UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

CLASIFICACIÓN DE OPINIONES MEDIANTE APRENDIZAJE DE MÁQUINAS: EL CASO DE RESEÑAS SOBRE PELÍCULAS

TESIS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A:

EDMUNDO PAVEL SORIANO MORALES

DIRECTOR DE TESIS:
DR. ALFONSO MEDINA URREA

Ciudad Universitaria, México D.F. 2011

Resumen

En esta tesis se realiza un sistema capaz de clasificar en positivas o negativas, y a nivel oración, opiniones sobre películas. Esto se logra con ayuda de técnicas del aprendizaje de máquinas y del procesamiento de lenguaje natural. El sistema trabaja con opiniones sobre películas, obtenidas del sitio web IMDb. Este sistema primero separa los enunciados de las opiniones por polaridad, positivos o negativos, dependiendo de la calificación que los usuarios dieron a la película. Segundo, se realiza una selección de rasgos en los enunciados y, con base en ellos, se agrupan para obtener conjuntos de enunciados subjetivos similares. Tercero, se identifican enunciados subjetivos por la presencia en ellos de adjetivos, adverbios o disparadores de presuposición. Finalmente, se entrena, se prueba y se valida un clasificador bayesiano ingenuo (*naïve Bayes*) usando los enunciados obtenidos. Se alcanzó la mayor exactitud usando enunciados subjetivos identificados por la presencia de adjetivos o adverbios. Se concluye analizando las ventajas y desventajas del sistema y el trabajo futuro a realizar.

Érase una vez, ... en que las únicas inteligencias autónomas que conocíamos los humanos eran los humanos. Entonces pensábamos que si la humanidad creaba otra inteligencia sería resultado de un enorme proyecto, una gran masa de silicio y transistores antiguos y chips y placas de circuitos... una máquina con muchos circuitos interconectados imitaría la forma y función del cerebro humano. Por supuesto, las IAs no evolucionaron de ese modo. Se puede decir que se asomaron a la existencia cuando los humanos mirábamos hacia el otro lado.

Dan Simmons, *El Ascenso de Endymion*

Agradecimientos

A mi madre, Sara Micaela, y a mi padre, José Jelvin, por su esfuerzo constante, esfuerzo físico, mental y económico. Por dedicar sus vidas a mis hermanos y a mí. Por dejar pasar sus mejores años y oportunidades por nosotros.

A mi hermana Sara Isabel y a mi hermano Helvin, por acompañarme toda la vida, por ser constantes y estar ahí en todo momento. Por comenzar, antes que yo, este duro cambio hacia la verdadera vida de adulto.

A Alfonso Medina Urrea, mi director de tesis, por su ayuda y paciencia a lo largo de este difícil camino de plasmar en papel lo que se hace en la práctica. Por apoyarme cuando lo necesité, siempre sin dudarlo. Por confiar en mí y en mis capacidades.

A Rodrigo Alarcón por darme un poco de su tiempo para compartir sus ideas y asesorarme desde los primeros esbozos de la tesis hasta las últimas etapas. Por intentar inculcar en mí el sentido del orden y de los *deadlines*, que casi nunca cumplí en tiempo y forma. Al final me arrepentí de no haber usado esas buenas prácticas.

Al Dr. Gerardo Sierra, por apoyarme siempre también. Especialmente durante la aplicación a las becas de maestría y durante la recta final de este trabajo. Me quedo con su "todo se puede en esta vida".

A Teresita Mami por revisar mi tesis, en tiempo récord, y hacer de esta un trabajo más digno. Por ser mi compañera del 12 y por enseñarme las reglas de etiqueta de la alta sociedad.

A todos mis amigos, los que lo fueron y los que lo son, porque nunca me han fallado cuando los he necesitado. Especialmente a Alfredo y al Hobbit.

Al Grupo de Ingeniería Lingüística (GIL) que me brindó la oportunidad de aplicar y expandir mis conocimientos de aprendizaje de máquinas y minería de datos.

