presenta otra metodología para minar la reputación de productos en internet en la que automáticamente se recolectan las opiniones de personas en páginas de internet sobre productos específicos. Este trabajo resulta importante debido a que llegaron a buenos resultados mediante el uso de reglas lingüísticas y sintácticas para determinar si cada declaración de texto era o no una opinión, además de saber si una opinión era positiva o negativa.

En [Pang *et al.*, 2002], consideraron el problema de clasificar documentos no por el tema, sino por el sentimiento total de los documentos. Determinaron si una reseña era positiva o negativa usando reseñas de películas como datos. Encontraron que las técnicas tradicionales de aprendizaje automático definitivamente superaron a sistemas que funcionaban manualmente. Usaron tres algoritmos distintos para clasificar (i.e. Naive Bayes, clasificación por máxima entropía y máquinas de soporte vectorial) solo que estos no tuvieron tan buen desempeño en la clasificación del sentimiento.

Por otro lado en [Pang & Lee, 2008], los autores mencionan que como respuesta directa al interés en desarrollar nuevos sistemas que se enfrentan directamente a opiniones como datos de entrada, se dio un repentino interés en la investigación de minería de opiniones y análisis de sentimiento. Nuevamente en el trabajo antes mencionado, se formalizaron las técnicas y metodologías que permiten buscar información en forma de opiniones. Se enfocaron en métodos que permiten a los sistemas ser conscientes del sentimiento en texto; e incluyeron, generación automática de resúmenes sobre las opiniones para concluir con un análisis de las implicaciones sociales y el impacto económico que podría tener el desarrollo de este novedoso tipo de sistemas de información.

Otro trabajo que cabe destacar es el de [Rangel *et al.*, 2014], en el que se da a la tarea de estimar la frecuencia de uso de palabras para denotar un tipo de emoción especifico (e.g. alegría, miedo, tristeza, sorpresa y disgusto). Los autores mencionan que a diferencia de otros diccionarios, el diccionario que proponen contiene el porcentaje de probabilidad en que una palabra es usada con sentido emocional.

2.2. Análisis de sentimiento, el problema

El análisis de sentimiento o minería de opiniones, es una técnica que usa diferentes métodos para el estudio computacional de opiniones en texto que expresan opiniones sobre alguna entidad. A continuación presentamos la definición del problema considerando el marco de referencia definido por [Liu, 2012].

Definición 2.1. Análisis de Sentimiento (o Minería de Opiniones): Campo de estudio interdisciplinario entre la lingüística computacional, procesamiento de lenguaje natural, extracción y recuperación de información y minería de textos que identifica, extrae y analiza lenguaje natural en texto, como lo son las opiniones de personas, sentimientos, evaluaciones, apreciaciones, actitudes y emociones sobre entidades como: productos,

servicios, organizaciones, individuos, temas, eventos, discusiones. A partir de distintos tipos de fuentes como redes sociales, blogs, periódicos, etc.

Definición 2.2. Opinión: Una opinión es una quíntupla $Op = (e_i, a_{\text{ij}}, s_{\text{ijkl}}, h_k, t_l)$, donde:

1. e_i es el nombre de una entidad.
2. a_{ij} es un aspecto de e_i.
3. s_{ijkl} es el sentimiento de un aspecto. a_{ij} de la entidad e_i.
4. h_k es el autor de la opinión.
5. t_l es el tiempo cuando h_k expresa la opinión.

El sentimiento s_{ijkl} es positivo, negativo, neutral o expresado con diferente nivel de fuerza o intensidad, por ejemplo, una escala numérica discreta. Cuando una opinión es sobre la misma entidad como un todo, el aspecto especial GENERAL se utiliza para indicarlo. e_i y a_{ij} juntos representan la opinión objetivo.

Definición 2.3. Entidad: Una entidad e es un producto, servicio, tema, problema, persona, evento u organización. Descrito por una pareja: $e : (T, W)$. Donde T es la jerarquía de las partes, sub-partes y así sucesivamente. Mientras que W es el conjunto de atributos de e, donde cada parte o sub-partes pueden tener su propio conjunto de atributos.

2.2.1. Objetivos del análisis de sentimiento

El principal objetivo de este trabajo es clasificar la orientación semántica sobre un documento en su totalidad. A continuación presentamos la definición del objetivo del análisis de sentimiento basados en el trabajo de [Liu, 2012] así como las tareas que se encarga de resolver este subcampo de estudio.

Definición 2.4. Objetivo del Análisis de Sentimiento: Dada una opinión-documento d, descubrir todas las quíntuplas $Op = (e_i, a_{\text{ij}}, s_{\text{ijkl}}, h_k, t_l)$ [Liu, 2012].

De la definición 2.4, Liu derivó seis distintos problemas que el análisis de sentimiento busca solucionar. Es decir, dados un conjunto de opiniones-documentos D, el análisis de sentimiento o minería de opiniones se encarga de resolver 6 tareas:

1. **Extracción de entidad y categorización:**
   Consiste en extraer todas las expresiones sobre una entidad en D y categorizar o agrupar en expresiones sinónimas. Cada grupo o grupo de expresiones indica una única entidad e_i.

2. **Extracción de Aspecto y Categorización:**
   Se basa en extraer todas las expresiones sobre aspectos de entidades y categorizar estas expresiones de aspectos en grupos o categorías. En otras palabras, cada grupo de expresiones de aspecto de una entidad e_i representa un único aspecto a_{ij}.

3. **Extracción del sustentante de la opinión y categorización:** Esta tarea radica en extraer al autor de la opinión en el texto o en datos estructurados y se encarga de categorizarlo.

4. **Extracción de tiempo y estandarización:**
   Extrae los tiempos en que se dan las opiniones y estandarizan diferentes formatos de tiempo.

5. **Clasificación de sentimiento sobre un aspecto:**
   Determinar si una opinión sobre un aspecto a_{ij} es positivo, negativo o neutral o asignar una calificación numérica de sentimiento al aspecto.

6. **Generación de una quíntupla de opinión:**
   Producir todas las quíntuplas $Op = (e_i, a_{\text{ij}}, s_{\text{ijkl}}, h_k, t_l)$ expresada en el documento d basándose en los resultados de las tareas anteriores.

En [Liu, 2012] se presenta un ejemplo para cada uno de los objetivos de los cuales el análisis de sentimiento se ocupa de resolver. Notemos que las oraciones están numeradas y fueron extraídas a partir de un blog.

Ejemplo 2.1. *Escrito por bigJohn Fecha: Sept. 15, 2011.*

(1) Compré una cámara Samsung y el día de ayer mis amigos compraron una cámara Canon. (2) La semana pasada, ambos usamos mucho nuestras cámaras. (3) Las fotos de mi Samy no son tan buenas y la duración de la batería es corta. (4) Mi amigo estaba muy contento con su cámara y ama su calidad de imagen. (5) Quiero una cámara que pueda tomar buenas fotos. (6) Voy a regresarla el día de mañana.

Para la tarea 1 (extracción de entidad y categorización): se deberían extraer y agrupar en un cluster Samsung y Samy como: “Samsung”, “Samy” y “Canon”, que representan la misma entidad.

Para la tarea 2 (extracción de aspecto y categorización): es necesario extraer las expresiones de aspecto. Es decir, frases sustantivas, palabras sustantivas, verbos, adjetivos y adverbios. Por ejemplo, extraer *"fotos", "imagen"* y *"duración de la batería"*, para luego agrupar fotos e imagen en un cluster o grupo, debido a que son sinónimos.

Para la tarea 3 (extracción del autor de la opinión y categorización): se debe encontrar al autor del comentario en la oración (3) y el amigo del autor del comentario anterior como autor de oración (4).

Para la tarea 4 (extracción de tiempo y estandarización): se tiene que encontrar la fecha o momento en el que el blog fue escrito, es decir 15/09/2011.

La tarea 5 (clasificación de sentimiento sobre un aspecto): debería encontrar que (3) denota una opinión negativa a la calidad de la imagen de la cámara Samsung, así como a la duración de su batería. Para la tarea 6 (generación de una quíntupla de opinión) del enunciado (4) se necesitaría saber a qué se refiere con "su camara" y "su". La tarea 6 debería generar las siguiente 4 quíntuplas:

1. (Samsumg, calidad_de_la_imagen, negativo, autor_del_blog, 15092011)
2. (Samsumg,duración_de_la_batería, negativo, autor_del_blog, 15092011)
3. (Canon,GENERAL, positivo, amigo_del_autor_del_blog,15092011)
4. (Canon, calidad_de_la_imagen, positivo, amigo_del_autor_del_blog,15092011)

2.3. Antecedentes en idioma español

Uno de los trabajos que nosotros consideramos de gran importancia para el idioma español sobre la minería de opiniones fue desarrollado por [Martín-Valdivia *et al*., 2013]. En esta investigación, los autores describen un sistema que clasifica la polaridad de textos en idioma español usando el corpus *MuchoCine*. Los autores realizaron un estudio empírico mediante la combinación de algoritmos supervisados y no supervisados usando un corpus de opiniones de películas en conjunto con su corpus paralelo [1] traducido a ingles. Los autores proponen el uso de meta-clasificadores que combinan aprendizaje supervisado y no supervisado para el desarrollo de sistemas de clasificación de

[1] Un corpus paralelo es una colección de textos, en donde cada uno de los textos se traduce a uno o más idiomas en adición al idioma original. El caso más simple es cuando sólo están dos idiomas involucrados: uno de los corpus es una traducción exacta de la otra.

polaridad. Su sistema genera dos modelos individuales que toman como base el corpus de *MuchoCine* junto con su respectiva traducción en ingles. Posteriormente, integraron *SentiWordNet*, un lexicón de sentimiento[2] en el corpus en idioma ingles, generando un nuevo modelo no supervisado. Finalmente, los dos sistemas fueron combinados usando un meta-clasificador que les permitió aplicar distintas combinaciones de algoritmos como por ejemplo un sistema de votación o un sistema de apilamiento.

Otro trabajo que es importante considerar es el de [Vilares *et al.*, 2013], los autores propusieron un algoritmo supervisado, basado en análisis de dependencia sintáctica para determinar la orientación semántica de textos en idioma español, asignando un valor a la orientación de algunas construcciones sintácticas presentes en el corpus. Su metodología consiste en analizar léxicamente el texto para etiquetarlo con marcas de categorías gramaticales para después obtener la estructura sintáctica de las oraciones mediante un analizador de dependencias. Los autores abordaron tres de las más importantes construcciones lingüísticas (intensificación, cláusulas subordinadas adversativas y negación). También propusieron un método de adaptación semi-automático, usando diccionarios semánticos y algoritmos de aprendizaje automático con el fin de adaptar la orientación semántica de las palabras en el corpus a un dominio específico.

Mientras que en [Cruz *et al.*, 2008] los autores compilaron un corpus de alrededor de 4,000 opiniones de películas y aplicaron la metodología supervisada de [Turney, 2002] a un conjunto de reseñas de 200 opiniones negativas y 200 positivas.

En [Liu, 2010] se menciona que claramente las palabras negativas son importantes, debido a que su aparición o incidencia generalmente cambia la orientación semántica de una opinión. Sin embargo, la negación debe manejarse con cuidado por que no todas las ocurrencias de tales palabras significan que una opinión es negativa. Por ejemplo, *"no"* y *"no solo... pero también"* no cambia la dirección de la orientación.

Para el idioma ingles, existe un gran interés en el manejo de la negación para derivar la orientación semántica, ejemplo de esto es que en [Pang *et al.*, 2002] los autores trataron de modelar el efecto de la negación contextual, por ejemplo: *"bien"* y *"no muy bien"* para indicar orientaciones de sentimiento opuestas.

Por otro lado, en [Choi & Cardie, 2008] se presentó un clasificador que trataba a la negación desde un punto de vista composicional, es decir: primero calculaban la polaridad

[2]Los lexicones de sentimientos son listas de palabras o frases de sentimiento asociadas a un porcentaje.

de los términos independientemente y después aplicaron reglas de inferencia para lograr un puntaje de polaridad combinado.

Uno de los trabajos que incluye la negación para derivar la orientación semántica en idioma español es [Vilares *et al*., 2013], en donde los autores propusieron un algoritmo no supervisado basado en análisis de dependencia sintáctica para determinar la orientación de textos, asignando un valor para la orientación semántica de algunas construcciones lingüísticas.

2.4. Niveles de análisis en minería de opiniones

En [Liu, 2012] se indica que el análisis de sentimiento o minería de opiniones se encarga de estudiar principalmente dos tipos de niveles: nivel documento, nivel oración. Pero debido al reciente auge del problema, [Feldman, 2013] menciona que se han desarrollado dos niveles de análisis extra: nivel comparativo, nivel adquisición de lexicón de sentimiento.

2.4.1. Análisis a nivel documento

Existen diversas definiciones de documento por lo cual para fines de este trabajo a continuación consideraremos una interpretación desde el punto de vista del procesamiento de lenguaje natural.

Definición 2.5. Documento: Sea *d* una unidad de componentes léxicos. Un documento *D* es una colección o estructura extendida de unidades sintácticas o lexemas, tales como palabras, grupos y cláusulas, caracterizadas tanto por la coherencia entre sus elementos y la finalización.

En el trabajo de [Liu, 2012] el autor menciona que la clasificación de sentimiento a nivel documento supone que una opinión esta categorizada por un solo sentimiento. Es decir, la orientación semántica se determina considerando la totalidad de la información que contiene dicho documento y de ello dependerá a qué clase pertenece (negativo ó positivo). Este nivel de análisis supone que cada documento expresa opiniones sobre una sola entidad. Por esta razón, no es posible aplicarlo a documentos que evalúan o comparan múltiples entidades. Por ejemplo: Dada una reseña de un producto, a este nivel se determina si la reseña expresa un sentimiento positivo o negativo.

2.4.2. Análisis a nivel oración

Otro nivel de análisis importante es el análisis a nivel oración. Para poder entender mejor este tipo de análisis es útil conocer el concepto de oración.

Definición 2.6. Oración: Una oración es una unidad lingüística que consiste de una o más palabras que están vinculadas gramaticalmente. Una oración puede incluir palabras agrupadas para expresar: declaración, pregunta, exclamación, solicitud, orden, sugerencia, etc.

En este nivel se determina si cada enunciado se expresa como una opinión positiva, negativa ó neutra. Si un enunciado es neutro se da por entendido que no es una opinión. En [Liu, 2012] se menciona que este nivel de análisis esta ligado a la *clasificación de subjetividad*. El cual hace distinción entre enunciados objetivos, que expresan información factual, de los enunciados subjetivos, que expresan opiniones o puntos de vista subjetivos. Según [Liu, 2012] la subjetividad es distinta del sentimiento porque muchos enunciados objetivos pueden implicar subjetividad. Por ejemplo: *"el mes pasado compramos el auto y el limpia parabrisas se le cayó"*.

2.4.3. Análisis a nivel entidad/aspecto

En [Liu, 2012] se realiza un análisis granularizado en este nivel. Este tipo de análisis en lugar de buscar en construcciones lingüísticas como documentos, párrafos, enunciados, cláusulas o frases, explora directamente la opinión en si misma. Es decir, se basa en la idea de que una opinión es constituida a partir de un sentimiento positivo o negativo y un objetivo. Es importante mencionar que el objetivo de las opiniones es de gran utilidad para entender mejor el problema del análisis de sentimiento. Por ejemplo, *"amo este restaurante, aunque el servicio no sea la gran cosa"*.

Aunque la oración: *"amo, este restaurante, aunque el servicio no sea la gran cosa"* claramente tiene tono positivo, pero no podemos decir que esta oración es completamente positiva. De hecho, la oración es positiva acerca del restaurante (enfatizado), pero negativa acerca de su servicio (no se enfatiza). En muchas aplicaciones las opiniones objetivo se describen por entidades y sus diferentes aspectos. Por lo tanto, la meta a este nivel de análisis es descubrir sentimientos a partir de entidades y sus distintos aspectos. Por ejemplo, la oración *"La calidad de llamada del iPhone es buena, pero su duración de*

batería es corta" evalúa dos aspectos, la calidad de la llamada y la duración de la batería del iPhone (entidad). El sentimiento sobre la calidad de las llamadas en el iPhone es positivo, pero el sentimiento sobre su duración de batería es negativo. La calidad de la llamada y la duración de la batería del iPhone son opiniones objetivos.

2.4.4. Análisis a nivel comparativo

En [Feldman, 2013] menciona que en muchos casos los usuarios de algún producto no proveen una opinión directa acerca de una entidad, en lugar de ello dan opiniones comparadas. El objetivo a este nivel de análisis es identificar las oraciones que contienen opiniones de comparación y extraer las entidades prioritarias presentes en cada opinión. Uno de los primeros trabajos que se derivaron de este enfoque puede encontrarse en [Jindal & Liu, 2006]. Por ejemplo: *"Manejé el honda civic, pero no se maneja mejor que un TSX, no están ni siquiera cercanos"*.

2.4.5. Adquisición de lexicón de sentimiento

En [Liu, 2012] se destacó que uno de los indicadores más importantes de sentimiento son las palabras que expresan sentimiento, también llamadas palabras de opinión, éstas se usan comúnmente para expresar sentimientos positivos o negativos. Una lista de tales palabras o frases se llaman un lexicón de sentimientos.

Durante muchos años los investigadores han diseñado una enorme variedad de algoritmos para compilar tales lexicones.

Por ejemplo, en el enunciado: *"el teléfono es poderoso y ligero"*, si contamos con una lista de palabras en donde "poderoso' esté marcado como una palabra positiva, podemos suponer que al utilizarse con el conector *"y"*, la palabra *"ligero"* que a su vez también estaría marcada como una palabra positiva, entonces se puede decir que la opinión es positiva.

Aunque las palabras de sentimiento y frases son importantes para el análisis de sentimiento usarlas únicamente no resuelve el problema. En otras palabras el lexicón de sentimiento es necesario para realizar análisis de sentimiento, pero no suficiente

2.4.6. Procesamiento de lenguaje natural

En [Liu, 2012] se menciona que el nivel de análisis de sentimiento es un problema de procesamiento de lenguaje natural debido a que involucra muchos aspectos generalmente abordados por esta área (e.g. resolución de correferencia, manejo de la negación, desambiguación). De la misma forma, se menciona que todas estas subtareas anteriormente mencionadas añaden más dificultad al problema de analizar sentimiento en texto; dichos problemas aún siguen siendo problemas abiertos. En este sentido cabe mencionar que resulta de gran utilidad saber que el análisis de sentimiento esté restringido a resolver problemas de procesamiento de lenguaje natural, debido a que el sistema no necesita entender la semántica de cada enunciado o documento, solo necesita entender algunos aspectos de cada documento. Por ejemplo, sentimientos positivos o negativos y sus entidades o temas.

2.5. Tipos de opiniones

En [Liu, 2012] se reporta que existen distintos tipos de opiniones: opiniones regulares, opiniones comparativas, opiniones implícitas y opiniones explicitas, debido a que también es posible clasificar las opiniones basándose en lo que se expresa explícitamente o implícitamente en un texto.

1. **Opiniones regulares:**
   Una opinión regular se refiere normalmente como una opinión, tiene dos subtipos de opiniones.

   I Opinión directa:
   Una opinión directa se refiere a una opinión expresada directamente sobre una entidad o aspecto de una entidad. Por ejemplo, *"la calidad de la imagen es genial"*.

   II Opinión indirecta:
   Una opinión indirecta es una opinión que es expresada indirectamente sobre una entidad o aspecto de la entidad basada en sus efectos sobre otras entidades. Por ejemplo, *"despues de inyectarme el medicamento, mis articulaciones se sintieron peor"*.

2. **Opiniones comparativas:**
   Una opinión comparativa expresa una relación, de similitud o diferencia entre dos o más entidades, y/o una preferencia del sustentante de la opinión sobre algún o algunos aspectos compartidos de las entidades. Una opinión comparativa usualmente se expresa en grado comparativo o superlativo de un adjetivo o adverbio. Por ejemplo: *"coca-cola sabe mejor que pepsi"*.

3. **Opiniones explicitas:**
   Una opinión explicita es un enunciado subjetivo que da una opinión regular o comparativa. Por ejemplo, *"coca-cola tiene el mejor sabor"*

4. **Opiniones implícitas:**
   Una opinión implícita es un enunciado objetivo que implica una opinión regular o comparativa. Tal enunciado objetivo usualmente expresa un hecho deseable o no deseable. Por ejemplo, *"La duración de la batería de los teléfonos Nokia es más larga a la de los teléfonos Samsung."*

En particular la colección de opiniones de [Galicia-Haro & Gelbukh, 2014], que fue la utilizada para realizar este trabajo cuenta con opiniones directas por ejemplo:

"Silencio y facilidad. Con estas dos palabras se puede definir este aparato. Además de la multitud de programas que posee para diferentes tejidos..."

2.6. Emoción

En [Liu, 2012] se reporta que las emociones están relacionadas a los sentimientos. La fuerza de un sentimiento u opinión esta típicamente ligada a la intensidad de ciertas emociones como la alegría y la ira. Las opiniones que se estudiaron en este contexto fueron en su mayoría de evaluación. Estas pueden estar categorizadas en dos tipos: evaluaciones racionales y evaluaciones emocionales.