A la banda del GIL, en especial a Josué, José Luis y Víctor, por jalar (casi siempre) cuando había que trabajar o cuando había que divertirse o cuando había que trabajar y divertirse. También por hacer bastante entretenido este último año con sus ocurrentes y originales formas de actuar. A Alejandro y a Lázaro por llevar siempre la organización del GIL y, más importante, la organización de los eventos recreativos de fin de semana laboral. A Claudia por su paciencia al enseñarme, con ayuda de Jessi, el maravilloso mundo del baile y sus vueltas. A Azury, que siempre tiene pensamientos positivos a pesar de las adversidades. A Irasema por aguantar mis moditos y ahora brindarme el beneficio de la duda. A Brenda por sus correos *spam* que mantienen informado al GIL de casi todo. A Iria por preocuparse de lo que hago. Al resto de la banda que no menciono, sepan que les agradezco todo lo que han hecho por ayudarme. Estoy seguro de que entrar al GIL fue lo mejor que pude haber hecho en el último año de mi licenciatura. A cada uno de ustedes, gracias por todo.

A mis sinodales: Dr. Miguel Moctezuma, Dr. Alfonso Medina, Dr. Boris Escalante, Dr. Gerardo Sierra y Dr. Juan Manuel Torres, quienes me permitieron concluir este proceso de titulación lo más rápido posible. También al Mtro. Carlos Méndez porque sé que me habría apoyado de la misma forma si hubiera sido mi sinodal.

A la Facultad de Ingeniería, la que me genera una mezcla de tristeza y alegría. Tristeza porque invertí demasiados años en ella. Alegría porque al final, después de varios descalabros tempraneros, entendí que lo mío siempre fue la ingeniería. Comprendí en los últimos años que la

ingeniería mueve al mundo: nos permite ir al Espacio en máquinas increíbles, cruzar ríos tremendos, construir estructuras impensables, nos llevó del telégrafo al Internet y me permite escribir esto en una computadora de dos kilos, guardarlo en una memoria de cinco centímetros de longitud y enviarlo a mi *dropbox* como respaldo.

A los compañeros de la Facultad, conocidos o no, que siempre buscaron aprender más, conocer más y no quedarse sólo con lo que los maestros impartieron. A aquellos que hicieron su verdadero mejor esfuerzo y me obligaron a mejorarme a mí mismo para poder alcanzarlos.

A los profesores, a aquellos que están verdaderamente apasionados por enseñar. Afortunadamente la mayoría.

A mi universidad, la UNAM, por permitir todo esto.

A México y a toda la clase media trabajadora cautiva que pagó por mis estudios.

A todos los couchsurferos que he conocido, provenientes de lugares lejanos: desde Estocolmo hasta Buenos Aires y desde Tampere hasta las Canarias. Casi todos me han dejado un poco de ellos con sus experiencias, anécdotas y sueños. Por demostrarme que querer es poder. Por enseñarme que aunque seamos de lugares tan diferentes y distantes, al final tenemos las mismas cosas adentro.

A Golfi, que con todo y sus golpes, me llevó a mí y a otros cuantos sin fallas ni retrasos.

Muchas gracias.

Aquí les regreso a todos un cachito de lo que me han dado.

Por mi raza hablará el espíritu

Este trabajo se llevó a cabo con el apoyo del CONACyT, en el marco del proyecto *Extracción de conocimiento lexicográfico a partir de textos de Internet* con clave de registro 105711.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Planteamiento del problema

El crecimiento de la Web ha traído consigo la aparición de diversos sitios como foros de opinión y sitios de ventas donde cualquier usuario puede escribir un comentario acerca del producto o servicio que ofrece dicho sitio. Estos comentarios son de gran importancia ya que tendemos a tomar decisiones basadas en lo que los demás piensan acerca de algo o alguien.

La cantidad de opiniones que está disponible a cualquier persona que cuente con acceso a la Web es muy grande. Por ello, resulta imposible analizarlas todas y determinar la tendencia, orientación o polaridad general sobre algo. Ya sea positiva, negativa o neutra, o algo intermedio. El resultado para el usuario es que es imposible realizar una clasificación de forma rápida y/o automática.

El análisis de sentimientos o minería de opiniones es el tratamiento computacional de opiniones, sentimientos y subjetividad en textos. Es el área de la lingüística computacional y de la recuperación de información que hoy en día atiende este problema [1]. Además, la minería de opiniones no busca determinar el tema de un documento, sino la orientación de la opinión que se expresa en él.