Definición 2.7. Evaluaciones racionales: Son opiniones que consisten en evaluaciones derivadas de argumentos racionales, creencias tangibles y actitudes utilitaristas (Por ejemplo, *"La voz de este teléfono es clara" y "Este coche vale lo que cuesta"*).

Definición 2.8. Evaluaciones emocionales: Son las evaluaciones que son producto de respuestas subjetivas y emocionales sobre entidades que profundizan en el estado

de ánimo de las personas Por ejemplo, *"Estoy muy enojado con el personal de este servicio" y "Este es el mejor coche jamás construido. "*.

Para finalizar, en las secciones antes mencionadas se plantea el problema de minar o analizar a distintos niveles el sentimiento a partir de opiniones, así como los trabajos más significativos en este subcampo, principalmente para el idioma ingles y también para el idioma español. Se presentaron ejemplos y definiciones importantes para entender este problema.

2.7. Aprendizaje automático

En [Mund, 2015], se menciona que el aprendizaje automático explora la construcción de algoritmos que pueden aprender a hacer predicciones mediante un conjunto de datos. Esta clase de algoritmos opera construyendo un modelo a partir de instancias o ejemplos para hacer decisiones o predicciones a partir de los datos de entrada.

A partir de 1980 hubo una revolución en el procesamiento de lenguaje natural con la introducción de algoritmos de aprendizaje automático para procesar de lenguaje humano. Investigaciones recientes se enfocan en el estudio de algoritmos de aprendizaje supervisados y no supervisados. Los algoritmos modernos de procesamiento de lenguaje natural están basados en aprendizaje automático estadístico. Estos algoritmos toman como entrada un conjunto de características o atributos que se generan a partir de los datos de entrada.

La investigación en esta área se enfoca en modelos basados en estadística, que hacen decisiones probabilísticas basadas en ajustar datos en forma de vectores reales. Estos modelos tienen la ventaja de que pueden expresar la certeza relativa de muchas diferentes respuestas posibles en lugar de ofrecer sólo una, con lo cual se producen resultados más fiables, cuando un algoritmo de este tipo se incluye como un componente de un sistema más grande [Wikipedia, 2016a].

Es importante mencionar que el aprendizaje automático está significativamente relacionado al procesamiento de lenguaje natural debido a que a diferencia de los primeros sistemas de procesamiento de lenguaje, que estaban basados en conjuntos de reglas de palabras, en la actualidad los sistemas de procesamiento de lenguaje natural se sustentan en inferencia estadística. Es decir, para automáticamente aprender, un algoritmo de aprendizaje automático se basa en un corpus de ejemplos.

2.7.1. Tipos de aprendizaje automático

El aprendizaje automático es el subcampo de las ciencias de la computación que trata el estudio y construcción de sistemas de cómputo que pueden aprender de un conjunto de datos. En 1959 el investigador Arthur Lee Samuel definió aprendizaje automático como el campo de estudio que le da a las computadoras la habilidad de aprender sin ser programadas explícitamente [Simon, 2013].

Posteriormente el investigador Tom Michael Mitchell definió formalmente al aprendizaje automático como un problema bien planteado:

Definición 2.9. Aprendizaje automático: Un programa de computadora se dice que aprende de la experiencia E con respecto a una tarea T con una medida de rendimiento P, si su rendimiento en T, medido por P, mejora con la experiencia E [Anderson *et al.*, 1986].

Los algoritmos de aprendizaje automático pueden ser organizados de acuerdo a la siguiente lista:

Aprendizaje supervisado: El aprendizaje supervisado se fundamenta en suponer que existe un conjunto de entrenamiento que consiste en x puntos de datos (De algún conjunto X) y sus etiquetas y (De algún conjunto Y). El objetivo radica en aprender una función $f : X \rightarrow Y$ que prediga de forma precisa las etiquetas de puntos de datos que surjan en el futuro, por lo que el aprendizaje supervisado es sinónimo de clasificación [Sammut & Webb, 2011].

Aprendizaje no supervisado: El *aprendizaje no supervisado* es sinónimo de agrupamiento, está idea esta asociada a usar una colección de observaciones $X1,\ldots,Xn$ de una distribución $p(X)$ para describir las propiedades de $p(X)$. Sin embargo esta definición es muy genérica y podría describir cualquier proceso estadístico [Sammut & Webb, 2011]. Los métodos buscan aprender estructura en ausencia de una salida identificada o retroalimentación.

Aprendizaje semi-supervisado: El aprendizaje semi-supervisado es una técnica de aprendizaje automático que utiliza datos de entrenamiento tanto etiquetados como no etiquetados. Este tipo de aprendizaje se encuentra entre el aprendizaje no supervisado (i.e. sin datos de entrenamiento etiquetados) y el aprendizaje supervisado (con todos los datos de entrenamiento etiquetados) [Sammut & Webb, 2011].

Aprendizaje por refuerzo: Explora cómo los agentes de software deben actuar en un ambiente para maximizar alguna noción de recompensa. El agente ejecuta acciones que hacen que el estado observable del entorno cambie. A través de una secuencia de acciones, el agente intenta reunir conocimientos sobre cómo el medio ambiente responde a sus acciones, y sobre los intentos de sintetizar una secuencia de acciones que maximice una recompensa acumulada [Barto, 1998].

Aprendizaje activo: El termino aprendizaje activo generalmente es usado para referirse a un problema de aprendizaje o sistema donde el algoritmo de aprendizaje tiene algún tipo de rol en determinar sobre que conjuntos de datos será entrenado. El aprendizaje activo generalmente se usa en dominio donde obtener datos etiquetados resulta costoso o requiere una gran cantidad de tiempo [Sammut & Webb, 2011].

Aprendizaje profundo: Es un conjunto de algoritmos de aprendizaje que intentan modelar abstracciones de alto nivel en los datos mediante el uso de múltiples capas de procesamiento con estructuras complejas o compuestas por múltiples transformaciones no lineales [Bengio *et al.*, 2013].

Meta aprendizaje: En este tipo de aprendizaje se aplican algoritmos de aprendizaje automático en los meta-datos sobre experimentos de aprendizaje automático. Aunque diferentes investigadores mantienen opiniones diferentes en cuanto a lo que el término significa, el objetivo principal del uso de este tipo de meta-datos es entender cómo el aprendizaje automático puede llegar resolver diferentes tipos de problemas mejorando el rendimiento de algoritmos de aprendizaje existentes [Vilalta & Drissi, 2002].

2.8. Máquinas de soporte vectorial

Las máquinas de soporte vectorial o SVMs, por su siglas en inglés *Support Vector Machines* son una técnica para clasificar datos. Son clasificadores lineales que suponen la separabilidad lineal de los datos. Un clasificador lineal clasifica basándose en el valor de una combinación lineal de características. Es importante mencionar que las características de un objeto también se conocen como valores característicos, que generalmente son presentados al clasificador en forma de un vector de características. Por ejemplo, si consideramos el caso en $\mathbb{R}^2$, el algoritmo tratará de encontrar un vector óptimo que servirá para generar una línea que separe los puntos de datos. En otras palabras, el algoritmo

tratará de separar las distintas instancias con un margen máximo, que es un hiperplano que en $\mathbb{R}^2$ separaría los dos grupos de puntos y que esta a la misma distancia de ambos. De esta forma, el margen entre el hiperplano y los grupos de puntos es máximo.

La parte conceptual de este problema fue resuelta en 1965 por Vladimir Vapnik para el caso de hiperplanos óptimos para clases separables. Un hiperplano es definido por su función lineal de decisión con máximo margen entre los vectores de las dos clases, tal y como lo muestra la Figura 2.1. en donde podemos observar que para construir tales hiperplanos óptimos es necesario tomar en cuenta una pequeña cantidad de datos de entrenamiento, o los también llamados vectores de soporte, que son los que determinan ese margen.

Una tarea de clasificación generalmente involucra separar los datos en conjuntos de entrenamiento y prueba. A cada instancia del conjunto de entrenamiento le corresponde un valor objetivo, es decir la etiqueta de la clase, así como distintos atributos, es decir características o variables observadas. El objetivo de las máquinas de soporte vectorial es producir un modelo (basado en los datos de entrenamiento) que prediga los valores objetivo de los datos de prueba, dados solamente los atributos de los datos de prueba [Hsu *et al*., 2003].

2.8.1. Máquinas de soporte vectorial lineales

En [Wikipedia, 2016c] se menciona que dado un conjunto de entrenamiento de n puntos de la forma $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$, donde y_i es 1 o -1, donde cada uno indicará la clase a la cual el punto $\vec{x}_i$ pertenece. Cada $\vec{x}_i$ es un vector real de dimensión p. Queremos encontrar el hiperplano margen máximo, que divida el grupo de puntos $\vec{x}_i$ para $y_i = 1$ del grupo de los puntos para los cuales $y_i = -1$, que se define de modo que la distancia entre el hiperplano y el punto más cercano $\vec{x}_i$ de ninguno de los grupos se maximiza.

Cualquier hiperplano puede escribirse como el conjunto de puntos $\vec{x}$ que satisfacen $\vec{w} \cdot \vec{x} - b = 0,$ donde $\vec{w}$ es el vector normal al plano. El parámetro $\frac{b}{\|\vec{w}\|}$ determina el desplazamiento del hiperplano desde el origen a lo largo del vector normal $\vec{w}$.

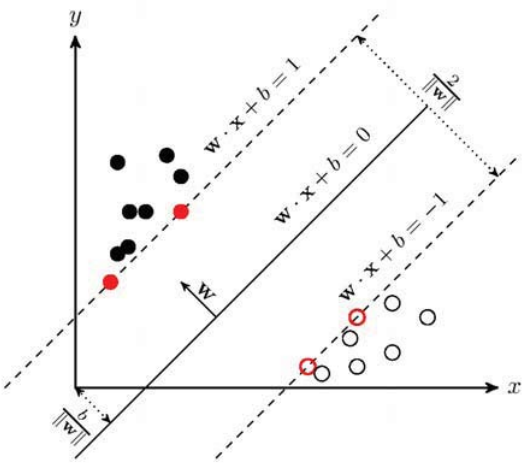


FIGURA 2.1: Representación en 2 dimensiones de una máquina de soporte vectorial

2.8.2. Margen duro

Si el conjunto de entrenamiento es linealmente separable, podemos seleccionar dos hiperplanos paralelos que separen las dos clases de datos, por lo tanto la distancia entre ellos siempre es lo más larga posible. A la región acotada por estos dos hiperplanos se le conoce como "margen", y el hiperplano máximo margen es el hiperplano que se encuentra en medio de ellos. Estos hiperplanos pueden ser descritos por las ecuaciones:

$$\vec{w} \cdot \vec{x} - b = 1$$

y

$$\vec{w} \cdot \vec{x} - b = -1$$

Geométricamente, la distancia entre estos dos hiperplanos es $\frac{2}{\|\vec{w}\|}$, entonces maximizar la distancia entre los planos que queremos minimizar es $\|\vec{w}\|$. Como también hay que evitar que los puntos de datos caigan dentro del margen, tenemos que considerar la siguiente restricción: para cada i ya sea:

$$\vec{w} \cdot \vec{x}_i - b \geq 1, \text{ si } y_i = 1$$

ó

$$\vec{w} \cdot \vec{x}_i - b \leq -1, \text{ si } y_i = -1$$

Estas restricciones indican que cada punto de datos estará situado en el lado correcto del margen, esto se puede reescribir como:

$$y_i(\vec{w}\cdot\vec{x}_i - b) \geq 1, \quad \text{para toda } 1 \leq i \leq n. \tag{2.1}$$

Podemos compaginar lo anterior para obtener el siguiente problema de optimización:

$$\text{Minimizar} \|\vec{w}\| \text{ restringido a } y_i(\vec{w}\cdot\vec{x}_i - b) \geq 1, \text{ con } i = 1, \ldots, n$$

El vector $\vec{x}$ y b que resuelve este problema determina nuestro clasificador: $\vec{x} \rightarrow \text{sign}\,(\vec{w}\cdot\vec{x} - b)$. Una consecuencia fácil de ver, pero importante de esta descripción geométrica es que hiperplano máximo margen se determina completamente por aquellos $\vec{x}_i$ que se encuentran cerca a el. Estos $\vec{x}_i$ son llamados vectores soporte.

2.8.3. Margen suave

Para extender el modelo de máquinas de soporte vectorial a los casos en los que los datos no son linealmente separables, a continuación se presenta la función "hinge loss":

$$\text{máx}\,(0, 1 - y_i(\vec{w}\cdot\vec{x}_i - b))$$

Esta función es igual a cero si la restricción en (2.1) se satisface, en otras palabras, si $\vec{x}_i$ esta del lado correcto del margen. Para los datos en el lado incorrecto del margen, el valor de la función es proporcional a la distancia del margen.

Cuando deseamos minimizar: $\left[\frac{1}{n}\sum_{i=1}^{n} \text{máx}\,(0, 1 - y_i(\vec{w}\cdot\vec{x}_i - b))\right] + \lambda\|\vec{w}\|^2$ donde el parámetro λ determina la compensación entre el aumento de tamaño del margen y asegurar que el vector $\vec{x}_i$ se encuentra del lado correcto del margen. Por lo tanto, para valores suficientemente pequeños de lambda, el margen suave de la maquina de soporte vectorial se comportará idénticamente al margen duro. Si los datos de entrada son linealmente clasificables pero aun podría aprender una regla clasificación valida si no.

2.8.4. Clasificación no lineal

En el aprendizaje automático, los métodos kernel son una clase de algoritmos para reconocer patrones. Este tipo de métodos trabaja con representaciones vectoriales como datos de entrada. Es decir, dado que los distintos tipos de algoritmos usados en aprendizaje automático no pueden operar directamente con texto, es necesario representar al texto en forma de vectores típicamente generados mediante atributos o propiedades medibles a partir de las distintas observaciones o instancias de ejemplos con las que se trabaja.

Por ejemplo, una propiedad medible en una opinión podría ser la frecuencia con la que se repiten las palabras. Consideremos los siguientes enunciados: *"A Juan le gusta ver películas. A Luis también le gusta ver películas"* y *"A Juan le gusta leer libros"*, luego, podemos extraer las palabras en una lista: `(1)[A, Juan, le, gusta, ver, leer, libros, películas, también]`, que nos servirá para armar los siguientes vectores de características: `(2)[2, 1, 2, 2, 2, 0, 0, 2, 1]` y `(3)[1, 1, 1, 1, 0, 1, 1, 0, 0]`. Donde la primer entrada del vector `(2)`, corresponde al número de veces que ocurre la primer palabra de la lista `(1)` en el primer enunciado, la segunda entrada del vector `(2)` corresponde al número de veces que ocurre la palabra *"Juan"* en el primer enunciado y así sucesivamente. Con respecto al segundo vector `(3)`, la primer entrada corresponde al número de veces que ocurre la palabra *"A"* en el segundo enunciado, la segunda entrada del vector `(3)`, corresponde al número de veces que la palabra *"Juan"* aparece en el segundo enunciado, de esta forma obtenemos los vectores de características: `[2, 1, 2, 2, 2, 0, 0, 2, 1]` y `[1, 1, 1, 1, 0, 1, 1, 0, 0]`.

Los métodos kernel hacen uso de funciones kernel, que les permiten operar en espacios de alta dimensión, donde se puede dar la separabilidad lineal de forma más sencilla, notemos que la separabilidad lineal es una propiedad es una propiedad que consiste en que dos conjuntos de datos pueden separarse si es posible generar un hiperplano que pueda separarlos. El modelo de las máquinas de soporte vectorial pertenece a esta familia de algoritmos, ya que en espacios de alta dimensión, el algoritmo puede calcular hiper-planos mediante el truco del kernel.

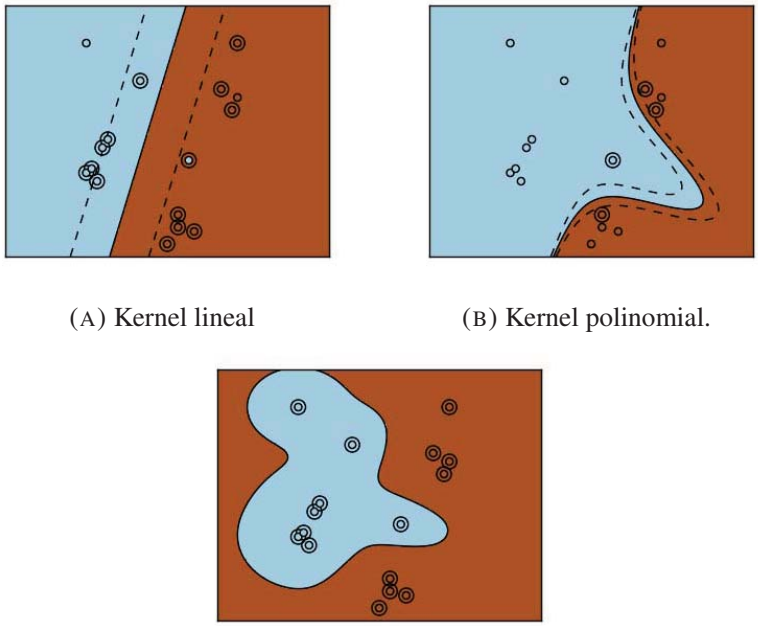
Funciones kernel

A continuación presentamos las funciones más usadas y sobre las cuales exploramos su funcionamiento en conjunto con las maquinas de soporte vectorial:

- Kernel lineal:
  Es la función kernel más simple. Es calculada por el producto interno $\langle x,y\rangle$ más una c constante:
  $$k(x,y) = x^{\intercal}Ty + c \tag{2.2}$$

- Kernel función de base radial:
  Este kernel mapea los datos a un espacio de dimensión infinita.
  $$k(x,y) = exp(\gamma||x-y||^2) \tag{2.3}$$

- Kernel polinomial:
  Esta función kernel es útil para problemas en los cuales los datos están normalizados:
  $$k(x,y) = (\alpha x^{\intercal}Ty + c)^d \tag{2.4}$$



(A) Kernel lineal (B) Kernel polinomial.

(C) Kernel RBF

FIGURA 2.2: Representación gráfica de el uso de funciones kernel con Máquinas de Soporte Vectorial, figura generada a partir de [Pedregosa *et al.*, 2011].

La figura 2.2 muestra cómo los distintos tipos de funciones kernel separan datos en el espacio. Una de las primeras cosas que claramente llama la atención, es como los vectores de soporte ayudan a crear los margenes que hacen posible la separación de

clases. De la misma forma podemos ver claramente como para el caso B y C, con ayuda de los métodos kernel podemos trabajar clasificación no lineal.

2.8.5. Formulación

Dado un conjunto de entrenamiento de parejas de instancias etiquetadas $(\mathbf{x}_i, y_i), i = 1, ..., l$ donde $\mathbf{x}_i \in \mathbb{R}^n$ y $y \in \{1, -1\}^l$, las máquinas de soporte vectorial necesitan resolver el siguiente problema de optimización:

$$min_{w,b,\xi} \quad \frac{1}{2} \quad w^T w + C \sum_{i=1}^{l} \xi_i \tag{2.5}$$

restringido a:

$$y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \quad \forall \xi_i \geq 0. \tag{2.6}$$

Donde los vectores de entrenamiento x_i pueden mapearse a espacios de mayor dimensión por la función ϕ. Las máquinas de soporte vectorial encuentran un hiper-plano separador lineal con el máximo margen en un espacio de mayor dimensión. $C > 0$ es, el parámetro de penalización del término de error.

Más información al respecto puede consultarse en [Hamel, 2011]. Debido a que el planteamiento teórico es ligeramente diferente a la implementación en [Chang & Lin, 2011a] y [Fan *et al.*, 2008] pueden encontrarse detalles técnicos que hacen posible el uso de este algoritmo.

2.8.6. Parámetros

Las máquinas de soporte vectorial cuentan con varios parámetros que deben ser cuidadosamente ajustados para tener buenos resultados. A continuación presentamos los parámetros más usados para realizar una clasificación multi-clase.

El parámetro C

El parámetro C se refiere a que tanto penaliza el modelo los vectores de datos dentro del margen [3]. Una C grande significa que habrá una gran penalización. Si la C es grande y los vectores son difíciles de separar, la máquina de soporte vectorial tratará de encontrar un hiper-plano y un margen complejo tal que pocos vectores de datos queden dentro del margen. Una C pequeña tiene como repercusión un gran error sobre el conjunto de entrenamiento, pero encuentra un margen más grande que podría llegar a ser más robusto.

En conclusión podemos decir que una C pequeña, produce un "margen suave", y permite más vectores soporte, dentro del margen. En cambio, una C más grande hace que el costo del error de la clasificación sea mayor, convirtiéndolo en un "margen duro". La Figura 2.3 generada por libsvm, muestra un ejemplo del efecto que tiene el parámetro C al clasificar datos. Los vectores de soporte son los puntos que se encuentran más cerca de la superficie de decisión o hiperplano. Son los puntos más difíciles de clasificar y estos tienen relación directa con la ubicación óptima de la superficie de la decisión. Un margen o hiperplano suave con $c = 1$ separa de forma deficiente a los puntos de la Figura 2.3., mientras que un margen duro con $c = 1000$ separa de forma casi perfecta a los puntos rojos de los azules, permitiendo una menor cantidad de vectores de soporte.

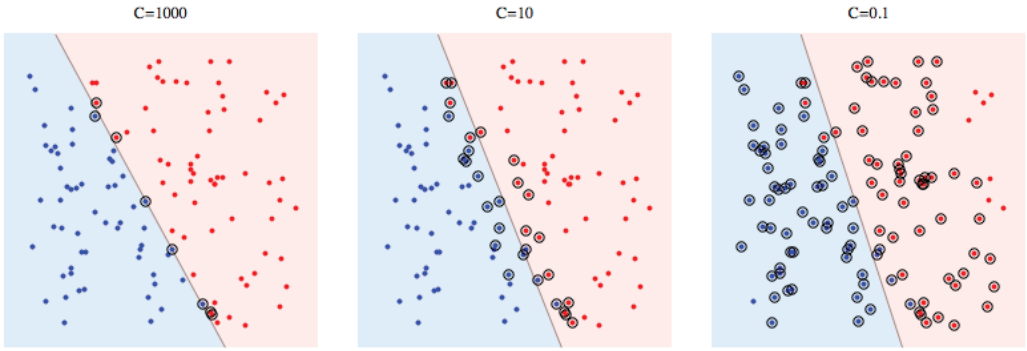


FIGURA 2.3: Representación en 2 dimensiones del efecto que tiene el parámetro C, figura generada a partir de [Chang & Lin, 2011b]

[3] "cuanto", se refiere a la tolerancia con la que separa el hiper-plano, los vectores de datos de color azul que están del lado equivocado

El parámetro gama

El parámetro gama generalmente se usa en conjunto con una función kernel RBF. Dicho parámetro establece que tan iguales deben ser dos vectores de datos para ser considerados "similares".

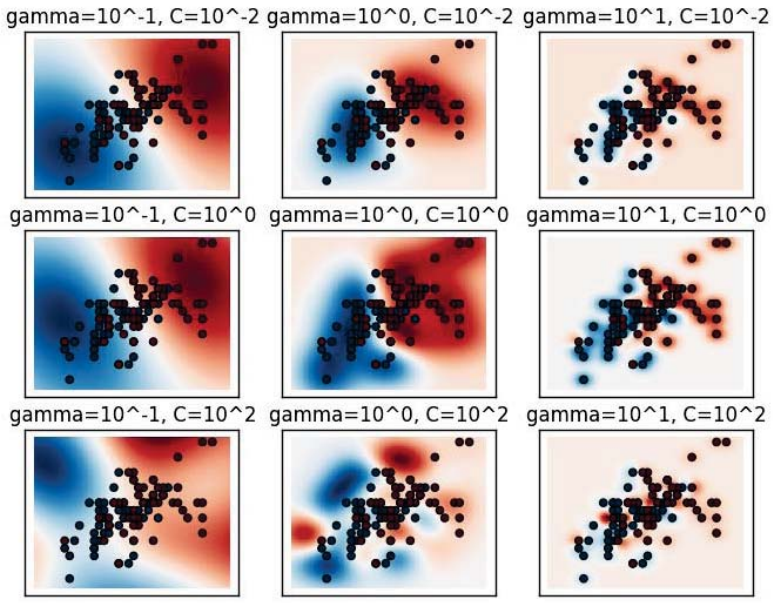


FIGURA 2.4: Representación en 2 dimensiones del efecto que tiene el parámetro gama figura generada a partir de [Pedregosa *et al.*, 2011].

2.8.7. El caso multi-clase

En [Hsu & Lin, 2002] se menciona que las máquinas de soporte vectorial fueron originalmente diseñadas para resolver problemas de clasificación binaria, sin embargo, su extensión al caso multi-clase continua siendo un problema abierto de investigación. Se han propuesto distintas metodologías donde generalmente se construye un clasificador multi-clase combinando distintos clasificadores binarios. Cabe destacar que los autores, mencionan que varios investigadores han propuesto métodos que consideran todas las clases al mismo tiempo, pero resolver este tipo de problemas de clasificación resulta ser computacionalmente muy costoso.

Principalmente, existen dos metodologías para resolver problemas de clasificación multi-clase que han demostrado ser practicas a la hora de resolver problemas que usan grandes

cantidades de datos *"Uno vs. Todas"* y *"Una vs. Una"*. A grandes rasgos la primer metodología, construye y combina varios clasificadores, mientras que la segunda considera todos los datos al formular un problema de optimización.

A continuación mencionamos de forma general dos de las tres metodologías de mayor aceptación, más información y detalles pueden consultarse en [Hsu & Lin, 2002].

Una vs. todas

Por [Hsu & Lin, 2002] sabemos que una de las primeras implementaciones para máquinas de soporte vectorial multi-clase es el método *"Una vs. Todas"*. Esta metodología construye k máquinas de soporte vectorial, donde k es el número de clases. La m-ésima máquina de soporte vectorial se entrena con todos los ejemplos en la m-ésima clase con etiquetas positivas y los demás ejemplos con etiquetas negativas. Por lo tanto, dados l datos de entrenamiento $(x_1, y_1), ..., (x_l, y_l)$, donde $x_i \in R^n$, $i = 1, ..., l$ y $y_i \in \{1, ..., k\}$ es de la clase de y_i, la m-ésima máquina de soporte vectorial resuelve el siguiente problema:

$$\begin{aligned} \min_{w^m, b^m, \xi^m} \quad & \frac{1}{2}(w^m)^T w^m + C\sum_{i=1}^{l} \xi_i^m \\ & (w^m)^T \phi(x_i) + b^m \geq 1 - \xi_i^m, \text{ if } y_i = m, \\ & (w^m)^T \phi(x_i) + b^m \leq -1 + \xi_i^m, \text{ if } y_i \neq m, \\ & \xi_i^m \geq 0, \ i = 1, \ldots, l, \end{aligned} \tag{2.7}$$

Donde los datos de entrenamiento x_i se mapean a un espacio de mayor dimensión por la función ϕ, y C el parámetro de penalización. Luego, el minimizar $\frac{1}{2}(w^m)^T\ w^m$ significa que debemos maximizar $\frac{2}{\|w^m\|}$, el margen entre los dos grupos de datos. Cuando los datos no son linealmente separables existe un termino de penalidad $C\sum_{i=1}^{l} \xi_i^m$, que puede reducir el número de errores de entrenamiento. El concepto básico detrás de las máquinas de soporte vectorial es buscar un balance entre la regularización del termino $\frac{1}{2}(w^m)^T w^m$ y los errores de entrenamiento. Después de resolver (2.7) tenemos que se generan k funciones de decisión:

$$\begin{aligned} &(w^1)^T\phi(x) + b^1, \\ &\vdots \\ &(w^k)^T\phi(x) + b^k. \end{aligned} \tag{2.8}$$

Entonces decimos que x pertenece a la clase que tenga el máximo valor generado a partir de la función de decisión:

$$x \equiv \mathbf{argmax}_{m=1,\ldots,k}((w^m)^T\phi(x) + b^m). \tag{2.9}$$

Una vs. una

Una vs. Una se refiere a otro método para resolver el problema de multi-clasificación con máquinas de soporte vectorial. Este método fue presentado originalmente en [Knerr *et al.*, 1990], pero se utilizó junto con máquinas de soporte vectorial por [Friedman, 1996] y [Kreßel, 1999]. A grandes rasgos este método construye $k(k-1)/2$ clasificadores donde cada uno se entrena en los datos de dos clases.

Para entrenar los datos de la i-ésima y la j-ésima clase, se resuelve el siguiente problema de clasificación binaria:

$$\begin{aligned} \min_{w^{ij},b^{ij},\xi^{ij}} \quad & \frac{1}{2}(w^{ij})^T w^{ij} + C\sum_t \xi_t^{ij} \\ & (w^{ij})^T\phi(x_t) + b^{ij} \geq 1 - \xi_t^{ij}, \text{ if } y_t = i, \\ & (w^{ij})^T\phi(x_t) + b^{ij} \leq -1 + \xi_t^{ij}, \text{ if } y_t = j, \\ & \xi_t^{ij} \geq 0. \end{aligned} \tag{2.10}$$

De acuerdo a [Hsu & Lin, 2002], existen diferentes métodos para hacer pruebas futuras después de que se construyeron los $k(k-1)/2$ clasificadores. Los autores decidieron usar los esquemas de votación planteados por [Friedman, 1996]: si la signatura $((w^{ij})\phi(x) + b^{ij})$ dice que x esta en la i-ésima clase, entonces se le añade un voto para la i-ésima clase. En otro caso el voto se le da a la j-ésima clase. Este esquema de votación también se llama estrategia "Max Wins". La predicción es que X esta en la clase

con mayor votación. En caso de que ambas clases tengan el mismo número de votos, se escoge la que tenga el número menor de índice.

De esta forma se resuelve el problema dual de las ecuaciones (2.10), cuyo número de variables son el mismo número de datos en las dos clases. Por lo tanto, si en promedio cada clase tiene l/k puntos de datos, se tienen que resolver $k(k-1)/2$ problemas de programación cuadrática donde cada uno de ellos tiene alrededor de $2l/k$ variables.

En las secciones anteriores se hace mención particular al modelo de las máquinas de soporte vectorial. Se presentó la formulación matemática así como los parámetros más importantes que repercuten en la obtención de buenos resultados. Se presenta el caso multi-clase y se presentan dos distintas metodologías para resolverlo (*Una Vs. todas* y *Una Vs. una*).

Para este trabajo después de probar con las distintas funciones kernel mencionadas en la sección 2.7.4., concluimos que los mejores resultados se lograban usando un kernel lineal.

3 Orientación semántica

En este capítulo presentamos los materiales utilizados para calcular la orientación semántica. De acuerdo al marco teórico presentado en el capítulo anterior describimos el trabajo realizado para obtener de esos materiales las expresiones que portan sentimiento. Seleccionamos frases siguiendo el modelo de lenguaje de n-gramas que se ha aplicado ampliamente en diversas tareas del procesamiento de lenguaje natural. Estas frases corresponden a las características que elegimos para el método de aprendizaje automático.

Definición 3.1. Orientación semántica: La orientación semántica o polaridad de una palabra se refiere a la dirección que la palabra deriva a partir de su norma hacia su grupo o campo semántico [Hatzivassiloglou & McKeown, 1997].

3.1. Orientación semántica mediante métodos no supervisados

Existen diferentes metodologías para derivar la orientación semántica en texto, desde un enfoque no supervisado el método de [Turney, 2002] es el más representativo, ya que se sustenta en el uso de las secuencias de categorías gramaticales presentadas en la Tabla 3.1.

	First Word	Second Word	Third Word (Not Extracted)
1.	JJ	NN or NNS	anything
2.	RB, RBR, or RBS	JJ	not NN nor NNS
3.	JJ	JJ	not NN nor NNS
4.	NN or NNS	JJ	not NN nor NNS
5.	RB, RBR, or RBS	VB, VBD, VBN, or VBG	anything

TABLA 3.1: Patrones de etiquetas extraídas en el trabajo de [Turney, 2002].

En este trabajo seguimos esta idea, por lo que a continuación describimos brevemente el método. En la Tabla 3.1, podemos observar los distintos patrones de marcas o etiquetas gramaticales que fueron usadas para el trabajo de [Turney, 2002]. El algoritmo propuesto por el Peter Turney realiza la clasificación no supervisada basándose en algunos patrones sintácticos conformados por etiquetas gramaticales y por siguientes tres pasos:

Paso 1: Se extraen dos palabras consecutivas, si sus categorías gramaticales conforman cualquier patrón de la Tabla 3.1.

Paso 2: Se estima la orientación semántica de la opinión calculando la perdida de información mutua mediante las palabras de las frases extraídas:

$$\text{SO (frase)} = log_2 \frac{\text{hits (frase CERCA excelente) hits(pobre)}}{\text{hits (frase CERCA pobre) hits(excelente)}} \tag{3.1}$$

La ecuación 3.1 se refiere al número de veces que las palabras de una frase aparecieron en más de una ocasión con “excelente”, entonces la orientación semántica es positiva. La orientación semántica es negativa cuando la frase aparecía mas de una vez con “pobre”. La orientación semántica de cada secuencia de dos palabras consecutivas se usó para determinar la orientación semántica de oraciones y opiniones completas.

3.2. Orientación semántica mediante métodos supervisados

En [Liu, 2010] se menciona que la clasificación de sentimiento puede formularse como un problema de aprendizaje supervisado con dos clases o etiquetas (positivo o negativo). Una de las ventajas de usar esta metodología es la gran cantidad de córpora disponible. Debido a que típicamente los sitios de opiniones, como Amazon, tienen un puntaje asignado por el usuario (por ejemplo, 1-5 estrellas), es natural entrenar y probar con distintos clasificadores supervisados.

Uno de los primeros trabajos en usar este enfoque fue [Pang *et al*., 2002], lo hicieron mediante palabras individuales como características. La clasificación tanto con máquinas de soporte vectorial como con Naive Bayes tuvo buen desempeño.

Un método ya clásico es el de [Hatzivassiloglou & McKeown, 1997]. Los autores presentaron un método completamente supervisado para determinar la polaridad de adjetivos en un corpus de tamaño significativo. Su enfoque se basa en crear un grafo, el cual los adjetivos son nodos y las aristas son ponderadas entre los adjetivos de acuerdo a una función de (dis)similiridad principalmente basada en que si los dos adjetivos ocurrieron en una conjunción o disyunción en el corpus. El grupo que contiene los adjetivos con las frecuencias promedio más altas es etiquetado como positivo, mientras que los otros se etiquetan como negativos.

Otro enfoque supervisado es el método basado en lexicones, el cual usa un diccionario de palabras de sentimiento con sus orientaciones asociadas y fuerza, también incorpora intensificación y negación para calcular el puntaje de sentimiento para cada documento [Taboada *et al*., 2011].

Como en la mayoría de las aplicaciones que hacen uso de algoritmos supervisados, la efectividad de la clasificación de sentimiento depende en gran medida del diseño de un conjunto de características efectivas. Algunos de estos aspectos son mencionadas por el trabajo de [Liu, 2012]:

- **Términos y su frecuencia:**
  Estas características son palabras individuales o secuencias de palabras con su frecuencia numérica. Son las características más usadas en la clasificación de

texto basada en temas o tópicos, pero han mostrado ser altamente efectivas para la clasificación de sentimiento.

- **Categorías Gramaticales:**
  Las categorías gramaticales agrupan palabras con propiedades gramaticales similares en términos de la sintaxis. Es decir, tienen funciones similares en la estructura gramatical de una opinión. En español son: sustantivos, verbos, adjetivos, pronombres, adverbios, preposiciones, conjunciones, determinantes. Las categorías que principalmente pueden portar sentimientos son los adjetivos y adverbios que funcionan en gran medida como atributos de sustantivos y verbos respectivamente.

- **Frases y palabras de sentimiento:**
  Las palabras de sentimiento son las palabras en un idioma que se usan para expresar sentimiento positivo o negativo (por ejemplo: *bueno, maravilloso y asombroso* se usan para expresar sentimiento positivo; *malo, pobre* y *terrible* son palabras usadas para expresar sentimiento negativo).

- **Reglas de opiniones:**
  Además de frases y palabras individuales, también hay frases de sentimiento y convenciones. Por ejemplo, *"La calidad de la voz es buena"*.

- **Conmutadores de sentimiento:**
  Son expresiones usadas para cambiar la orientación de sentimiento o polaridad en el texto, de positivo a negativo y viceversa. Los conmutadores de sentimiento se deben tratar con cuidado porque no todas las ocurrencias de este tipo de palabras implican el cambio en la polaridad de un texto. Las negaciones son la clase de palabras más importantes para cambiar la polaridad en un texto. Por ejemplo, *no* y *no solamente*.

- **Dependencia sintáctica:**
  Las características basadas en dependencia son generadas a partir del análisis sintáctico o de árboles de dependencias. Por ejemplo: `silencio NCMS000, y CC, facilidad NCFS000`. En donde `NCMS000`, `CC` y `NCFS000` son etiquetas o marcas gramaticales.

En estas secciones se planteó el problema de derivar la orientación semántica mediante métodos supervisados y métodos no supervisados, asimismo se mencionó trabajo relevante como es el caso de [Hatzivassiloglou & McKeown, 1997], [Turney, 2002] y [Liu, 2012] sobre los cuales nos basamos para resolver esta tarea.

Investigaciones más recientes usan muchos tipos de características o patrónes y técnicas para aprender. Como muchas aplicaciones de aprendizaje automático, la principal tarea de clasificar sentimiento es ingeniar un conjunto de características.

3.3. Recursos necesarios

Un corpus lingüístico es un conjunto amplio y estructurado de ejemplos reales del uso de la lengua. Estos ejemplos generalmente son textos o muestras orales transcritas. Otra característica importante es que un corpus lingüístico es un conjunto de textos relativamente grande creado de forma independiente a sus posibles aplicaciones. Es decir, en cuanto a su estructura, variedad y complejidad, un corpus debe reflejar una lengua o dominio de la forma más exacta posible. En cuanto a su uso, preocuparse de que su representación sea real. Los córpora tienen similitudes con los textos porque están compuestos por texto, por otro lado, no son textos en sí, porque a diferencia de los textos, un corpus no tiene sentido analizarlos en su totalidad. Un texto tiene un principio y un fin e incluso es en mayor o menor grado cohesivo y coherente. Por otro lado un corpus carece de tales características por no poseer una estructura, sino una composición [Wikipedia, 2015b].

La lingüística de corpus se refiere a la subdisciplina de la lingüística que estudia la lengua a través de estas muestras. Esta subdisciplina, dado el volumen de datos que maneja, suele asociarse con la lingüística computacional, según esta última se acerca a las aplicaciones del procesamiento de lenguaje natural [Wikipedia, 2015b].

3.3.1. Corpus de opiniones

Típicamente para derivar la orientación semántica a partir de opiniones es necesario un corpus. Para este trabajo contamos con la colección de opiniones de [Galicia-Haro & Gelbukh, 2014] sobre lavadoras automáticas. La colección fue compilada automáticamente del sitio *ciao.es* y consta de 2800 opiniones de lavadoras. El tamaño promedio

por opinión en lexemas es de 345. El número total de lexemas de la colección es de 854,280. La colección total fue anotada con información de lema[1] y etiquetas de categorías gramaticales (ó también llamadas POS-tags, que significan parts of speech en idioma inglés) utilizando FreeLing [Padró & Stanilovsky, 2012], una biblioteca de código abierto. En la Sección 4.2 especificamos detalles de esta anotación.

De la colección total de opiniones en español, recuperamos un subconjunto significativo de opiniones diferentes: 2598 opiniones. No eliminamos las opiniones que claramente son anuncios de empresas de mantenimiento (SPAM) ya que tanto este tipo de textos como las opiniones pagadas por fabricantes aparecen en cualquier colección de opiniones de productos.

Por lo tanto, con base a las características más importantes de las opiniones del producto analizado y con el puntaje de los usuarios que corresponde a: malo (una estrella), regular (dos estrellas), bueno (tres estrellas), muy bueno (cuatro estrellas) o excelente (5 estrellas) usamos esta colección para entrenar un clasificador supervisado, cuyo objetivo es determinar qué tan excelente o mala es la calidad de un producto.

Las opiniones tienen tanto errores gramaticales como ortográficos y de puntuación, pero debido a la diversidad de errores decidimos no aplicar métodos de corrección. *Freeling* es capaz de dar una categoría gramatical correcta aún con errores de ortografía.

Algunos de los datos más importantes de esta colección en cuanto a número de opiniones por puntaje se presentan en la Tabla 3.2. Debido al hecho de que estamos considerando opiniones de aparatos electrodomésticos y a la gran utilidad de estos ultimos en la vida cotidiana es un resultado natural que las opiniones favorables son mayores en una proporción de 6:1 con respecto a las negativas.

TABLA 3.2: Corpus de opiniones

Clase	Numero de instancias	Etiqueta (Número de estrellas)
Excelente	1190	5
Muy Bueno	838	4
Bueno	239	3
Regular	127	2
Malo	204	1

[1]En lingüística, un lema es una unidad autónoma constituyente del léxico de un idioma. Es una serie de caracteres que forman una unidad semántica.