La clasificación de opiniones es una tarea de la minería de opiniones que se encarga de asignar a un documento una etiqueta de acuerdo al tipo de opinión que en él se expresa: positiva, negativa o neutral.

La minería de opiniones encuentra sus aplicaciones en diversas áreas. En la recuperación de información, puede ser útil eliminar las opiniones de un documento para obtener mejores resultados a las consultas realizadas [1]. En el área comercial, es muy útil conocer la percepción de un producto o servicio y estar al tanto de lo que la gente opina, como las características negativas o positivas comentadas con mayor frecuencia. Poder contar con un sistema que pueda encontrar y condensar de alguna forma todas estas opiniones para mejorar o replantear un producto o servicio disminuiría el tiempo invertido por un analizador humano que tendría que leer cientos o miles de opiniones, muchas veces iguales.

En el área de inteligencia gubernamental resultaría útil conocer las fuentes de hostilidades, los temas que causan más polémica, entre qué tipo de población surgen opiniones negativas, e inclusive los temas que generan opiniones subversivas. Asimismo, sería útil conocer los temas que causan malestar entre la población para poder atenderlos y así mejorar la vida de los habitantes.

1.2. Objetivos

El objetivo general de este trabajo de tesis es crear un sistema capaz de clasificar, en positivas o negativas, oraciones[1] provenientes de reseñas en inglés sobre películas, según la opinión que contengan. Este clasificador será supervisado, entrenado con un corpus de reseñas de películas recabado desde la Web. Al entrenar el clasificador, se proponen y se ponen a prueba cuatro métodos que tienen por objetivo mejorar el desempeño del clasificador.

Los objetivos específicos de este trabajo son:

- Extraer de la Web reseñas u opiniones acerca de películas. Preprocesar esa información y dejarla lista para ser usada por las siguientes etapas del sistema.

[1]En este trabajo, se usa oración y enunciado como sinónimos; aunque se entiende la diferencia en el significado de estas dos palabras: mientras que una oración es una estructura que contiene necesariamente un verbo, un enunciado puede ser cualquier cosa que ocurra entre dos puntos. Como en este trabajo no se ocupa de estructuras oracionales, al hablar de oraciones me estaré refiriendo a enunciados.

- Agrupar automáticamente los enunciados, con el fin de separar los enunciados con algún tipo de opinión de los enunciados descriptivos, usando selección de rasgos[2].
- Elegir las oraciones que cuentan con adjetivos o adverbios con el fin de entrenar el sistema, ya que típicamente estos se utilizan para hacer juicios, tanto negativos como positivos [2, 3] .
- Elegir las oraciones que cuentan con disparadores de presuposición, ya que las oraciones que contienen estos disparadores presuponen otro tipo de información que podría ofrecer un juicio.
- Elegir las oraciones que cuentan con adjetivos o adverbios o con disparadores de presuposición. La selección de este tipo de enunciados aumentará la cantidad de oraciones elegidas, manteniendo solo las más relevantes. Este método es la unión de los dos métodos anteriores.
- Crear un clasificador de tipo Bayes ingenuo (*naïve Bayes*) que reciba las oraciones agrupadas o las oraciones subjetivas (encontradas con alguna de las técnicas mencionadas) y que cuenten ya con una clase asignada (separadas en dos conjuntos, uno de oraciones negativas y el segundo de oraciones positivas). Probar el clasificador con ejemplos nuevos. Como último paso, evaluar por medio de medidas de desempeño, el comportamiento del clasificador.

1.3. Estado del arte

Hoy en día, en el área de minería de opiniones, se trabaja principalmente en resolver los dos principales problemas de esta área, la identificación del texto subjetivo y la clasificación de la opinión contenida en ese tipo de textos. Sin embargo, existen otras aplicaciones, de acuerdo a [4, 5, 6], que representan también nuevos problemas y retos. Estas aplicaciones son tres principalmente:

[2]La selección de rasgos es un método que utiliza sólo los rasgos que ofrezcan la mayor información de acuerdo a cierto criterio.