Si la proporción de ejemplos de entrenamiento en una clase es mayor o menor, esto puede inducir un sesgo en la clasificación. Es por eso que en el área de aprendizaje automático se ha abordado el problema del desequilibrio de clases, en la cantidad de ejemplos de entrenamiento para cada una de ellas y las soluciones que se han dado fueron clasificadas por [López *et al*., 2013] de la siguiente manera:

- Modificación del algoritmo. Este enfoque tiene como objetivo adaptar los métodos de aprendizaje automático para que sean más sensibles a los problemas de desequilibrio de clases, por ejemplo [Sun *et al*., 2007].
- Asignación de pesos distintos a los ejemplos de entrenamiento, introduciendo diferentes costos a los ejemplos positivos y negativos. Este enfoque asigna costos más altos para la clasificación errónea de la clase mayoritaria, respecto a la clase minoritaria durante el entrenamiento de clasificadores, por ejemplo [Pazzani *et al*., 1994].
- Muestreo de datos, incluye bajo-muestreo, sobre-muestreo y métodos híbridos. El bajo-muestreo elimina instancias de clases mayoritarias mientras que el sobre-muestreo crea nuevas instancias de las clases menos frecuentes. Los métodos híbridos combinan los dos métodos anteriores, por ejemplo: [Tang *et al*., 2009]

La herramienta que empleamos en este trabajo permite hacer sobre-muestreo, sin embargo lo hemos dejado para un trabajo futuro, considerando que la proporción similar de desequilibrio de nuestra colección no es un caso que requiera métodos específicos de balanceo. Nos basamos en los resultados obtenidos por [Akbani *et al*., 2004], donde para una colección con similar proporción de desequilibrio no obtiene mejoras con diferentes métodos de balanceo.

3.4. El modelo del N-grama

En [Jurafsky, 2000] el autor describe la idea de predecir palabras mediante modelos probabilísticos llamados N-gramas, los cuales predicen la siguiente palabra a partir de las $N-1$ palabras anteriores. Estos modelos estadísticos de secuencias de palabras también se llaman modelos de lenguajes. Calcular la probabilidad de predecir la siguiente palabra está muy relacionado a calcular la probabilidad de una secuencia de palabras. El

autor menciona que los estimadores como los *n*-gramas, asignan una probabilidad condicional a las siguientes posibles palabras y pueden usarse para asignar una probabilidad conjunta a una oración completa.

Debido a que los *n*-gramas pueden predecir las siguientes instancias de palabras o secuencias de palabras completas, los *n*-gramas son una de las herramientas más importantes en procesamiento del lenguaje.

En otras palabras, un *n*-grama es una secuencia de lexemas. Estos lexemas generalmente son palabras, aunque también pueden ser caracteres o subconjuntos de caracteres. La *n* denota el número de lexemas o caracteres. Los *n*-gramas son sencillos y resultan ser de gran utilidad para capturar información estadística a partir de un conjunto de datos.

A continuación presentamos una oración contenida en el corpus de opiniones sobre lavadoras y los resultados al aplicar los modelos de 1-gramas, 2-gramas y 3-gramas.

Tiene diferentes revoluciones para el centrifugado, además de programas especiales para ropa sensible y planchado fácil.

- Para el modelo de 1-grama: *tiene, diferentes, revoluciones, para.*
- Para el modelo 2-gramas: *diferentes revoluciones, revoluciones para, el centrifugado*
- Para el modelo 3-gramas: *tiene diferentes revoluciones, diferentes revoluciones para.*

Si quisiéramos extraer las secuencias de dos palabras conformadas por las ocurrencias de sustantivos seguidos de adjetivos tendríamos el siguiente bigrama: *programa especial ropa sensible, planchado fácil, función especial, tiempo cercano*.

En el trabajo de [Wang & Manning, 2012] se afirma que la inclusión de bigramas logra mejoras significativas al momento de realizar tareas de minería de opiniones o análisis de sentimiento. Por esta razón, nosotros consideramos que los bigramas propuestos en las secciones 5.3 y 5.5 pueden aplicarse como características de entrenamiento para máquinas de soporte vectorial.

3.4.1. Bigramas positivos

Los bigramas se han utilizado como características en métodos no supervisados (i.e. métodos que no cuentan con ejemplos previamente anotados con la clasificación que se quiere aprender) basándose en el conteo de términos positivos y negativos, de esta forma es posible determinar automáticamente si el término es positivo o negativo.

En el trabajo de [Turney, 2002], se determinó la orientación semántica de una opinión mediante un algoritmo que primero extrae bigramas, es decir, secuencias de dos palabras donde una de ellas siempre es un atributo. Enseguida toma cada bigrama para realizar una búsqueda en la Web empleando el operador NEAR de AltaVista para encontrar cuántos documentos tienen ese bigrama cerca de un término positivo (*excellent*) y cerca de un término negativo (*poor*).

El puntaje para los dos conjuntos se realiza mediante la medida de información mutua puntual (Pointwise Mutual Information, PMI). La diferencia de puntuación para los dos conjuntos se utiliza para determinar el valor de la orientación semántica (SO-PMI), que da como resultado el grado en que cada (bigrama, término) es positivo o negativo. El puntaje PMI de dos palabras w_1 y w_2 se obtiene mediante la probabilidad de que las dos palabras aparezcan juntas dividida por las probabilidades de cada palabra se muestre en forma individual:

$$PMI(w_1, w_2) = \log_2[P(w_1, w_2)/[P(w_1)P(w_2)]] \tag{3.2}$$

Considerando el número de hits obtenidos de la Web, el cálculo de la orientación semántica fue realizado de la siguiente manera:

$$SO-PMI(frase) = \frac{log\ [hits(frase\ NEAR\ \text{“}excellent''})\ hits(\text{“}poor'')}{hits(frase\ NEAR\ \text{“}poor'')\ hits(\text{“}excellent'')]} \tag{3.3}$$

La orientación semántica de bigramas fue utilizada para determinar la orientación semántica de oraciones y opiniones completas. Turney tomó 410 comentarios de *epinions.com*. Los resultados oscilaron entre el 84 % para las revisiones de automóviles y el más bajo de 66 % para las críticas de películas. Siguiendo esta misma idea, en este trabajo, consideramos los siguientes bigramas sintácticos como características para el entrenamiento del método supervisado:

1. *sustantivo-adjetivo*

2. *verbo-adverbio*
3. *adverbio-adjetivo*
4. *adjetivo-adverbio*

Estos bigramas que llamamos sintácticos, no corresponden a compuestos obtenidos de un analizador sintáctico. En el caso sustantivo–adjetivo el programa que extrae estos bigramas comprueba la concordancia en género y número. Para todos los bigramas se extraen no las palabras sino los lemas, esto permite agrupar diversas formas en una sola característica. Por ejemplo: *prenda vaquera* y *prendas vaqueras*, *lavadora nueva* y *lavadoras nuevas*, se agrupan en un solo bigrama para cada par.

Consideramos los bigramas *adverbio-adjetivo* debido que los adjetivos y los adverbios modifican o describen otras palabras y su asociación sintáctica tiene significados especiales. Cuando un adverbio está junto a un adjetivo, su función semántica es cuantificar o cualificar [Spitzová *et al*., 1994], el caso más frecuente es cuando un adverbio tiene el rol de cuantificar. Incluimos también el bigrama *adjetivo–adverbio* porque aún cuando en español es más común la forma *adverbio-adjetivo*, encontramos que la forma inversa está presente en algunas opiniones de esta colección, por ejemplo, *super bien* y *barato siempre*. Mostramos la distribución de estos bigramas en la colección de opiniones en la Tabla 3.3:

	# Bigrams	**# Opinions**
Adjective-adverb	504	401
Adverb-adjective	7,484	2,024
Noun-adjective	21,144	2,598
Verb-adverb	27,900	2,006

TABLA 3.3: Distribución de estos bigramas en la colección de opiniones.

3.4.2. La negación

Una operación morfosintáctica es una relación dinámica y ordenada entre dos formas lingüísticas. Esta operación generalmente se manifiesta por operadores formales. Por ejemplo, los prefijos y los sufijos. De esta forma podemos decir que la negación es una operación morfosintáctica en la cual un componente léxico niega o invierte el significado de otro componente léxico o construcción lingüística. Por ejemplo, *“no”, ‘jámas”, “nunca”*.

La negación está presente en todos los lenguajes humanos y es usada para revertir la polaridad de enunciados que son afirmativos por defecto [Blanco & Moldovan, 2011]. Los autores subrayaron que un enunciado negado generalmente denota un significado positivo implícitamente, pero determinar la parte positiva de la negativa es difícil, por ejemplo, *Todos los vegetarianos no comen carne* significa que los vegetarianos no comen carne y el cuantificador universal todos tiene alcance sobre la negación. En [Wiegand *et al*., 2010] los autores describieron métodos para modelar la negación en una secuencia cronológicamente y los dividieron en representaciones que no contienen conocimiento explícito de expresiones polares y representaciones que contienen tal conocimiento. Ellos concluyeron que a pesar de la falta de plausibilidad lingüística, la clasificación supervisada usando *bag-of-words*, de la cual dimos una intuición en el ejemplo de la sección 2.8.4, ofrece un buen rendimiento en particular, si entrenamiento y prueba se hacen en el mismo dominio.

Un ejemplo de métodos que contienen conocimiento explícito de expresiones polares es el modelo implementado por [Zafra *et al*., 2015], donde la negación se codifica como características y se combina con aprendizaje automático supervisado. Ellos usaron un lexicón de cerca de 8,000 indicios subjetivos y un analizador sintáctico de dependencias para extraer caracterizas negativas, características que cambian el sentido positivo-negativo y características que modifican la polaridad de una opinión. También crearon un corpus para hacer experimentos añadiendo juicios de polaridad contextual para expresiones subjetivas.

3.4.3. Bigramas negativos

La negación en español fue dividida por [Sanz Alonso, 1996] en negación total y negación parcial. El autor analizó el efecto de la negación parcial considerando sintagmas [2] en palabras adyacentes a sintagmas y en palabras de negación.

Con respecto a estas últimas, el autor mencionó que los pronombres indefinidos *nadie, ninguno nada* y los adverbios *nunca, jamás, nada* resultan ser de gran importancia para identificar la negación en texto. En [Bergareche, 1992] el autor especificó que como adjetivo, la posición normal de *ninguno* es antes de un sustantivo; la alternancia entre antes y después de un sustantivo es solo posible cuando la secuencia se coloca después del verbo. El autor indica que *nunca* y *jamás* tienen una función análoga. También

[2] Un sintagma es un tipo de constituyente sintáctico formado por un grupo de palabras que forman otros subconstituyentes, en donde al menos uno de los cuales es un núcleo sintáctico.

analizó la negación de palabras de acuerdo a su posición antes y después del verbo; en una posición pre-verbal la negación de palabras en español nunca se acompaña por un adverbio de negación.

Seguimos el criterio anterior para este trabajo y manejamos la negación a un nivel superficial de secuencias morfosintácticas. Las formas negadas se obtuvieron buscando patrones específicos formados por secuencias de etiquetas de categorías gramaticales generados. Definimos los siguientes patrones:

1. $\text{No}_{\text{ADVERBIO}}$-$\text{verbo}_{\text{AUX\_PASADO PARTICIPIO}}$.
2. $\text{Ninguno}_{\text{LEMA\_DET}}$-sustantivo.
3. $\text{Jamás}_{\text{ADVERBIO}}$-verbo.
4. $\text{Nunca}_{\text{ADVERBIO}}$-verbo.
5. $\text{No}_{\text{ADVERBIO}}$-verbo.
6. $\text{Nada}_{\text{PRONOMBRE}}$-adjetivo.
7. $\text{No}_{\text{ADVERBIO}}$-pronombre-verbo.

La Tabla 3.4 muestra análisis estadístico de los indicios de negación existentes en el corpus. Notemos que el indicio $\text{no}_{\text{ADVERBIO}}$ constituye el 87 % de todos los indicios seleccionados en la colección.

TABLA 3.4: Frecuencia de indicios de negación en el corpus

Indicio	POS	Frecuencia	%	Indicio	POS	Frecuencia	%
no	Noun	52	0.4	ninguno	Pronombre	55	0.5
no	Adverbio	9,388	83.7	nada	Sustantivo	4	0.035
nunca	Sustantivo	5	0.04	nada	Pronombre	786	7.0
nunca	Adverbio	348	3.1	nada	Adverbio	125	1.1
ninguno	Determinante	427	3.8	jamás	Adverbio	20	0.17

Por ejemplo, consideremos el siguiente enunciado:

No nos cobro nada. Total que hemos cambiado de lavadora. Y que tenga un servicio técnico tan malo no la queremos.

Para el bigrama negativo 4, como anteriormente se menciono usamos la construcción sintáctica: $\text{no}_{\text{ADVERBIO}}$-pronombre-verbo

Similarmente si quisiéramos extraer las secuencias de tres palabras conformadas por una negación como lo es ”no” seguidas por un pronombre y un verbo tendríamos el siguiente N-grama con $n = 2$, o un bigrama: *no nos_cobrar, no lo_querer*

3.5. Consideraciones para este trabajo

Como anteriormente se dijo, para este trabajo se utilizó el corpus de [Galicia-Haro & Gelbukh, 2014], el cual consta de una colección de opiniones en formato *.txt* y una segunda colección marcada con categorías gramaticales mediante Freeling, que será mencionada a más detalle en el capítulo 4. Esta colección la transformamos a formato *.csv* considerando solamente cuatro características conformadas por: sustantivo-adjetivo (i.e. *programa especial*), verbo-adverbio (i.e. *tener además*), adverbio-adjetivo (i.e. *casi seco*) y adjetivo-adverbio (i.e. *tardó más*).

La Tabla 3.5 muestra el formato final del corpus usado en este trabajo, donde la columna “label” es la etiqueta asociada a la opinión (i.e. número de estrellas), la columna “content” es el contenido de cada opinión, mientras que el “id” es el identificador de cada opinión extraída. Para solamente considerar los bigramas [Pedregosa *et al.*, 2011] proporciona un método para especificar el rango del *n*-grama. Por lo que el rango del *n*-grama fue fijado a 2, de esta forma siempre se garantizó considerar bigramas.

De la misma forma procesamos la colección para obtener archivos que solo contienen las siguientes siete características negativas compuestas por: $\text{no}_{\text{ADVERBIO}}$-$\text{verbo}_{\text{AUX\_PASADO PART}}$ (i.e. *no haber_tener*), $\text{ninguno}_{\text{LEMA\_DET}}$-sustantivo (i.e. *ninguno sobresalto*), $\text{jamás}_{\text{ADVERBIO}}$-verbo (i.e. *jamás cambiar*), $\text{nunca}_{\text{ADVERBIO}}$-verbo (i.e. *nunca ver*), $\text{no}_{\text{ADVERBIO}}$-verbo (i.e. *no desagradar*), $\text{nada}_{\text{PRONOMBRE}}$-adjetivo (i.e. *nada blanco*), $\text{no}_{\text{ADVERBIO}}$-pronombre-verbo (i.e. *no me_enrollar*).

Para el método supervisado abordado en este trabajo cabe mencionar que nos basamos en las conclusiones del trabajo de [Wang & Manning, 2012], donde los autores hacen énfasis en que distintas variantes de métodos supervisados como Naive Bayes y máquinas de soporte vectoarial generalmente se usan como métodos para clasificar texto, sin embargo su rendimiento varía en gran medida en las particularidades inherentes de

cada modelo, características elegidas y hasta el mismo conjunto de datos usado para el entrenamiento.

En este trabajo, se destaca el potencial de la inclusión de bigramas para resolver tareas de análisis de sentimiento. En particular, los autores concluyen que el uso de máquinas de soporte vectorial para opiniones largas supera a varios métodos supervisados.

	id	content	label
0	AEG_Electrolux_60840_Lavamat__Opinion_1506705.txt	programa especial ropa sensible planchado fáci...	5
1	AEG_Electrolux_62610_Lavamat__Opinion_2000923.txt	carga superior punto medio color blanco carga ...	4
2	AEG_Electrolux_L14800Vl__Opinion_2005396.txt	carga superior forma frontal programa majo efi...	5
...	...	...	...
2595	Zanussi_ZWQ_598__Opinion_2004693.txt	mas grande display digital muy fácil	2
2596	Zanussi_ZWT_260__Opinion_1091527.txt	año aprox cosa raro utilizar mucho salir muy i...	2
2597	Zanussi_ZWT_260__Opinion_1157262.txt	lavadora normal lavadora pequeño pensar muy ce...	5

TABLA 3.5: Corpus de opiniones en formato *.csv*.

En este capítulo describimos los materiales que utilizamos para recuperar las características que permiten obtener las características de cada documento.

4 Materiales y Herramientas

A continuación presentamos las herramientas consideradas para realizar este trabajo. Así como algunas de las características más importantes de cada herramienta elegida para procesar el conjunto de las opiniones.

4.1. Python

Python es un lenguaje de programación multi-paradigma bastante conocido por su simplicidad y poder, generalmente se usa como "lenguaje de unión" o "lenguaje de pegado" para componentes escritos en otros lenguajes. La interfaz entre C/C++ y Python provee una sólida integración, mientras que los componentes definidos en Python pueden usarse por el núcleo de C/C++ [Demšar *et al*., 2004].[1]

[1]Python cuenta con un amplio soporte en el área de cómputo científico: visión por computadora, inteligencia artificial, matemáticas, astronomía, entre muchas otras. Como era de esperarse, esto también es válido para el área de aprendizaje automático.

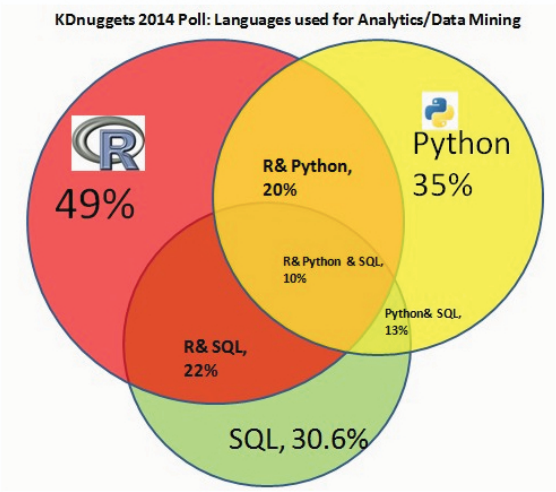


FIGURA 4.1: Los 4 lenguajes más usados para resolver tareas de aprendizaje automático

En la Figura 4.1 tenemos que según el sitio KDDNuggets[2]: R, SAS, Python y SQL son lenguajes que se han consolidado para la resolución de tareas que involucran técnicas de aprendizaje automático y minería de datos.

En [Pedregosa *et al.*, 2011], los autores resaltan que Python es un lenguaje de programación que se ha establecido como uno de los más utilizados para cómputo científico. Aunado a esto, Python cuenta con una gran cantidad de bibliotecas científicas y de análisis exploratorio de datos (e.g. Numpy, Scipy, pandas), además es un lenguaje de propósito general ampliamente usado en la industria y en la academia. En este contexto, usar un lenguaje de propósito general como Python ofrece una gran flexibilidad en comparación a otros lenguajes.

Muchas comunidades, consideran que la principal ventaja real de Python y de bibliotecas como scikit-learn sigue siendo su capacidad de aglutinar diferentes módulos, su facilidad de probar rápidamente en nuevos clasificadores y su flexibilidad a la hora de usar datos ya que no utiliza un formato de datos especifico.

[2]Four main languages for data analytics, data mining, and machine learning

4.2. Freeling

Freeling es una biblioteca de código abierto para procesamiento de lenguaje natural en varios idiomas. Provee una amplia gama de analizadores para varios idiomas, ofreciendo anotación o etiquetado de categorías gramaticales. Freeling es personalizable, extensible y en términos de robustez y velocidad tiene una fuerte orientación hacia aplicaciones para el mundo real. Los desarrolladores pueden usar los recursos lingüísticos que vienen configurados. Por ejemplo: diccionarios, lexicones y gramáticas para extenderlas y adaptarlas a dominios específicos o nuevos idiomas [Padró & Stanilovsky, 2012].

	as	ca	cy	en	es	gl	it	pt	ru
Tokenization	X	X	X	X	X	X	X	X	X
Sentence splitting	X	X	X	X	X	X	X	X	X
Number detection		X		X	X	X	X	X	X
Date detection		X		X	X	X		X	X
Morphological dictionary	X	X	X	X	X	X	X	X	X
Affix rules	X	X	X	X	X	X	X	X	
Multiword detection	X	X	X	X	X	X	X	X	
Basic named entity detection	X	X	X	X	X	X	X	X	X
B-I-O named entity detection				X	X	X		X	
Named Entity Classification				X	X	X		X	
Quantity detection		X		X	X	X		X	X
PoS tagging	X	X	X	X	X	X	X	X	X
Phonetic encoding				X	X				
WN sense annotation		X		X	X				
UKB sense disambiguation		X		X	X				
Shallow parsing	X	X		X	X	X		X	
Full/dependency parsing	X	X		X	X	X			
Co-reference resolution					X				

TABLA 4.1: Análisis de las distintas funcionalidades que ofrece Freeling para cada idioma.

La Tabla 4.1 muestra de lado izquierdo las funcionalidades soportadas por Freeling en distintos idiomas (as, ca, cy, en, es, gl, it, pt, ru). Curiosamente podemos observar que una ventaja de Freeling es que es la única herramienta que provee soporte para el idioma español, a diferencia de otras herramientas como Stanford Core NLP [Manning *et al.*, 2014] y NLTK [Bird *et al.*, 2009].

El analizador morfológico para el idioma español del que hace uso Freeling utiliza un conjunto de etiquetas o marcas para representar la información morfológica de las palabras, estas etiquetas o marcas corresponden a categorías gramaticales, a continuación presentamos las consideradas para este trabajo.

En las Tablas 4.2, 4.4, 4.6 y 4.8 encontramos un número que hace referencia al orden y posición en que aparecen los atributos. La columna 2 de las tablas antes mencionadas, hace referencia a los atributos, el número de los cuales varía dependiendo de la categoría gramatical. En la columna 3, podemos observar los valores que puede tomar cada atributo. Por otro lado, la columna 4 representa la convención de codificación que se estableció para la representación de categorías gramaticales.

Los adjetivos son representados de la siguiente forma: Adjetivo AQ- - - -, donde "-" corresponde a una codificación de la Tabla 4.2:

ADJETIVOS			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Adjetivo	A
2	Tipo	Calificativo	Q
		Ordinal	O
3	Grado	Aumentativo	A
		Diminutivo	D
		Comparativo	C
		Superlativo	S
4	Género	Masculino	M
		Femenino	F
		Común	C
5	Número	Singular	S
		Plural	P
		Invariable	N
6	Función	-	0
		Participi	P

TABLA 4.2: Especificación de las etiquetas para los adjetivos, tomado de [TALP-UPC, 2012].

De la Tabla 4.2 tendríamos los siguientes ejemplos que se presentan en la Tabla 4.3. Una de las primeras cosas que podemos notar, es que para la palabra "alegres", tenemos que su lema es "alegre" y su etiqueta esta formada por AQ0CP0, que corresponde a la codificación conformada por adjetivo, calificativo, sin una función, comparativo, plural, sin ninguna función. :

Forma	Lema	Etiqueta
alegres	alegre	AQ0CP0
alegre	alegre	AQ0CS0
bonita	bonito	AQ0FS0
grandazo	grande	AQAMS0
pésimo	malo	AQSMP0
pequeñitas	pequeño	AQDFP0
antiarrugas	antiarrugas	AQ0CN0
desnuda	desnudo	AQ0FSP

TABLA 4.3: Ejemplificación de las codificaciones posibles para adjetivos, tomado de [TALP-UPC, 2012].

Los adverbios son representados por R [G | N]. La etiqueta de adverbio negativo (RN) está reservada exclusivamente para el adverbio "no".

ADVERBIOS			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Adverbio	R
2	Tipo	General	G
		Negativo	N

TABLA 4.4: Especificación de las etiquetas para los adverbios, tomado de [TALP-UPC, 2012].

Por ejemplo, en la Tabla 4.5, para el adverbio "despacio", podemos ver que se encuentra representado por RG, es decir: adverbio, general.

Forma	Lema	Etiqueta
despacio	despacio	RG
ahora	ahora	RG
siempre	siempre	RG
hábilmente	hábilmente	RG
posteriormente	posteriormente	RG
a_cuatro_patas	a_cuatro_patas	RG
a_granel	a_granel	RG
no	no	RN

TABLA 4.5: Ejemplificación de las etiquetas para los adverbios, tomado de [TALP-UPC, 2012].

Los sustantivos están marcados como NC, definidos por la Tabla 4.6, por ejemplo, :

NOMBRES			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Nombre	N
2	Tipo	Común	C
		Propio	P
3	Género	Masculino	M
		Femenino	F
		Común	C
4	Número	Singular	S
		Plural	P
		Invariable	N
5-6	Clasificación semántica	Persona	SP
		Lugar	G0
		Organización	O0
		Otros	V0
7	Grado	Aumentativo	A
		Diminutivo	D

TABLA 4.6: Especificación de las etiquetas para los sustantivos, tomado de [TALP-UPC, 2012].

Es decir, por ejemplo: “chico” esta codificado por NCMS000, donde N significa nombre, C y P significa que corresponde al tipo común y al tipo propio.

Forma	Lema	Etiqueta
chico	chico	NCMS000
chicas	chico	NCFP000
gatito	gato	NCMS00D
oyente	oyente	NCCS000
oyentes	oyente	NCCP000
cortapapeles	cortapapeles	NCMN000
tesis	tesis	NCFN000
Barcelona	barcelona	NP000G0
COI	coi	NP000O0
Pedro	pedro	NP000P0

TABLA 4.7: Ejemplificación de las etiquetas para los sustantivos, tomado de [TALP-UPC, 2012].

Los verbos están marcados por la siguiente codificación, descrita por la Tabla 4.7:

VERBOS			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Verbo	V
2	Tipo	Principal	M
		Auxiliar	A
		Semiauxiliar	S
3	Modo	Indicativo	I
		Subjuntivo	S
		Imperativo	M
		Infinitivo	N
		Gerundio	G
		Participio	P
4	Tiempo	Presente	P
		Imperfecto	I
		Futuro	F
		Pasado	S
		Condicional	C
		-	0
5	Persona	Primera	1
		Segunda	2
		Tercera	3
6	Número	Singular	S
		Plural	P
7	Género	Masculino	M
		Femenino	F

TABLA 4.8: Especificación de las etiquetas para los verbos, tomado de [TALP-UPC, 2012].

Por ejemplo: tenemos que la palabra “cantada” tiene como lema “cantar” y le corresponde la etiqueta VMP00SF, que significa verbo del tipo principal donde carece de algún tiempo gramatical especifico, del número singular, del genero femenino.

Forma	Lema	Etiqueta
cantada	cantar	VMP00S***F***
cantadas	cantar	VMP00P***F***
cantado	cantar	VMP00S***M***
cantados	cantar	VMP00P***M***

TABLA 4.9: Ejemplificación de las etiquetas para los verbos, tomado de [TALP-UPC, 2012].

4.3. pandas

El proyecto pandas es una biblioteca de Python con una gran cantidad de estructuras de datos y herramientas para trabajar con datos estructurados comunes en estadística, finanzas, ciencias sociales y muchos otros campos. Esta biblioteca provee rutinas para hacer operaciones de manipulación y análisis de datos. El objetivo de esta biblioteca es el de convertirse en la herramienta fundamental de cómputo estadístico en Python. Siendo una gran alternativa para bibliotecas estadísticas actuales o lenguajes de programación como R [McKinney, 2010].

McKinney menciona que esta biblioteca desarrollada desde el 2008, esta enfocada en cerrar la brecha entre la riqueza de herramientas disponibles para el análisis de datos y el lenguaje de programación Python. Finalmente esta biblioteca no solo provee la misma funcionalidad que otros proyectos, sino que implementa nuevas características como el alineamiento automático de datos. Una característica importante de pandas son los *DataFrames*, una estructura de datos heterogénea tabular de tamaño mutable con ejes etiquetados (filas y columnas). Esta estructura de datos puede ser pensada como un diccionario para series de objetos, es la principal estructura de datos en pandas, por ejemplo, la Tabla 3.5 .

En el trabajo de [McKinney, 2010] destaca que el nombre de la estructura “Dataframe” es el mismo que en el proyecto R y que en pandas se complementa mucho de la funcionalidad de su contraparte en R pero con importantes mejoras. Pero pandas implementa una robusta y completa cantidad de herramientas para Python que superan a las de R. Por ejemplo, R no implementa funcionalidades de índices ni esta cercanamente integrado como es el caso de pandas. Otra ventaja que cabe destacar es que pandas nos permite combinar, unir o fusionar datos que estén relacionados, debido a que los desarrolladores suelen estar interesados en asociar observaciones de un conjunto de datos con otros, mediante una llave.

Para este trabajo usamos pandas para darle formato al corpus, alinear los datos, leer y vectorizar el corpus de opiniones en arreglos Numpy, mediante un dataframe.

4.4. NumPy

Numpy es la biblioteca fundamental para realizar cómputo científico en Python. Es una biblioteca que provee objetos en forma de arreglos multidimensionales, así como varios objetos que se derivan a partir de arreglos multidimensionales (tales como matrices y matrices enmascaradas), y una variedad de rutinas para operaciones rápidas y eficientes sobre matrices, incluyendo operaciones matemáticas, lógicas, manipulación, clasificación, selección, entrada y salida de datos, transformadas de Fourier discretas, álgebra lineal, operaciones estadísticas, simulación aleatoria.

En [Van Der Walt *et al*., 2011] se menciona que que a diferencia de los arreglos convencionales, los arreglos de Numpy pueden tener cualquier dimensión. Además, pueden contener otros tipos de elementos (o incluso combinaciones de elementos), tales como booleanos o fechas. Internamente, un arreglo NumPy en realidad es una forma conveniente de describir uno o más bloques de memoria en disco. Por lo tanto, los números representados en estos arreglos pueden manipularse fácilmente.

En este sentido el autor hace énfasis que los arreglos NumPy son la representación estándar de datos numéricos en el lenguaje de programación Python. Estos arreglos permiten operaciones de cómputo numérico eficientes en un lenguaje de alto nivel. De la misma forma ofrece operaciones para el cálculo de vectorización sin tener que copiar los datos en memoria, minimizando por ende el cómputo de frecuencias.

En [Van Der Walt *et al*., 2011] se define el arreglo de NumPy como una colección uniforme multidimensional de elementos. Un arreglo se caracteriza por el tipo de elementos que contiene y por su forma. Por ejemplo, una matriz puede representarse como un arreglo multidimencional que contenga números, por ejemplo, de punto flotante o números complejos.

El autor, hace énfasis en que los arreglos n-dimensionales de NumPy son una estructura de datos de alto nivel que facilitan operaciones de vectorización. Su sofisticada descripción de memoria permite una amplia variedad de las operaciones para realizarse sin copiar datos a la memoria, con lo cual las ganancias de rendimiento son significativas,cuando se trata de grandes cantidades de datos.

Numpy package se conforma de arreglos NumPy y de un conjunto de funciones matemáticas de alto nivel. Esta biblioteca ha encontrado una muy extendida adopción en

la industria, el mundo académico y laboratorios nacionales, con aplicaciones que van desde los juegos hasta la exploración espacial.

Finalmente, para este trabajo los arreglos Numpy fueron útiles para generar una representación matricial que acepta la implementación de máquinas de soporte vectorial.

4.5. Scikit-learn aprendizaje automático en Python

El proyecto scikit-learn es una biblioteca para Python que integra una gran cantidad de algoritmos de aprendizaje automático para problemas de mediana escala. La estructura básica de datos en scikit-learn es NumPy, los datos de entrada se presentan como arreglos de numpy, por lo tanto se integra fácilmente con otras bibliotecas científicas como SciPy, y Cython [Pedregosa *et al*., 2011]. Entre sus ventajas resaltan las siguientes:

- Herramientas simples y eficientes para minar y analizar datos.
- Código reutilizable en varios contextos.
- Construido sobre NumPy, SciPy y matplotlib.
- Es una biblioteca de código abierto, comercialmente útil.
- Licencia Berkeley Software Distribution.

En particular, scikit-learn implementa 6 distintos tipos de algoritmos de aprendizaje automático:

- **Clasificación:**
  Útiles para identificar a qué categoría pertenece un objeto (e.g. máquinas de soporte vectorial, vecinos cercanos, bosques aleatorios).
- **Regresión:**
  Útil para predecir un atributo de valor contínuo asociado a un objeto (e.g. Support Vector Regression, regresión de arista, Modelo lineal Lasso)
- **Clustering:**
  Útil para agrupar automáticamente objetos similares en conjuntos (e.g. K-means, agrupamiento espectral)

- **Reducción de dimensionalidad:**
  Reducen el número de variables aleatorias a considerar (e.g. Análisis de Componentes Principales, selección de características, factorización de matrices no negativas).

- **Selección de Modelos:**
  Sirven para comparar, validar y escoger tanto parámetros como modelos (e.g. grid-search, validación cruzada, distintas métricas).

- **Preprocesamiento:**
  Proporciona varias funciones y clases que transforman datos a vectores de características en una representación que puede ser entendible por los distintos algoritmos implementados en scikit-learn (e.g. extracción de características y normalización).

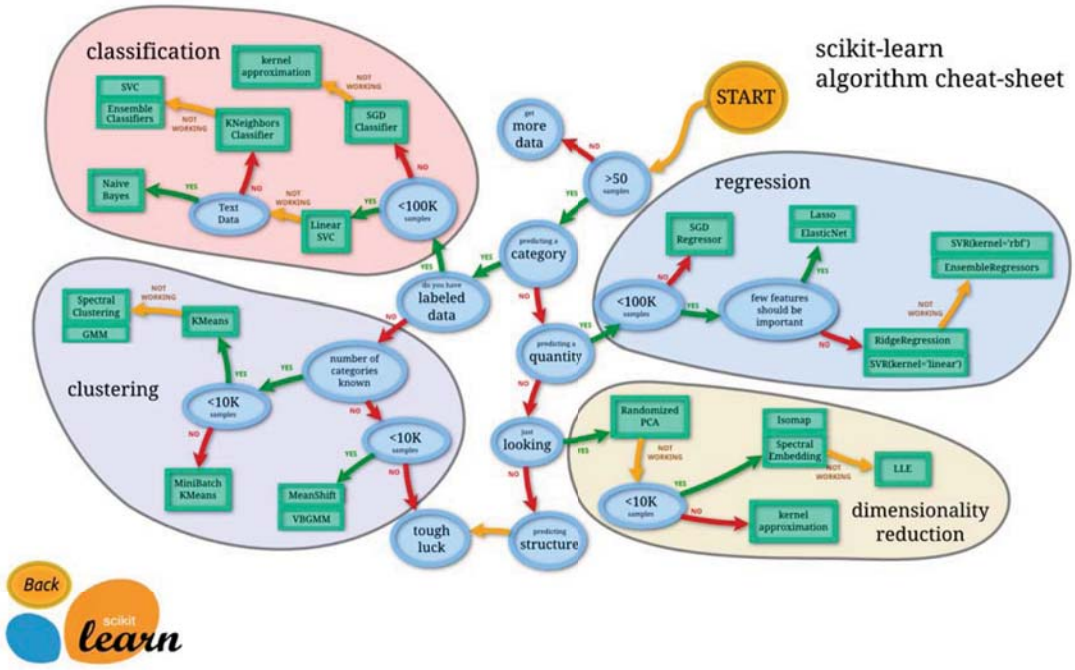


FIGURA 4.2: Algoritmos disponibles en scikit-learn, figura tomada de [Pedregosa *et al.*, 2011].

La Figura 4.2 muestra las distintas implementaciones de algoritmos disponibles en el proyecto *scikit-learn*. Esta figura nos da una idea de cómo proceder a la hora de realizar tareas de aprendizaje automático. Si nos posicionamos en el comienzo y vamos contestando las preguntas que vienen en cada cada arista tendremos una idea de como resolver una tarea de aprendizaje automático.

4.6. Aprendizaje supervisado con scikit-learn

El **aprendizaje supervisado** consiste en el aprendizaje de la relación entre dos conjuntos de datos: los datos observados X y una variable y externa que estamos tratando de predecir, por lo general llamado "objetivo" o "etiqueta". Típicamente, y es una matriz de dimensión n-muestras que denota las etiquetas. Todos los estimadores supervisados en scikit-learn implementan y hacen uso un método de ajuste: `fit()`, que ajusta la matriz X a las etiquetas y, para emparejar el modelo. Además cuenta con un método `predict(X_new)` para predecir (X_{nuevas}) que, dadas las observaciones no etiquetadas X, devuelve las etiquetas predichas de y [Pedregosa *et al*., 2011].

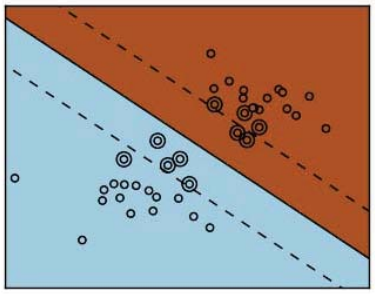


FIGURA 4.3: Máquina de soporte vectorial regularizada [Pedregosa *et al*., 2011].

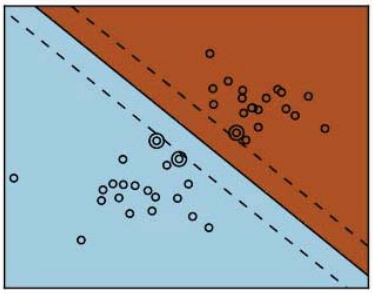


FIGURA 4.4: Máquina de soporte vectorial no regularizada [Pedregosa *et al*., 2011].

En este trabajo hicimos uso de máquinas de soporte vectorial, que pertenecen a la familia de modelos discriminantes. Este tipo de modelos trata de encontrar una combinación de muestras para construir un hiper-plano que separé los datos en distintas clases.

La regularización, como se mencionó anteriormente en la Sección 2.8.2, del Capítulo 2, es determinada por el parámetro C: un valor pequeño para C significa que el margen se calcula utilizando muchas o todas las observaciones alrededor de la línea de separación; un valor grande para C, significa que el margen se calcula en observaciones cercanas a la línea de separación. Las Figuras 4.3 y 4.4 ilustran el efecto de la regularización en máquinas de soporte vectorial.

4.7. Optimización de hiper-parámetros

Muchos de los algoritmos de aprendizaje automático supervisado pueden ser vistos de la siguiente forma:

1. Sean (x_i, y_i) un conjunto de puntos de entrenamiento donde queremos encontrar una función f que "ajuste a los datos correctamente".

$$y_i \approx f(x_i) \tag{4.1}$$

2. Por otro lado si (4.1) es muy simple, posiblemente no se cumplirá para todos los valores de i; si es muy compleja, ajustará los datos muy bien, pero no lo podrá hacer con nuevas instancias de datos (i.e. **sobre-ajuste**).

3. Para determinar qué tan complejo es seleccionar (4.3) se toma una clase de funciones $\boldsymbol{F}$ donde la complejidad sea más sencilla de controlar.

4. Encontrar una función $f \in \boldsymbol{F}$ que ajuste correctamente en los datos de entrenamiento. Luego, el problema de controlar la complejidad de f se ha reducido al problema de controlar la complejidad de $\boldsymbol{F}$. Debido a que no hay forma de encontrar una $\boldsymbol{F}$ óptima, se deben intentar un conjunto de clases de funciones $\boldsymbol{F_1,...,F_k}$ y tomar la que mejor lo haga en instancias de datos no vistos.

5. Por lo tanto, al proceso de escoger la mejor clase se le conoce como optimización de hiper parámetros.

En [Davidson-Pilon, 2014] se menciona que los hiper-parametros son los parámetros de los parámetros de un modelo. Por ejemplo, si C tiene una distribución de Poisson: $C \sim Poisson(\lambda)$ y λ tiene una distribución Exponencial: $\lambda \sim Exp(\alpha)$ Entonces α es el hiper-parametro.

La tarea de optimizar hiper-parámetros o encontrar un conjunto cercano óptimo de parámetros libres que pueden asignarse manualmente fuera del espacio un algoritmo de aprendizaje, se refiere a tratar de encontrar la configuración apropiada para los distintos "botones" a configurables, para poder obtener una generalización correcta entre el rendimiento de la muestra de datos. Podemos pensar lo anterior como otro espacio de características que estamos explorando a un nivel abstracto o meta-nivel de un modelo que potencialmente puede ser altamente dimensional.

4.7.1. Grid Search

Para este trabajo decidimos usar Grid Search debido a que tradicionalmente es la más usada para encontrar la mejor configuración, es decir la que aporte la mejor exactitud, F1-score, recall, precisión. Una Grid Search consiste en:

1. Un estimador.
2. Un espacio de parámetros.
3. Un método para buscar o muestrear candidatos.
4. Un esquema de validación cruzada.
5. Una función de score.

Una Grid search es una búsqueda exhaustiva a través de un subconjunto del espacio de hiperparametros de un algoritmo de aprendizaje. Un algoritmo de grid search debe ser guiado por una métrica de rendimiento, típicamente medida por la validación cruzada en el subconjunto de entrenamiento [Hsu *et al.*, 2003].

4.8. Matrices de confusión

Respecto al problema de estimar k clases para un conjunto de prueba con n instancias, las clases correctas se denotan por C_i, mientras que las clases estimadas, definidas por el clasificador considerado, se denotado por $\hat{C}(1 \leq i \leq k)$ [Labatut & Cherifi, 2012].

En [Wikipedia, 2015a] se menciona que una matriz de confusión es una herramienta que permite la visualización del desempeño de un algoritmo que se emplea en aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. Uno de los beneficios de las matrices de confusión es que permiten, detectar con facilidad, si el sistema está confundiendo las clases.

Muchas métricas no se procesan directamente a partir de las predicciones de un clasificador; no obstante, una matriz de confusión se construye a partir de los resultados de un clasificador.

Esta matriz representa la distribución sobre instancias estimadas (columnas) y clases acertadas (filas):

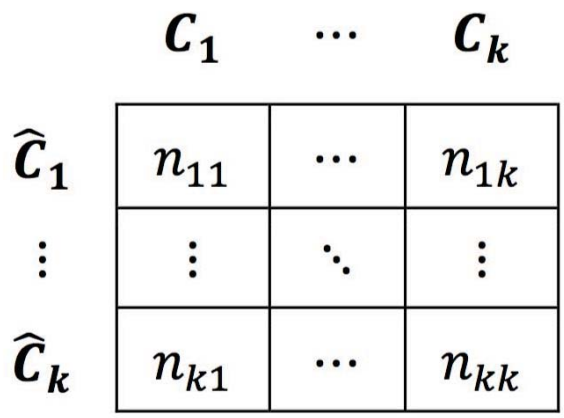


FIGURA 4.5: Matriz de Confusión [Labatut & Cherifi, 2012].