1. **Comparación de productos**: para poder ofrecer una comparación acerca de un producto con otro producto, es necesario conocer qué se opina sobre las características que definen a ese producto. Asimismo, si se conocen las opiniones sobre las características del producto, el usuario podría leer solo las opiniones que conciernen a las características en las cuales él está interesado. En [7] se realizó un sistema capaz de comparar las características de diferentes productos que compiten entre sí. Primero, se usaron técnicas de minería de patrones de lenguaje para identificar las características sobre las cuales los consumidores han expresado alguna opinión. Segundo, por cada característica identificada, se averiguó si la opinión de cada usuario es positiva o negativa. Con esta información realizaron una interfaz que permite visualizar y comparar las opiniones de diferentes productos.

2. **Resumen de opiniones**: el número de opiniones generadas en línea crece rápidamente, y más para productos populares. Para un usuario es complicado leer todas las opiniones, más aún cuando las oraciones juiciosas están contenidas en un texto largo, donde la mayoría de las oraciones no ofrecen opinión alguna. En [8] se realizaron resúmenes de opiniones mediante los siguientes tres pasos:

   a) Se identifican las opiniones de las características del producto,

   b) Se determina la polaridad de esas opiniones, y

   c) Se genera un resumen con la información obtenida.

   Con un resumen, los potenciales usuarios pueden observar fácilmente como otros usuarios se sienten respecto al producto.

3. **Minería de motivaciones de la opinión**: Además de conocer la orientación de la opinión de un producto, es de gran utilidad conocer las razones por las cuales el autor de la opinión se expresó positiva o negativamente. En [9] se detectaron expresiones con opiniones mediante la búsqueda de oraciones que explícitamente indican las ventajas y desventajas de los productos. Posteriormente se entrenó un sistema de reconocimiento de oraciones, para

obtener los enunciados con las características que generan la opinión final de los usuarios.

Además de estos nuevos retos, se ha desarrollado, siguiendo la clasificación positiva o negativa de opiniones, la clasificación de emociones. Las emociones humanas son múltiples y depende de la investigación qué conjunto de ellas usar. En [10], se utiliza un modelo de emociones distinguibles ya verificado empíricamente y se sugiere su uso potencial en el procesamiento del lenguaje natural para la clasificación automática de emociones escritas en textos en inglés en ocho niveles. En [11] se utilizan seis emociones básicas: ira, desagrado, miedo, alegría, tristeza y sorpresa. En este trabajo se construyó un corpus formado con encabezados de noticias. Posteriormente se etiquetó manualmente con las seis emociones mencionadas. Finalmente se probaron distintas técnicas para detectar automáticamente las emociones en los encabezados.

Otra área de investigación desarrollada, muy ligada a la minería de opiniones, es la detección de ironía[3]. En [13] se enfocaron en detectar la ironía en enunciados (escritos en portugués) que contienen predicados positivos, dado que presumen que estos son los más expuestos a la ironía. Esto lo lograron explorando ciertos indicios orales y gestuales en los comentarios de usuarios de un sitio web de noticias. Mediante características lingüísticas, [12] identificó tres grupos diferentes de ironía. También, en ese trabajo, se plantea la posibilidad de identificar automáticamente la ironía de dos de los grupos identificados.

En lo que concierne a la identificación de frases subjetivas y a la clasificación de las opiniones contenidas en esas frases, al ser problemas de clasificación, se pueden resolver por medio de aprendizaje supervisado.

Para la identificación de oraciones subjetivas, en [14], se seleccionaron como rasgos de entrenamiento elementos influenciadores de opinión contextuales, tales como la negación (no, nunca) y la contradicción (pero, sin embargo). En [15, 16] se usaron elementos léxicos, para clasificar oraciones en subjetivas u objetivas. Otro enfoque usado es la aproximación por similitud; este método se basa en la hipótesis de que las frases subjetivas se parecen más a otras frases subjetivas que

[3]La deinición de ironía usada es la encontrada en [12]: la ironía como palabras que expresan lo contrario de lo que se quiere decir.