Los términos n_{ij} con $(1 \leq i, j \leq k)$ corresponden al número de instancias en la clase número i colocadas por el clasificador (i.e. $\widehat{C}_j$) cuando pertenecen a la clase número j (i.e. C_j), por lo que los términos que se encuentran en la diagonal corresponden a las instancias clasificadas correctamente, $(i = j)$, mientras que los términos fuera de la diagonal $(i \neq j)$ representan las instancias clasificadas incorrectamente. Algunas métricas se definen en términos de $p_{ij} = n_{ij}/n$.

La suma de los elementos de la matriz de confusión sobre la fila i y la columna j son denotadas respectivamente por n_{i+} y n_{+j}. Las sumas correspondientes a las proporciones, p_{i+} y p_{j+}, se definen de forma análoga a las anteriores.

Cuando se considera una clase i, una de las formas de distinguir cuatro tipos de instancias es: verdaderos positivos y falsos positivos son instancias correcta e incorrectamente clasificadas como $\widehat{C}_i$, mientras que los verdaderos negativos y falsos negativos son instancias correcta e incorrectamente no clasificadas por el estimador $\widehat{C}_i$. Cada una de las frecuencias correspondientes son definidas por $n_{TP} = n_{ii}, n_{FP} = n_{i,+} - n_{i,i}$ y $n_{TN} = n - n_{TP}$ y $n_{TN} = n - n_{TP} - n_{FP} - n_{FN}$. Mientras que las proporciones correspondientes son denotadas respectivamente por $P_{TP}, P_{FP} y P_{TN}$

4.9. Validación cruzada

La validación cruzada es un modelo para evaluar la manera en que resultados de análisis estadísticos generalizarán en un futuro y en un conjunto de datos completamente nuevo. Principalmente, se usa en tareas donde el objetivo principal es predecir y uno desea estimar qué tan exacto es un modelo predictivo y el rendimiento que tendrá en la práctica.

La Figura 4.6 muestra de forma gráfica cómo se evalúa con validación cruzada. El cuadrado rojo denota la parte de los datos que se usa para probar, mientras que el azul denota la parte de los datos que se usa para entrenar.

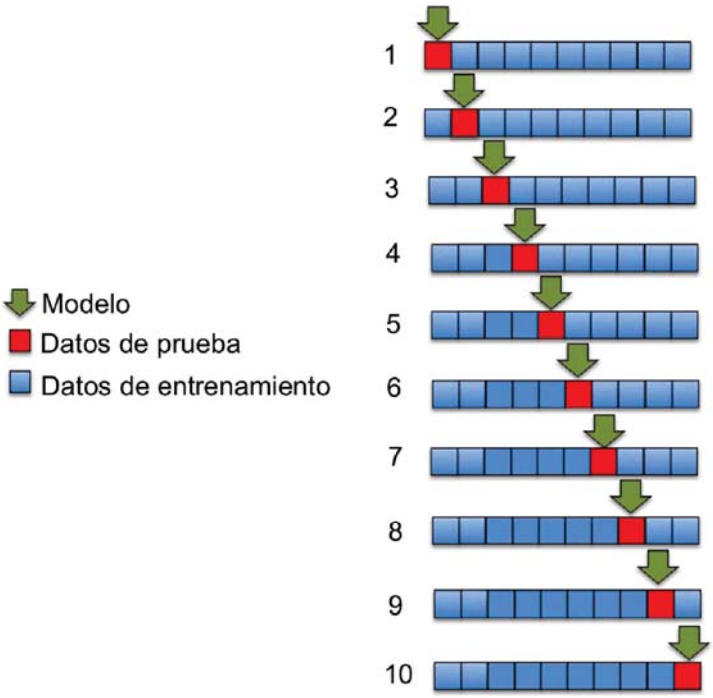


FIGURA 4.6: Validación cruzada con 10 pliegues o folds.

Aprender los parámetros de una función de predicción y probarlos en los mismos datos es un error metodológico: un modelo que sólo repita las etiquetas de las mismas muestras que se acaba de analizar tendría una puntuación perfecta, pero fallaría en predecir algo útil sobre muestras de datos completamente nuevas. A esta hecho se le conoce como *sobre-ajuste*. Para contrarrestar este hecho se realiza una validación cruzada la cual reduce la posibilidad de sesgo en el modelo.

Durante un experimento [3] de aprendizaje automático supervisado una práctica común es separar una parte de los datos como conjunto de prueba y conjunto de entrenamiento.

En [Pedregosa *et al.*, 2011] los autores mencionan que scikit-learn implementa distintas estrategias para realizar validación cruzada y separar en un conjunto de prueba y entrenamiento el total de los datos:

- **K-fold:**
  Esta estrategia divide todas las muestras en k grupos de muestras, llamadas pliegues si $k = n$, esto es equivalente a la estrategia *Leave One Out strategy*, de tamaños iguales (si es posible). La función de predicción se aprende utilizando $k - 1$ pliegues, el pliegue excluido se utiliza para la prueba.

- **Stratified k-fold:**
  Es una variación de una estrategia *k-fold* que devuelve pliegues estratificados. Es decir, cada conjunto contiene aproximadamente el mismo porcentaje de muestras de cada clase objetivo.

- **Label k-fold:**
  LabelKFold es una variación de k-fold que asegura que la misma etiqueta no está en ambos conjuntos de prueba y de entrenamiento. Esto es necesario, por ejemplo, si se han obtenido los datos de diferentes temas y se quiere evitar el sobre-ajuste, mediante pruebas y entrenamiento en diferentes temas.

- **Leave One Out:**
  Es una estrategia simple de validación cruzada en donde se crea cada conjunto de entrenamiento tomando todas las muestras con excepción de una, las relacionadas a una etiqueta especifica.

- **Leave-One-Label-Out:**
  LeaveOneOut (o LOLO) es una estrategia de validación cruzada que consiste en que cada conjunto de aprendizaje se crea tomando todas las muestras con excepción de una, el conjunto de prueba es la muestra que se deja fuera. Por lo tanto, para n muestras, tenemos n distintos conjuntos de entrenamiento y n distintos conjuntos de prueba.

[3] Tomemos en cuenta que la palabra "experimento" no pretende denotar solamente para uso académico, porque incluso en los entornos comerciales el aprendizaje automático por lo general comienza experimentalmente [Pedregosa *et al.*, 2011]

- **Leave-P-Out:**
  Esta estrategia de validación cruzada es crea todos los conjuntos de prueba y entrenamiento removiendo las P muestras del conjunto completo. Para n muestras, esto produce $\binom{n}{p}$ de pares para prueba.
- **Leave-P-Label-Out:**
  Es similar a LOLO anterior pero remueve muestras relacionadas a P etiquetas para cada conjunto de prueba-entrenamiento.
- **Label Shuffle Split:**
  La estrategia ShuffleSplit, al igual que k-fold genera un número (definido por el usuario) de divisiones de conjuntos de datos de entrenamiento y prueba independientes. Las muestras se mezclan primero y luego se dividen en un par de conjuntos de entrenamiento y de prueba. Cabe mencionar que si el orden de los datos no es arbitrario (por ejemplo, las muestras con la misma etiqueta son consecutivas), el revolver los datos es esencial para conseguir un resultado de la validación cruzada correcto. Sin embargo, lo contrario puede ser cierto si las muestras no son independientes e idénticamente distribuidas. Por ejemplo, si las muestras corresponden a artículos de noticias, y se necesitan ordenar según su fecha de publicación, entonces revolver los datos probablemente conducirá a un modelo que se sobre-ajuste y a malas métricas de evaluación, es por esto que para este trabajo usamos esta metodología de validación cruzada. En otras palabras, esta metodología de validación cruzada se comporta como una combinación de shuffle split y Leave-P-labels-Out y genera una secuencia de particiones aleatorias en la cual se separa un subconjunto de etiquetas para cada pliegue.

4.9.1. Sobre-ajuste y bajo-ajuste

El sobre-ajuste ocurre cuando un modelo describe un error aleatorio o ruido en lugar de la relación subyacente; es decir, cuando el modelo es excesivamente complejo, presenta demasiados parámetros relacionados al número de observaciones o muestras. Un modelo que ha sido sobre-ajustado generalmente tendrá poder predictivo pobre. El concepto de sobre-ajuste es muy importante en aprendizaje automático, generalmente un algoritmo se entrena usando algún conjunto de ejemplos de entrenamiento donde la salida deseada se conoce como algoritmo de aprendizaje. El algoritmo supone alcanzar un estado donde podrá ser capaz de predecir instancias nunca antes vistas, sin embargo se vuelve peor [Wikipedia, 2016b].

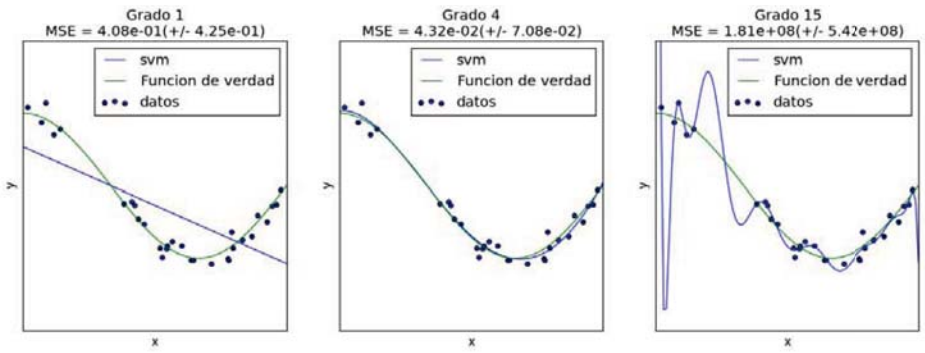


FIGURA 4.7: Ejemplo de como sobrea-justa y bajo-aujusta una máquina de soporte vectorial.

En la Figura 4.7 podemos observar, en el ejemplo generado con scikit-learn, cómo una función lineal o función polinomial de grado 1, no es suficiente para separar los vectores o puntos de entrenamiento, a esto se le conoce como bajo-ajuste. Similarmente podemos observar que con una función polinomial de grado 4 los datos son separados por la máquina de soporte vectorial de forma casi perfecta, mientras que con una función de grado 15 la máquina de soporte vectorial sobre-ajusta a los datos, es decir separa perfectamente los puntos específicos.

4.10. Reportes de clasificación

La función `classification_report` construye un informe de texto que muestra las principales métricas de clasificación. A continuacion, un ejemplo con `target_names` personalizados y etiquetas inferidos [Pedregosa *et al.*, 2011]:

```
1 from sklearn.metrics import classification_report
2 y_true = [0, 1, 2, 2, 0]
3 y_pred = [0, 0, 2, 2, 0]
4 target_names = ['class 0', 'class 1', 'class 2']
5 print(classification_report(y_true, y_pred,
     target_names=target_names))
6               precision    recall  f1-score   support
7
8      class 0       0.67      1.00      0.80         2
9      class 1       0.00      0.00      0.00         1
```

```
10      class 2        1.00       1.00       1.00          2
11
12  avg / total        0.67       0.80       0.72          5
```

En este capítulo se mencionaron y exploraron las herramientas necesarias para poder realizar este trabajo. Principalmente nos basamos en Python debido a que parece ser un ambiente de aprendizaje automático prometedor ya que cuenta con una gran variedad de solidas bibliotecas bien soportadas para resolver problemas de inteligencia artificial, procesamiento de lenguaje natural y análisis estadístico. Se hizo mención de la funcionalidad de Freeling, un marcador de categorías gramaticales con soporte para idioma en español. De la misma forma, se examina el proyecto pandas que nos sirvió para darle estructura relacional a las opiniones que finalmente son clasificadas por scikit-learn.

5 Experimentos: resultados y evaluación

En este capítulo presentamos los resultados al aplicar bigramas morfosintácticos[1] simples como características para entrenar una máquina de soporte vectorial y desarrollar un sistema base de predicción de la orientación semántica de opiniones en idioma español. Posteriormente reportamos el efecto de la negación en la estimación de la orientación semántica. Analizamos el uso de negaciones como características de entrenamiento para métodos supervisados. Aplicamos esta metodología en un corpus de opiniones sobre lavadoras de ropa, luego comparamos el rendimiento de cada negación léxica. Encontramos que la negación tiene un desempeño análogo a usar otros recursos textuales.

5.1. Preprocesamiento de datos

Para obtener las secuencias morfosintácticos definidas en la sección 3.4.3 se generaron distintas expresiones regulares para empatar esos patrones y posteriormente, el programa desarrollado genera archivos del tipo *.csv* como el presentado en la tabla 3.5 con los bigramas propuestos. Decidimos utilizar este formato porque se usa ampliamente por la comunidad de aprendizaje automático; por ejemplo: la gran mayoría de los datos de UCI machine learning repository o de Kaggle se encuentran en formato *.csv* [2] [3].

Una vez que se generaron los archivos *csv*, se preprocesaron con *pandas*. En este trabajo utilizamos algunas funciones de pandas para leer el corpus y presentárselo al algoritmo

[1] Recordemos que la morfosintaxis es un conjunto de elementos y reglas que permiten construir estructuras lingüísticas con sentido y sin ambigüedad mediante la consideración de la el orden jerárquico de los distintos tipos de constituyentes sintácticos.

[2] http://archive.ics.uci.edu/ml/

[3] https://www.kaggle.com/competitions

de máquinas de soporte vectorial. Posteriormente estos datos fueron separados bajo un esquema de validación cruzada con la finalidad de prevenir el sobre ajuste del modelo.

5.2. Clasificación

En [Joachims, 1998] se menciona que una ventaja de las máquinas de soporte vectorial es la gran variedad de funciones kernel que pueden usarse para la clasificación. Por otro lado, también menciona que se puede generalizar aún en presencia de muchas características con un amplio margen, usando funciones de nuestro espacio de hipótesis. Lo anterior infiere el uso de heurísticas como Grid search para la optimización de un conjunto de hiper-parámetros para un estimador en el espacio de un algoritmo de aprendizaje.

Para una tarea de clasificación es necesario separar los datos entre conjunto de entrenamiento y conjunto de prueba, en nuestro caso separamos el corpus de opiniones en 67 % para entrenamiento y 33 % para prueba. Cada ejemplo o instancia es asociado a una clase, categoría o etiqueta, es decir el 67 % de los datos de entrenamiento fueron etiquetados con la clase correspondiente (e.g. 1 a 5 estrellas) y el 33 % de los datos se usó para probar.

5.2.1. Evaluación

La precisión de un sistema de análisis de sentimiento se basa en obtener información para identificar las interpretaciones positivas o negativas que emiten las opiniones de las personas. Esto se mide por la precisión y exhaustividad.

Sin embargo, en el trabajo de [Ogneva, 2010] se menciona que si le solicitáramos a personas que clasificaran opiniones como positivas o negativas, solamente el 79 % de las veces harían una clasificación correcta. Por lo tanto, resulta interesante mencionar que un programa que tiene esa precisión está resolviendo la tarea al igual que un ser humano. Por otro lado en [Roebuck, 2012], se menciona que si un programa fuera “correcto” el 100 % de las veces, los seres humanos todavía no estarían de acuerdo con el 20 % de las veces que el programa clasificara las opiniones, ya que los seres humanos siempre están en desacuerdo en relación a distintos tipos de respuestas.

Medidas más sofisticadas se pueden aplicar, pero la evaluación de los sistemas de minería de opiniones o análisis de sentimiento sigue siendo un asunto complejo. Tradicionalmente, la tasa de exactitud se ha usado como una medida empírica de criterio de evaluación el rendimiento de un clasificador, al igual que F1-score, precisión y exhaustividad [Manning *et al.*, 2008]. Sin embargo, en un esquema de datos no balanceados el uso de la exactitud como métrica de rendimiento no es apropiada [López *et al.*, 2013], debido a que no es capaz de distinguir entre los números de ejemplos correctamente clasificados en distintas clases. En dominios no balanceados, la evaluación del rendimiento de los clasificadores debe llevarse acabo usando métricas específicas para poder tomar en cuenta la distribución de la clase. Los autores antes mencionados propusieron obtener las siguientes cuatro métricas para medir el rendimiento de la clasificación para clases positivas y negativas de forma independiente:

TABLA 5.1: Métricas para un dominio no balanceado

Metrica	Formula	Porcentaje
Verdaderos positivos	$TP_{rate} = \frac{t_p}{t_p+f_n}$	de instancias positivas correctamente clasificadas
Verdaderos negativos	$TN_{rate} = \frac{t_n}{f_p+t_n}$	de instancias negativas correctamente clasificadas
Falsos positivos	$FP_{rate} = \frac{f_p}{f_p+t_n}$	de instancias negativas incorrectamente clasificadas
Falsos negativos	$FN_{rate} = \frac{f_n}{t_p+f_n}$	de instancias positivas incorrectamente clasificadas

La Tabla 5.4 muestra la formulación para identificar verdaderos positivos (t_p), verdaderos negativos (t_n), falsos positivos (f_p) y falsos negativos (f_n). Para posteriormente evaluar la calidad de la clasificación se emplearon las siguientes métricas:

- **F1-score:**
  Puede ser interpretado como un promedio balanceado entre la precisión y el exhaustividad, una F1-score alcanza su mejor valor en 1 y su peor valor en 0. La contribución relativa de precisión y exhaustividad al F1-score son iguales.

- **Exhaustividad:**
  Es la capacidad que tiene un estimador de encontrar todas las muestras positivas. El exhaustividad es el radio $\frac{t_p}{t_p+f_n}$ donde t_p es el número de verdaderos positivos y f_n es el número de falsos negativos.

- **Precisión:**
  Intuitivamente podemos decir que es la capacidad que tiene un estimador de no etiquetar como positiva una muestra que es negativa. El radio de precisión: $\frac{t_p}{t_p+f_p}$ donde t_p es el número de verdaderos positivos y f_p el número de falsos positivos.

En cada entrenamiento obtuvimos la matriz de confusión respectiva. A continuación presentamos un ejemplo usando la característica negativa 4 mencionada en la Sección 3.4.3 ($\text{No}_{\text{ADVERBIO}}$-verbo).

Una vez que la clasificación se realiza, podemos generar con la ayuda de scikit-learn, una matriz de confusión como se muestra en la Tabla 5.2. Podemos observar, si sumamos la fila 1 de la matriz, tenemos que hay 71 opiniones con la clase 1 o con una estrella. Esto significa que es el modelo fue exitoso en identificar correctamente 54 opiniones en la clase 1 (1 estrella), pero 17 fueron marcadas con la clase 5 (5 estrellas). Análogamente si observamos la segunda fila marcado con 2, hubo 43 opiniones en la clase 2, pero 36 de ellas fueron clasificados correctamente, mientras que 7 fueron marcados incorrectamente.

En otras palabras el clasificador predijo 1 en la clase 4 y 6 en la clase 5, lo mismo se cumple para las demás filas (hacia abajo). Por otro lado si el clasificador hubiera tenido 100 % de precisión hubiera producido una matriz diagonal, es decir hubiera predicho cada opinión de bigramas en su clase correspondiente.

	1	2	3	4	5		
1	54	0	0	0	17	=	71
2	0	36	0	1	6	=	43
3	0	0	66	5	18	=	89
4	0	0	0	273	15	=	288
5	0	0	0	0	367	=	367
Total						=	858

TABLA 5.2: Una matriz de confusión generada por scikit-learn

En cuanto al reporte de la clasificación podemos observar en la Tabla 5.3. la precisión y exhaustividad. Tenemos 71 opiniones en la primer clase 1, el clasificador pudo reconocer 54 instancias correctamente. Es decir 54/71 = 0.76, que es nuestro exhaustividad para la clase 1. Solo hay una entrada de la matriz con 54, los demás ceros en la matriz de

la Figura 5.1. significan que el clasificador marco 54 vectores en la clase 1. La precisión es 54/54 = 1.

Si observamos en la Figura 5.3. la columna marcada con 5, podemos notar que hay elementos dispersos o con ceros en las filas, 367 de ellos fueron marcados correctamente, el resto es incorrectos. Eso fue lo que redujo la precisión. Para F1-score y exhaustividad es con la media armónica.

Reporte de clasificación:				
Clase	Precisión	exhaustividad	F1-score	Support
1	1.0	0.76	0.86	71
2	1.0	0.84	0.91	43
3	1.0	0.74	0.85	89
4	0.98	0.95	0.96	288
5	0.87	0.1	0.93	367
avg/total	0.94	0.93	0.93	858

TABLA 5.3: Un reporte de clasificación generado por scikit-learn

Por otro lado como partimos los datos en 33 % para probar y 67 % para entrenar. Tenemos que el 33 % de 2599 (instancias de opiniones de bigramas) es 857.67, si redondeamos nos da 858.

5.2.2. Entrenamiento

En [Joachims, 1998] se menciona que las máquinas de soporte vectorial son un buen algoritmo para clasificar texto. El texto es discreto, disperso y altamente dimensional. Típicamente un clasificador que usa texto puede tener alrededor de entre 10,0000 y 1,000,000 características. El autor, identificó que inherentemente las máquinas de soporte vectorial tienen las siguientes características al ser entrenadas con características textuales:

- Espacios de gran dimensión como entrada al algoritmo.
- Pocas características irrelevantes.
- Los vectores hechos a partir de los documentos son dispersos.

- Muchos problemas de clasificación textual son linealmente separables.

Por otro lado, el sistema se configuró con tres funciones kernel distintas (e.g. lineal, polinomial, función de base radial) y se fueron agregando los bigramas mencionados en la Sección 3.4.3. y se evaluó el rendimiento con las métricas mencionadas en la Sección 5.3.1. Finalmente concluimos que la mejor configuración era con $C = 1$ y un kernel lineal, debido a que presentó un buen desempeño en términos de F1-score, exhaustividad y precisión.

5.3. Resultados del sistema base

A continuación en la Tabla 5.2. presentamos los resultados de clasificar las opiniones del corpus de [Galicia-Haro & Gelbukh, 2014] mediante distintos grupos de características de entrenamiento. El entrenamiento fue realizado con scikit-learn. Usamos una metodología de clasificación multi-clase con un kernel lineal y con un parámetro $C = 1$ que obtuvo los mejores resultados en comparación con un kernel polinomial y de función de base radial.

Conjunto	Características	Métrica	Valor
1	sustantivo-adjetivo	F1-score exhaustividad Precision	0.8419 0.8287 0.8556
2	sustantivo-adjetivo verbo-adverbio	F1-score exhaustividad Precision	0.9287 0.9266 0.9309
3	sustantivo-adjetivo verbo-adverbio adverbio-adjetivo	F1-score exhaustividad Precision	0.9258 0.9230 0.9286
4	sustantivo-adjetivo verbo-adverbio adverbio-adjetivo adjetivo-adverbio	F1-score exhaustividad Precision	0.9339 0.9312 0.9366

TABLA 5.4: Rendimiento de la clasificación al usar bigramas lingüísticos simples

El conjunto 1. considera únicamente el bigrama Sustantivo-adjetivo que cumple una relación sintáctica, ya que como lo indicamos previamente se verificaron concordancias de género y número. La razón para iniciar con este bigrama es que expresa atributos de sustantivos que corresponderían a atributos de características del producto. La evaluación obtenida en cuestión de precisión fue de 0.8556.

El conjunto 2. adiciona el bigrama verbo-adverbio mediante el cual se puede expresar el modo en que se realiza la acción descrita por el verbo. La adición de este bigrama mejora la clasificación en casi un 8 %.

El conjunto 3. adiciona a los bigramas anteriores el correspondiente a adverbio-adjetivo. Consideramos este bigrama porque cuando el adverbio se une con un adjetivo, su función semántica es calificadora o cuantificadora. Sin embargo, su aportación a la clasificación es negativa en relación a los otros bigramas.

El conjunto 4. adiciona el bigrama Adjetivo-adverbio, el cuál no corresponde a un orden muy común en el español. Sin embargo, como algunas de estas secuencias se encuentran presentes en algunas opiniones aportan un incremento del 1 %. Estos resultados fueron publicados en [Palomino-Garibay & Galicia-Haro, 2015].

Cabe mencionar que para comparar el rendimiento del sistema con otro clasificador, usamos un clasificador de prueba [4] que usa una estrategia de clasificación más frecuente, es decir, predice con la clase más frecuente. De esta forma se obtuvieron los siguientes resultados con el sistema base 0.330 para F1-score, para exhaustividad: 0.330 y para precisión: 0.329. Los resultados de todos los experimentos se muestran en la Tabla 5.5, donde se observa que mejoran significativamente los resultados de la clasificación base.

5.4. Aplicación de bigramas negativos

Los resultados de la aplicación de bigramas negativos para el método supervisado pueden encontrarse en la Tabla 5.3.. La primer columna muestra los cuatro patrones que corresponden a los bigramas simples. En la segunda columna podemos observar en cada fila, la medida de valores obtenidos para cada uno de los patrones para la negación.

Haciendo una comparación entre la última linea de los resultados del sistema base en la Tabla 5.2., para cada fila de los valores medidos obtenidos en la Tabla 5.3., podemos

[4] Dummy classfier

notar que el único patrón que no incrementa los resultados es el que corresponde a $no_{ADVERBIO}$-$verbo_{AUX\_PASADO\ PARTICIPIO}$ en la primer fila.

Features		Metric	Values
Noun-Adjective	+ no_{ADVERB}-$verb_{AUX\_PAST\ PARTICIPLE}$	F1score	0.9315
		Recall	0.9289
		Precision	0.9342
+ Verb-Adverb	+ $ninguno_{LEMMA\_DET}$ -noun	F1score	0.9406
		Recall	0.9382
		Precision	0.9431
	+ $jamás_{ADVERB}$-verb	F1score	0.9380
		Recall	0.9359
		Precision	0.9401
+ Adverb-Adjective	+ $nunca_{ADVERB}$ -verb	F1score	0.9524
		Recall	0.9510
		Precision	0.9537
	+ no_{ADVERB} -verb	F1score	0.9407
		Recall	0.9382
		Precision	0.9432
+ Adjective-Adverb	+ $nada_{PRONOUN}$ -adjective	F1score	0.9501
		Recall	0.9487
		Precision	0.9515
	+ no_{ADVERB}-pronoun-verb	F1score	0.9395
		Recall	0.9370
		Precision	0.9420

TABLA 5.5: Rendimiento al añadir indicios de negación.

El rango de mejoras fue de 0.41 % a 1.85 %. Dos patrones superan el 1 %: $nunca_{ADVERBIO}$-verbo y $nada_{PRONOMBRE}$-adjetivo. El primero apareció en 77 reseñas positivas, mientras que el segundo en 17 opiniones negativas, $nada_{PRONOMBRE}$-adjetivo apareció en 401 opiniones positivas y 90 opiniones negativas.

5.5. Discusión de resultados

En relación al método supervisado empleado en para este trabajo, nuestros resultados al usar todos los bigramas de negación se describen en la Tabla 5.4, la mejora en contraste a la última a la última fila de los resultados para el sistema base fue de 1.09 %.

Features		Metric	Value
Noun-Adjective Verb-Adverb Adverb-Adjective Adjective-Adverb	+ $ninguno_{LEMMA\_DET}$ -noun $jamás_{ADVERB}$-verb $nunca_{ADVERB}$-verb no_{ADVERB}-verb $nada_{PRONOUN}$ -adjective no_{ADVERB}-pronoun-verb	F1score Recall Precision	0.9448 0.9429 0.9467

TABLA 5.6: Resultados al emplear todos los bigramas de negación.

No contrastamos los resultados finales con otros trabajos, debido a que han hecho clasificación binaria en lugar de clasificación multi-clase. En relación a la clasificación multi-clase, la matriz de confusión para los resultados mostrados en la Figura 5.4 reveló que los peores resultados fueron aquellas para las opiniones que estaban etiquetadas con 3 estrellas. 25 % de las opiniones fueron erróneamente clasificadas como opiniones con evaluaciones 4 o 5 estrellas. Para la clase 5, no hubo errores de clasificación de opiniones y para la clase 4, 5 % de las opiniones fueron clasificadas como de 5 estrellas. Hubo más errores de clasificación para las clases 1 y 2 con 16 % y 18 % respectivamente. La mayoría de opiniones mal clasificadas fueron asignadas con 4 y 5 estrellas. Estos resultados fueron publicados en [Galicia-Haro *et al*., 2015].

6 Conclusiones

A continuación se presentan las conclusiones, una recapitulación del contenido de este trabajo, las consideraciones finales de los experimentos realizados y acotamos las lineas de investigación futuras.

6.1. Resumen

En este trabajo presentamos las distintas consideraciones que deben tomarse en cuenta para resolver una tarea de análisis de sentimiento sobre un corpus de opiniones en idioma español, así como el efecto de la negación para predecir de forma supervisada la orientación semántica de opiniones textuales en idioma español.

Presentamos los resultados del sistema de clasificación base con bigramas simples conformados por patrones lingüísticos que a su vez fueron tomados como características para el método de máquinas de soporte vectorial. También presentamos los resultados al usar bigramas basados en negación léxica como características de entrenamiento para máquinas de soporte vectorial. Encontramos que la inclusión de la negación como características de entrenamiento para el método de máquinas de soporte vectorial producen mejores resultados para la clasificación de la polaridad u orientación semántica de opiniones sobre lavadoras automáticas. Es un hecho que la negación en lenguaje natural es compleja, por lo tanto, el tratamiento adecuado de ésta puede resultar en mejoras en términos de precisión.

Con el crecimiento imperante de diversos medios de comunicación electrónica, y la interacción que se deriva de los usuarios de las mismas, se crean grandes cantidades de información cuyas aplicaciones pueden servir a propósitos tanto comerciales como

académicos y sociales. No obstante, el manejo e interpretación de dichos datos se erige como un reto para todo aquel que busca datos precisos sobre algún tema en particular.

En cuanto a su faceta comercial es útil conocer qué opinión tienen los usuarios sobre un producto específico o saber que fallas quejan a los usuarios de un servicio. En cambio, desde la perspectiva social es importante conocer el grado de aceptación del público hacia la implementación de nuevas medidas políticas o bien fomentar la acción ciudadana en pos de mejorar la vida local; en contraste, por el lado académico el manejo estas herramientas pueden servir para hacer sondeos sobre cantidades mayores de sujetos y facilitar el acceso a información de primera mano.

Debido a las posibilidades que puede brindar el desarrollo de sistemas computacionales que sean capaces de procesar directamente la información que comparten los usuarios, a través de opiniones; es necesario realizar investigaciones que nos permitan determinar si una opinión específica es favorable o es desfavorable.

En este sentido el presente trabajo abordó la problemática de derivar la orientación semántica o polaridad de opiniones mediante un método automático de clasificación, sobre un corpus de opiniones comerciales, escrito en español.

En el primer capítulo se inició a partir de la definición dada por [Liu, 2012], quien señala que el termino análisis de sentimiento aparece probablemente por primera vez en [Nasukawa & Yi, 2003], en donde se propone una metodología para el análisis de sentimiento asociado con polaridades positivas o negativas, de temas específicos en un documento. Así mismo se exploraran algunos precedentes sobre la minería de opiniones y análisis de sentimiento en español para desarrollar el marco teórico.

En el capítulo dos se presentaran los materiales utilizados para calcular la orientación semántica de acuerdo al marco teórico desarrollado. Además se describió el trabajo realizado para explotar la información, de dichos materiales, las expresiones que portan sentimiento.

El capítulo tres abordó el cómo la orientación semántica o polaridad de una palabra refiere a la dirección que la palabra deriva, a partir de la norma hacia su grupo o campo semántico [Hatzivassiloglou & Wiebe, 2000].

En el capítulo cuatro se mencionaron los materiales y herramientas que preceden y sirven a la investigación, y en concreto del caso de Python el cual es un lenguaje de programación multi-paradigma conocido por su simplicidad y poder. Generalmente dicho programa es usado como un “lenguaje de unión” o “lenguaje de aglutinamiento”

para diversos componentes escritos en otros lenguajes. La interfaz entre C/C++ y Python provee una sólida integración, mientras que los componentes definidos en Python pueden usarse por el núcleo de C/C++ [Demšar *et al.*, 2004].

Finalmente, en el capítulo cinco presentamos los resultados al aplicar bigramas morfosintácticos simples como características para entrenar un clasificador máquinas de soporte vectorial y desarrollar un sistema base de predicción de la orientación semántica de opiniones. Posteriormente reportamos el efecto de la negación en la estimación de la orientación semántica. Analizamos el uso de negaciones como características de entrenamiento para métodos de aprendizaje automático supervisado. Aplicamos esta metodología en el corpus de opiniones sobre lavadoras de [Galicia-Haro & Gelbukh, 2014], luego comparamos el rendimiento de cada negación léxica. Encontramos que la negación tiene un desempeño análogo al usar diversos recursos textuales.

6.2. Consideraciones finales

Finalmente podemos concluir que el procesar y explotar lenguaje natural es difícil y tiene muchas complicaciones:

- Al ser datos que carecen de estructura o que se encuentran fuera de una base de datos, es necesario darles un tratamiento adecuado que permita la correcta explotación de estos.
- Al ser texto procedente de internet, inherentemente es ambiguo, implícito, complejo y difícil de procesar. Además, es común que en este tipo de datos exista uso de lenguaje informal.
- Otro factor que determina la dificultad de procesar este tipo de datos es que al ser comentarios las opiniones cuentan con abreviaturas, falta de mayúsculas, mala ortografía, mala puntuación y gramática pobre.
- La carencia de córpora en idioma español, es otra limitación para poder procesar de forma automática opiniones, ya que sin un corpus de entrenamiento no es posible clasificar o predecir la polaridad de una opinión.

6.3. Trabajo futuro

El análisis de sentimiento al ser un problema con muchas particularidades, sigue siendo un tema de investigación activo y existen muchos problemas abiertos en este tema, por ejemplo:

1. Profundizar en el análisis de las emociones contenidas en las opiniones estableciendo niveles de polaridad para estas, ya que la clasificación positiva negativa es una análisis muy simple, uno de los desafíos futuros es responder con exactitud a las siguientes preguntas ¿Cómo extraer las emociones y medirlas?, ¿Qué cantidad de odio, felicidad o tristeza hay dentro de una opinión?.

2. Reconocer entidades nombradas, por ejemplo: *"Hace algunas horas pasé por Benjamin Franklin"*. ¿Benjamin Franklin es un personaje histórico o el nombre de una avenida?.

3. Resolución anáfora - el problema de saber a lo que un pronombre o un sintagma nominal se refiere. Por ejemplo, *"Vimos la película y fuimos a cenar, era horrible."*. ¿A que se refiere?

4. Sarcasmo: Si no sabemos mucho sobre el autor, entonces no tenemos idea de si "malo" significa bueno o malo.

5. ¿Cómo analizar frases o párrafos subjetivos?. A veces, incluso para los seres humanos es muy difícil ponerse de acuerdo sobre el sentimiento de este tipo de textos.

6. ¿Como detectar opiniones engañosas o SPAM?. Es decir, opiniones que hagan juicios falsos sobre un producto, con la intensión de hacer que este producto se venda o no se venda más.

7. Detectar intensión en una opinión, puede ser de gran ayuda para mejorar los sistemas de recomendación actuales. Por ejemplo, *"La lavadora no me gustó, necesito una opción que gaste menos luz"*. Sería interesante a partir de esa opinión inducir una lista que recomiende productos similares, o enviar ese comentario a un soporte técnico especializado en esa área.

Finalmente, recientemente se han adoptado nuevas técnicas basadas en aprendizaje profundo para realizar análisis de sentimiento en idioma ingles. Por ejemplo, en el trabajo

de [Kalchbrenner *et al.*, 2014] los investigadores obtuvieron buenos resultados analizando sentimiento con un enfoque basado en redes neuronales convolucionales, que a su vez usa un operador k-max pooling como una función de submuestreo. Otros resultandos bastantes interesantes derivados de esta investigación son que este enfoque induce un grafo de características que representa los componentes más importantes de una opinión y que este enfoque supero en cuestión de rendimiento a otros algoritmos, en particular a las maquinas de soporte vectorial en los corpus *Twitter sentiment dataset* y *movie review dataset*.

Recientemente se liberó TensorFlow bajo una licencia Apache 2.0 en Noviembre del 2015. En [Abadi *et al.*, 2016] se especifica que TensorFlow es una interfaz para expresar y ejecutar algoritmos de aprendizaje automático. En particular cuenta con una implementación de redes neuronales convolucionales. Otra ventaja de TensorFlow es que es una biblioteca flexible y puede usarse para expresar una gran variedad de algoritmos, por ejemplo: algoritmos de entrenamiento e inferencia para modelos de redes neuronales profundas. Esta biblioteca se ha usado en una gran variedad de sistemas en muchos subcampos de las ciencias de la computación, desde reconocimiento del habla, visión por computadora, robótica, recuperación de información, procesamiento de lenguaje natural, extracción de información geográfica y descubrimiento computacional de medicamentos.

Finalmente, basados en los resultados de los trabajos anteriormente mencionados, un trabajo futuro sería explorar el rendimiento de las redes neuronales convolucionales, bajo la implementación de TensorFlow, creando encajes de palabras a partir del mismo corpus para realizar tareas de análisis de sentimiento en idioma español.

A Apéndice

A continuación presentamos las partes mas significativas del código que se usó para realizar este trabajo. Sin embargo, el código en su totalidad puede encontrarse la siguiente liga: https://github.com/alonsopg.

A.1. Preprocesamiento de datos

A.1.1. Generación de bigramas

- Bigrama sustantivo-adjetivo:

```
sust_adj = r'(\S+)\s+(NC\S+).*\n.*\s(\S+)\s+(AQ\S+)'
```

- Bigrama verbo-adverbio:

```
verb_adv = r'(\S+)\s+(VM\S+).*\n.*\s(\S+)\s+(RG\S*)'
```

- Bigrama adverbio-adjetivo:

```
adv_adj = r'(\w+)\s*(RG)[^\n]*\n[^\n]*?(\w+)\s*(AQ\w*)'
```

- Bigrama adjetivo-adverbio:

```
adj_adv = r'(\S+)\s*(RG)[^\n]*\n[^\n]*?(\S+)\s*(AQ\S*)'
```

A.2. Recuperación de bigramas

```

# -- coding: utf-8 --

import glob, os, re
opinions_directory =
   '/Users/user/Desktop/OpinionsTAG_txt_utf-8/'

sust_adj = r'(\S+)\s+(NC\S+).*\n.*\s(\S+)\s+(AQ\S+)'

verb_adv = r"(\S+)\s+(VM\S+).*\n.*\s(\S+)\s+(RG\S*)"

adv_adv=r'(\S+)\s+(RG\S*).*\n.*[^\S\n](\S+)[^\S\n]+(RG\S*)'

adv_adj = r'(\w+)\s*(RG)[^\n]*\n[^\n]*?(\w+)\s*(AQ\w*)'

def preprocesando_corpus(directory, a_regex):
    for filename in
   sorted(glob.glob(os.path.join(directory, '*.txt'))):
        with open(filename, 'r') as file:
            important_stuff = re.findall(a_regex,
   file.read())
            pre_bigrama = [x[::2] for x in important_stuff]
            #Todo aqui hay que componer el UTF-8
            bigrama = [''.join(
                c for c in ' '.join(str(v) for v in
   pre_bigrama)
                if c not in "',()")]
            yield filename.split('/')[-1] + ', '
   +str(bigrama)


```

```
27 print "\n\nEstos son los bigramas sustantivo/adj de todas
      las opiniones:\n\n", \
28     #list(bigramas_sust_adj)
29
30
31 print '\n'.join([ str(myelement) for myelement in
      bigramas_sust_adj]),'\n'
32
33 bigramas_verb_adv =
      preprocesando_corpus(opinions_directory,verb_adv)
34 print "\n\nEstos son los bigramas verbo/adverbio de todas
      las opiniones:\n\n",
35
36 print '\n'.join([ str(myelement) for myelement in
      bigramas_verb_adv]),'\n'
37
38 bigramas_adv_adv =
      preprocesando_corpus(opinions_directory,adv_adv)
39 print "\n\nEstos son los bigramas adverbio/adverbio de
      todas las opiniones:\n\n",
40
41 print '\n'.join([ str(myelement) for myelement in
      bigramas_adv_adv] ),'\n'
42
43
44 bigramas_adv_adj =
      preprocesando_corpus(opinions_directory,adv_adj)
45 print bigramas_adv_adj
46 print "\n\nEstos son los bigramas adverbio/adj de todas las
      opiniones:\n\n",
47
48 print '\n'.join([ str(myelement) for myelement in
      bigramas_adv_adj] ),'\n'
```

A.2.1. Generación de bigramas negativos

- Bigrama negativo:

```
bigrama_neg_1 = r'^[\w]* (\w* RN)
   \d(?:\.\d*)?$\s^[^\s]* (\w* VA[^\s]*)
   \d(?:\.\d*)?$\s^[^\s]* (\w* VM[^\s]*) \d(?:\.\d*)?$'
lista = [tuple([j.split()[0] for j in i]) for i in
   weird_triple]
```

- Bigrama negativo:

```

bigrama_neg_2 = r'(?m)^.*?\b(ninguno)\s+(\S+)\s+[0-9.]
+\n.*?\s(\S+)\s+(NC\S+)\s+[0-9.]+$'

lista = [t[::2] for t in bigrama_neg_2]
```

- Bigrama negativo:

```
bigrama_neg_3 = '(?m)^.*?\b(jamas)\s+(\S+)\s+[0-9.]+\n
.*?\s(\S+)\s+(VM\S+)\s+[0-9.]+$'
```

- Bigrama negativo:

```
bigrama_neg_4
   =r'(nunca)\s(\S+)\s\d\S*\n\S+\s(\S+)\s(VM\S+)', s
```

- Bigrama negativo:

```
bigrama_neg_5 = r'(?m)^.*?\b(no)\s+(\S+)\s+[0-9.]+\n.*
?\s(\S+)\s+(VM\S+)\s+[0-9.]+$'

lista = [t[::2] for t in bigrama_neg_5]
```

- Bigrama negativo:

```
bigrama_neg_6 = r'(?m)^.*?\b(nada)\s+(\S+)\s+[0-9.]+\n
.*?\s(\S+)\s+(AQ\S+)\s+[0-9.]+$'
```

- Bigrama negativo:

```
bigrama_neg_7 = r'\n?\s*\S+\s+(\w+\W+RN\w*)[^\n]*[^\n]
*?\n\s*\S+\s+(\w+\W+PP\w*)[^\n]*[^\n]*
?\n\s*\S+\s+(\w+\W+VM\w*)[^\n]*'
```

A.3. Recuperación de bigramas negativos

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

import glob, os, re

directory = '/Users/user/Desktop/OpinionsTAG_txt_utf-8/'


#regex_ = r'.*?\b(nunca)\s+(\S+)\s+[0-
9.]+[\r\n]+\S+\s+(\S+)\s+(VM)\s+[0-9.]+'
#regex_ = r'(nunca)\s(\S+)\s\d\S*\n\S+
\s(\S+)\s(VM\S+)'

#regex_ = r'(?: *\S+ (\S+ RN\S*))\n(?:
*\S+ (\S+ VA\S*))\n(?: *\S+ (\S+ VM\S*))\n'

def retrive(directory, a_regex):
    for filename in glob.glob(os.path.join(directory,
   '*.txt')):
        with open(filename, 'r') as file:
            important_stuff = re.findall(a_regex,
   file.read(), re.S)
```

```
20              my_list = [tuple([j.split()[0] for j in i]) for
    i in important_stuff]
21              print my_list
22
23 bigramas_tigrama = retrive(directory,regex_)
24 print "\n\nEstos son los bigramas adverbio/adverbio de
    todas las opiniones:\n\n",
25
26 print '\n'.join([str(myelement) for myelement in
    bigramas_tigrama]),'\n'
```

A.3.1. Generación de corpus

- Ordenando por orden alfabético las opiniones.