a frases sin opinión alguna. En [17] se mide la similitud entre oraciones mediante rasgos como palabras compartidas, frases y *synsets*[4] de *Wordnet*.[5]

Para la clasificación de opiniones, en [18] se utilizó el análisis de conjunciones entre adjetivos para detectar la orientación de las frases subjetivas. Al analizar pares de adjetivos (unidos por *y, o, pero,* etc.) extraídos de un conjunto de documentos, la intuición es que el hecho de unir adjetivos está sujeto a limitaciones lingüísticas que definen la orientación de los adjetivos unidos. (por ejemplo, *y* usualmente une dos adjetivos de la misma orientación, mientras que *pero* une regularmente dos adjetivos de orientación opuesta). En [19] se sigue la hipótesis de que dos palabras tienden a ser de la misma orientación semántica si existe entre ellas una fuerte asociación semántica. Haciendo uso de las relaciones léxicas encontradas en *Wordnet*, se pudo calcular una cierta distancia entre adjetivos y definir la orientación de cada uno de ellos.

1.4. Descripción de la tesis

Este trabajo de tesis esta formado por cinco capítulos. En el segundo capítulo se presentan los conceptos relacionados a la clasificación de opiniones, o minería de opiniones, como la minería de datos, de donde se desprende la minería de textos. Asimismo, la minería de textos, como la de datos, aplica técnicas y algoritmos que provienen del aprendizaje de máquinas. Estos temas se abordan en ese capítulo. Se concluye con la minería de opiniones, tema principal de esta tesis, donde se conjuntan todos los conceptos revisados.

El tercer capítulo aborda las herramientas usadas y la metodología seguida para la creación del sistema de clasificación de opiniones. Se presentan las etapas que dieron forma al sistema y también ejemplos de cómo la información recabada fue procesada y almacenada.

[4]Un conjunto de sinónimos o synset es un grupo de datos que son considerados equivalentes semánticamente.

[5]Wordnet es una base de datos léxica para el inglés. Agrupa palabras en synsets, provee definiciones cortas y contiene las diferentes relaciones semánticas entre estos synsets. http://wordnetweb.princeton.edu/perl/webwn

En el cuarto capítulo se presentan y analizan los resultados obtenidos por el clasificador de opiniones. Se muestran tablas de resultados relevantes y diagramas que ayudan a visualizar rápidamente el desempeño de los diferentes métodos utilizados.

En el quinto capítulo se encuentran las conclusiones a las que se llegaron con base en los objetivos planteados y los resultados obtenidos. Se presentan las ventajas y desventajas del sistema y se abordan las posibles mejoras a realizar en el futuro.

En el apéndice A se presentan las cinco tablas con las matrices de confusión para cada uno de los métodos usados para entrenar el clasificador bayesiano. También se presenta la tabla con los resultados promediados de precisión, exhaustividad y medida F.

En el apéndice B se describen los módulos programados que componen al sistema creado en esta tesis. También se presenta un diagrama que muestra la relación que existe entre cada uno de estos módulos.

Finalmente, en los apéndices C y D se describen los módulos de agrupamiento y clasificación, respectivamente. Se indican los parámetros de entrada, los objetos que regresa y los archivos que guarda en disco cada uno de estos módulos.

Capítulo 2

Conceptos de la tesis

2.1. Minería de datos

La minería de datos es la tecnología de la cual se desprende la minería de textos, por lo que es importante definirla y comprenderla.

Hoy, en el mundo, se generan inmensas cantidades de información diariamente gracias, en gran parte, a la Web, a las computadoras personales omnipresentes y a los aparatos electrónicos que permiten guardar nuestros documentos, nuestras fotografías, nuestras decisiones, nuestros hábitos de consumo, entre otros tipos de información digital. La Web nos permite acceder a toda esta información, al mismo tiempo que todos los quehaceres personales (comercio, encuestas, juegos, sitios sociales, etc.) son almacenados. La brecha que existe entre la generación de información y la comprensión de esa información crece vertiginosamente y conforme aumenta el volumen de datos, la cantidad de personas que lo entienden disminuye de forma alarmante [20].