```
1 from string import strip
2 with open(raw_input("give me the file")) as f:
3     lines = sorted(map(strip, f))
4
5     with open(r'/Users/user/Desktop/Ar
6     reglando_el_corpus/verificando/sta
7     rs_sorted.txt', 'w') as a:
8         a.write('\n'.join(lines))
```

- Revisando que los nombres de las etiquetas coincidan con los nombres de las opiniones.

```
1 import csv
2
3 file1 =
    '/Users/user/Desktop/Arreglando_el_corpus/Arreglando
    el corpus 2/verificando/stars_sorted.csv'
4 file2 =
    '/Users/user/Desktop/Arreglando_el_corpus/Arreglando
    el corpus 2/verificando/bigrama_neg_neg.csv'
```

```
5
6  with open(file1) as fp1:
7      root = csv.reader(fp1, delimiter=',')
8      rows1 = {}
9      for i in root:
10         rows1[i[0]]=i
11     if "id" in rows1:
12         del rows1["id"]
13
14 with open(file2) as fp1:
15     root = csv.reader(fp1, delimiter=',')
16     rows2 = {}
17     for i in root:
18         rows2[i[0].split(".")[0]]=i
19     if "id" in rows2:
20         del rows2["id"]
21
22 result =
      set(rows1.keys()).intersection(set(rows2.keys()))
23
24 print "Same Id :", list(result)
```

- Transformación a formato *.csv*

```
1  # -- coding: utf-8 --
2  import pandas as pd
3  pd.set_option('display.max_rows', 3000)
4
5  df1=pd.read_csv('/Users/user/Desktop/MICAI_2/positivos/
6  sust_adj+verb_adv+adv_adj+adj_adv.csv',sep=',')
7  .drop('label',1)
8
9  df2=pd.read_csv('/Users/user/Desktop/MICAI_2/negativos/
10 bigrama_negativo_1.csv',sep=',').drop('id',1)
11
12 label = pd.read_csv('/Users/user/Desktop/
13 CORPUS_LISTO/labels.csv',sep=',').drop('id',1)
14 new_df.reset_index(drop=True)
```

```
15 new_df.to_csv('/Users/user/Desktop/MICAI_2/negativos_2/
16 +bigrama_neg_1.csv',
17               sep=',', encoding='utf-8', index=False)
```

A.4. Clasificación

A.4.1. Clasificando con máquinas de soporte vectorial

- Clasificación con SVMs

```
1  # -- coding: utf-8 --
2  #Aqui es con los 4 features juntos
3
4
5  from sklearn.feature_extraction.text import
      TfidfVectorizer
6  import numpy as np
7  tfidf_vect= TfidfVectorizer(use_idf=True,
      smooth_idf=True, sublinear_tf=False,
      ngram_range=(2,2))
8  from sklearn.cross_validation import train_test_split,
      cross_val_score
9
10 import pandas as pd
11
12
13 df = pd.read_csv('path',header=0,sep=',', names=['id',
      'content', 'label'])
14
15
16 ##################Algunos datos extra:
17 #imprimimos las clases y el numero de instancias del
      dataset
18 print "\nImprimimos las clases y su contenido:\n",
      df.count()
```

```
19
20 #Ahora como se ve el corpus:
21 print "\nImprimimos como se ve todo el corpus\n",
      df.head()
22 print "\n Imprimimos el numero de opiniones por
      clase\n", \
23     df['label'].value_counts()
24
25 print "\n imprimimos como se ven las
      opiniones\n",df['content'].head(10)
26
27 #Describimos el dataset:
28
29 print "\n Imprimimos la descripcion del dataset\n",
      df['label'].describe()
30
31 print "\nImprimimos cuantas opiniones hay por
      etiqueta\n", \
32     df['label'].value_counts()
33
34 print "\n****\n",
      df['label'].value_counts()/df['label'].count()
35
36 X = tfidf_vect.fit_transform(df['content'].values)
37 y = df['label'].values
38
39
40 from sklearn import cross_validation
41 X_train, X_test, y_train, y_test =
      cross_validation.train_test_split(X,
42                                                    y,
      test_size=0.33)
43
44 #first svm
45 from sklearn.svm import SVC
46 wclf = SVC(kernel='linear', class_weight={5: 10}, C= 1)
47
48
49 # class_weight : {dict, 'auto'}, optional
```

```
50 # Set the parameter C of class i to class_weight[i]*C
      for SVC.
51 # If not given, all classes are supposed to have weight
      one.
52 # The 'auto' mode uses the values of y to automatically
      adjust
53 # weights inversely proportional to class frequencies.
54
55 # Esto significa que cada clase que tienes(en las
      clases)
56 # obtiene un peso igual a uno dividido el numero de
      veces
57 # que la clase aparece en los datos(y), por lo tanto las
58 # clases que tienen mayor frecuencia van a tener menos
      peso.
59 # Esto entonces se divide por la media de todas las
      frecuencias de
60 # clase. La ventaja es que usted ya no tiene que
      preocuparse por
61 # ajustar los pesos de clase por uno mismo: esto
      ídebera ser bueno
62 # para la ímayora de aplicaciones.
63
64 #Si vemos en el ócdigo fuente, pora None, el peso
65 # áest lleno de unos,
66 # por lo que cada clase tiene el mismo peso.
67
68 wclf.fit(X, y)
69 weighted_prediction = wclf.predict(X_test)
70
71 #w = clf.coef_[0]
72 #a = -w[0] / w[1]
73 #xx = np.linspace(-10, 10)
74 #yy = a * xx - clf.intercept_[0] / w[1]
75
76
77 clf = SVC(kernel='linear', C=1)
78 clf.fit(X, y)
79 prediction = clf.predict(X_test)
```

```
80
81 # ww = wclf.coef_[0]
82 # wa = -ww[0] / ww[1]
83 # wyy = wa * xx - wclf.intercept_[0] / ww[1]
84 #
85 # # plot separating hyperplanes and samples
86 # import matplotlib.pyplot as plt
87 # h0 = plt.plot(xx, yy, 'k-', label='no weights')
88 # h1 = plt.plot(xx, wyy, 'k--', label='with weights')
89 # plt.scatter(reduced_data[:, 0], reduced_data[:, 1],
       c=y, cmap=plt.cm.Paired)
90 # plt.legend()
91 #
92 # plt.axis('tight')
93 # plt.show()
94
95
96 #auto
97 auto_wclf = SVC(kernel='linear', C=1)
98 auto_wclf.fit(X, y)
99 auto_weighted_prediction = auto_wclf.predict(X_test)
100
101 #metrics:
102
103 from sklearn.metrics import precision_score, \
104     recall_score, confusion_matrix,
       classification_report, \
105     accuracy_score, hamming_loss,
       jaccard_similarity_score, \
106     fbeta_score, f1_score
107
108 print "************* No Balanceado: ************\n"
109 print 'Accuracy:', accuracy_score(y_test, prediction)
110 print 'F1 score:', f1_score(y_test, prediction)
111 print 'Score:', clf.score(X_train, y_train)
112 print 'Recall:', recall_score(y_test, prediction)
113 print 'Precision:', precision_score(y_test, prediction)
114 print 'Hamming loss:',hamming_loss(y_test,prediction)
115 print 'Jaccard similarity:',jaccard_similarity_score(
```

```
116 y_test,prediction)
117 print 'F-Beta Score:',fbeta_score(y_test, prediction,
       average='macro', beta=0.5)
118 print '\nClasification report:\n',
       classification_report(y_test, prediction)
119 print '\nConfussion matrix:\n',confusion_matrix(y_test,
       prediction)
120
121
122 import matplotlib.pyplot as plt
123 confusion_matrix_graph = confusion_matrix(y_test,
       prediction)
124 # print(confusion_matrix_graph)
125 plt.matshow(confusion_matrix_graph)
126 plt.title('Matriz de confusion para una \nsvm no
       balanceada')
127 plt.colorbar()
128 plt.xlabel('True label')
129 plt.ylabel('Predicted label')
130 plt.show()
131
132
133 print "\n***************Balanceado:**************\n"
134
135 print 'Accuracy:', accuracy_score(y_test,
       weighted_prediction)
136 print 'F1 score:', f1_score(y_test, weighted_prediction)
137 print 'Score:', wclf.score(X_train, y_train)
138 print 'Recall:', recall_score(y_test,
       weighted_prediction)
139 print 'Precision:', precision_score(y_test,
       weighted_prediction)
140 print 'Hamming
       loss:',hamming_loss(y_test,weighted_prediction)
141 print 'Jaccard similarity:'
142 ,jaccard_similarity_score(
143 y_test,weighted_prediction)
144 print 'F-Beta Score:',fbeta_score(y_test,
       weighted_prediction, average='macro', beta=0.5)
```

```
145 print '\n clasification report:\n',
       classification_report(y_test, weighted_prediction)
146 print '\n confussion
       matrix:\n',confusion_matrix(y_test,
       weighted_prediction)
147
148 import matplotlib.pyplot as plt
149 confusion_matrix_graph = confusion_matrix(y_test,
       weighted_prediction)
150 # print(confusion_matrix_graph)
151 plt.matshow(confusion_matrix_graph)
152 plt.title('Matriz de confusion \npara una svm
       balanceada')
153 plt.colorbar()
154 plt.xlabel('True label')
155 plt.ylabel('Predicted label')
156 plt.show()
157
158 print "\n*********Balanceo automatico:*********\n"
159
160 print 'Accuracy:', accuracy_score(y_test,
       auto_weighted_prediction)
161 print 'F1 score:', f1_score(y_test,
       auto_weighted_prediction)
162 print 'score:', auto_wclf.score(X_train, y_train)
163 print 'recall:', recall_score(y_test,
       auto_weighted_prediction)
164 print 'precision:', precision_score(y_test,
       auto_weighted_prediction)
165 print 'hamming
       loss:',hamming_loss(y_test,auto_weighted_prediction)
166 print 'Jaccard
       similarity:',jaccard_similarity_score(y_test,
167 auto_weighted_prediction)
168 print 'F-Beta Score:',fbeta_score(y_test,
       auto_weighted_prediction, average='macro', beta=0.5)
169 print '\nClasification Report:\n',
       classification_report(y_test,
       auto_weighted_prediction)
```

```
print '\nConfussion Matrix:\n',confusion_matrix(y_test,
    auto_weighted_prediction)

import matplotlib.pyplot as plt
confusion_matrix_graph = confusion_matrix(y_test,
    auto_weighted_prediction)
# print(confusion_matrix_graph)
plt.matshow(confusion_matrix_graph)
plt.title('Matriz de confusion para \nuna svm
    auto-balanceada')
plt.colorbar()
plt.xlabel('True label')
plt.ylabel('Predicted label')
plt.show()

import matplotlib.pyplot as plt

confusion_matrix_graph
labels = ['very negative', 'negative', 'neutral',
    'positive', 'very positive']
fig, ax = plt.subplots()
h = ax.matshow(confusion_matrix_graph)
fig.colorbar(h)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
ax.set_xlabel('Predicted')
ax.set_ylabel('Ground truth')
plt.show()
```

Bibliografía

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al*. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

R. Akbani, S. Kwek, and N. Japkowicz. Applying support vector machines to imbalanced datasets. In *Machine Learning: ECML 2004*, pages 39–50. Springer, 2004.

J. R. Anderson, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. *Machine learning: An artificial intelligence approach*, volume 2. Morgan Kaufmann, 1986.

A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35 (8):1798–1828, 2013.

B. C. Bergareche. Negación doble y negación simple en español moderno. *Revista de Filología Románica*, page 63, 1992.

S. Bird, E. Klein, and E. Loper. *Natural language processing with Python*. .o'Reilly Media, Inc.", 2009.

E. Blanco and D. Moldovan. Semantic representation of negation using focus detection. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 581–589. Association for Computational Linguistics, 2011.

C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011a.

C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011b.

Y. Choi and C. Cardie. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 793–801. Association for Computational Linguistics, 2008.

F. L. Cruz, J. A. Troyano, F. Enriquez, and J. Ortega. Clasificación de documentos basada en la opinión: experimentos con un corpus de crıticas de cine en espanol. *Procesamiento de Lenguaje Natural*, 41, 2008.

K. Dave, S. Lawrence, and D. M. Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528. ACM, 2003.

C. Davidson-Pilon. Probabilistic programming and bayesian methods for hackers, 2014.

J. Demšar, B. Zupan, G. Leban, and T. Curk. *Orange: From experimental machine learning to interactive data mining*. Springer, 2004.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9: 1871–1874, 2008.

R. Feldman. Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89, 2013.

J. Friedman. Another approach to polychotomous classifcation. *Dept. Statist., Stanford Univ., Stanford, CA, USA, Tech. Rep*, 1996.

S. N. Galicia-Haro and A. Gelbukh. Extraction of semantic relations from opinion reviews in spanish. In *Human-Inspired Computing and Its Applications*, pages 175–190. Springer, 2014.

S. N. Galicia-Haro, A. Palomino-Garibay, J. Gallegos-Acosta, and A. Gelbukh. Analysis of negation cues for semantic orientation classification of reviews in spanish. In *Advances in Artificial Intelligence and Its Applications*, pages 105–120. Springer, 2015.

L. H. Hamel. *Knowledge discovery with support vector machines*, volume 3. John Wiley & Sons, 2011.

V. Hatzivassiloglou and K. R. McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics*, pages 174–181. Association for Computational Linguistics, 1997.

V. Hatzivassiloglou and J. M. Wiebe. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 299–305. Association for Computational Linguistics, 2000.

C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002.

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

N. Jindal and B. Liu. Identifying comparative sentences in text documents. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 244–251. ACM, 2006.

T. Joachims. *Text categorization with support vector machines: Learning with many relevant features*. Springer, 1998.

D. Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing*, pages 41–50. Springer, 1990.

U. H.-G. Kreßel. Pairwise classification and support vector machines. In *Advances in kernel methods*, pages 255–268. MIT Press, 1999.

V. Labatut and H. Cherifi. Accuracy measures for the comparison of classifiers. *arXiv preprint arXiv:1207.3790*, 2012.

B. Liu. Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2:627–666, 2010.

B. Liu. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012.

V. López, A. Fernández, S. García, V. Palade, and F. Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113–141, 2013.

C. D. Manning, P. Raghavan, H. Schütze, *et al*. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014. URL http://www.aclweb.org/anthology/P/P14/P14-5010.

M.-T. Martín-Valdivia, E. Martínez-Cámara, J.-M. Perea-Ortega, and L. A. Ureña-López. Sentiment polarity detection in spanish reviews combining supervised and unsupervised approaches. *Expert Systems with Applications*, 40(10):3934–3942, 2013.

W. McKinney. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, pages 51–56, 2010.

S. Morinaga, K. Yamanishi, K. Tateishi, and T. Fukushima. Mining product reputations on the web. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 341–349. ACM, 2002.

S. Mund. *Microsoft Azure machine learning*. Packt Publ., 2015.

T. Nasukawa and J. Yi. Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2Nd International Conference on Knowledge Capture*, K-CAP '03, pages 70–77, New York, NY, USA, 2003. ACM. ISBN 1-58113-583-1. doi: 10.1145/945645.945658. URL http://doi.acm.org/10.1145/945645.945658.

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

L. Padró and E. Stanilovsky. Freeling 3.0: Towards wider multilinguality. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*, Istanbul, Turkey, May 2012. ELRA.

A. Palomino-Garibay and S. N. Galicia-Haro. Clasificación automática de la orientación semántica de opiniones mediante características lingüísticas. *Research in Computing Science*, 95:61–74, 2015.

B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

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

M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk. Reducing misclassification costs. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 217–225, 1994.

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

I. D. Rangel, S. S. Guerra, and G. Sidorov. Creación y evaluación de un diccionario marcado con emociones y ponderado para el español. *Onomazein*, 29(1):31–46, 2014.

K. Roebuck. *Sentiment Analysis: High-impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors*. Emereo Publishing, 2012. ISBN 9781743049457. URL https://books.google.com.br/books?id=kqsNBwAAQBAJ.

C. Sammut and G. I. Webb. *Encyclopedia of machine learning*. Springer, 2011.

B. Sanz Alonso. La negación en español. In *Actuales tendencias en la enseñanza del Español como lengua extranjera II: actas del VI Congreso Internacional de ASELE*, pages 379–384. Colegio de España, 1996.

P. Simon. *Too Big to Ignore: The Business Case for Big Data*, volume 72. John Wiley & Sons, 2013.

E. Spitzová *et al*. *Sintaxis de la lengua espaňola*. Masarykova Univerzita, Brno (1994), 1994.

Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307, 2011.

TALP-UPC. INTRODUCCIÓN A LAS ETIQUETAS EAGLES descripción de las etiquetas eagles. `www.cs.upc.edu/~nlp/tools/parole-sp.html`, 2012. Accessed: 2016-04-11.

Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser. Svms modeling for highly imbalanced classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(1):281–288, 2009.

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

S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2): 22–30, 2011.

R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, 2002.

D. Vilares, M. A. Alonso, and C. Gomez-Rodriguez. A syntactic approach for opinion mining on spanish reviews. *Natural Language Engineering*, 21(01):139–163, 2013.

S. Wang and C. D. Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics, 2012.

M. Wiegand, A. Balahur, B. Roth, D. Klakow, and A. Montoyo. A survey on the role of negation in sentiment analysis. In *Proceedings of the workshop on negation and speculation in natural language processing*, pages 60–68. Association for Computational Linguistics, 2010.

Wikipedia. Matriz de confusión — wikipedia, la enciclopedia libre, 2015a. URL https://es.wikipedia.org/w/index.php?title=Matriz_de_confusi%C3%B3n&oldid=86341291. [Internet; descargado 2-febrero-2016].

Wikipedia. Corpus lingüístico — wikipedia, la enciclopedia libre, 2015b. URL https://es.wikipedia.org/w/index.php?title=Corpus_ling%C3%BC%C3%ADstico&oldid=87190075. [Internet; descargado 29-diciembre-2015].

Wikipedia. Natural language processing — wikipedia, the free encyclopedia, 2016a. URL https://en.wikipedia.org/w/index.php?title=Natural_language_processing&oldid=703511297. [En línea; accesado el: 11-February-2016].

Wikipedia. Overfitting — wikipedia, the free encyclopedia, 2016b. URL https://en.wikipedia.org/w/index.php?title=Overfitting&oldid=708039973. [En línea; accesado el: 3-March-2016].

Wikipedia. Support vector machine — wikipedia, the free encyclopedia, 2016c. URL https://en.wikipedia.org/w/index.php?title=Support_vector_machine&oldid=715912651. [En línea; accesado el: 16-April-2016].

S. M. J. Zafra, E. M. Cámara, M. T. M. Valdivia, and M. D. M. González. Tratamiento de la negación en el análisis de opiniones en espanol. *Procesamiento del Lenguaje Natural*, 54:367–44, 2015.