La minería de datos se define como el proceso de descubrir patrones en grandes cantidades de datos. Este descubrimiento debe ser de forma automática o semi-automática y los patrones encontrados deben ser de alguna utilidad, ya sea de utilidad económica, de utilidad científica, que demuestren la existencia de fenómenos no encontrados o no estudiados con anterioridad, que sirvan para realizar sugerencias de acuerdo a los datos analizados o incluso para identificar posibles

amenazas sociales[1], entre otros tipos de utilidad.

Un sistema de minería de datos tiene generalmente una entrada y una salida. La entrada es un conjunto de ejemplos del cual se pretende generalizar nuevos ejemplos. La salida es una descripción que clasifica ejemplos desconocidos. Por ejemplo, si se cuenta con ejemplos de transacciones bancarias fraudulentas, sería de gran interés clasificar las nuevas transacciones en dos categorías, transacciones legítimas y transacciones fraudulentas.

Los métodos de minería de datos procesan información numérica estructurada, obteniendo medidas de cada ejemplo del conjunto de entrada y entregando una predicción, basada en los ejemplos de entrada, acerca de algún ejemplo nuevo y desconocido.

2.2. Minería de textos

Los textos son considerados como información sin estructura, por lo que se podría pensar que los métodos de minería de datos no se aplican a ellos [21]. Sin embargo los textos pueden convertirse a valores medidos: ya sea la presencia de palabras, la frecuencia con la que aparecen o alguna otra métrica existente. Si se tienen estos valores entonces se pueden aplicar los mismos métodos de minería de datos a los textos, aunque estos deben ser implementados con consideraciones, por ejemplo, a la alta dimensionalidad, ya que los textos contienen miles de palabras que los definen y existen miles de documentos. La alta dimensionalidad representa un reto importante, ya que el desempeño de los algoritmos usados en la minería de datos depende generalmente del número de rasgos que definen a un objeto, por lo que es necesario realizar optimizaciones en estos algoritmos con el fin de reducir el uso de los recursos computacionales (tales como el tiempo de procesamiento y el espacio en memoria).

Los beneficios de la minería de textos han resultado en innovaciones tecnológicas que ayudan a la gente a entender mejor y a usar la información disponible en repositorios de documentos [22]. Estas tecnologías, como detección de contenidos,

[1]Como el sistema ADVISE desarrollado por el Departamento de Seguridad Nacional de los Estados Unidos: http://en.wikipedia.org/wiki/ADVISE

rastreo y obtención de tendencias, son usadas hoy en día en una gran cantidad de ámbitos, ya sea en bancos y en finanzas, en la industria, en comercios, entre otros.

Como su nombre lo indica, la minería de textos hace uso de documentos y el conjunto de textos estructurados es llamado corpus. Generalmente estos conjuntos contienen grandes cantidades de textos, por lo que los algoritmos que los procesen deben ser escalables, independientes del lenguaje y confiables [22].

Computacionalmente, los métodos para analizar los corpus se dividen en tres categorías principales: los basados en métodos estadísticos, los basados en métodos simbólicos y los híbridos.

- Métodos Estadísticos: son aquellos que no toman en cuenta la información semántica ni las propiedades lingüísticas de un texto. Cada documento de un corpus es representado por un vector que contiene la frecuencia, o alguna otra métrica estadística, de cada palabra que aparece en el documento. Este modelo es llamado bolsa de palabras (*bag-of-words*). Luego los vectores (que representan a cada documento del corpus) unidos, forman una matriz que representa al modelo de espacio vectorial. Esta representación, a pesar de no contar con información semántica, entrega resultados extremadamente buenos para una variedad de aplicaciones [22].

- Métodos Simbólicos: los métodos simbólicos (a veces mal llamados "lingüísticos"), generalmente basados en técnicas de procesamiento de lenguaje natural (PLN o NLP por sus siglas en inglés, *Natural Language Processing*), utilizan modelos de lenguaje para extraer y representar relaciones y significados expresados en el mismo. Pueden obtener representaciones profundas acerca de la estructura del texto. Sin embargo, estos modelos resultan difíciles de construir, mantener y depurar debido a la gran cantidad de reglas y presuposiciones que los componen.

- Métodos híbridos: combinan técnicas de los dos descritos anteriormente.

A lo largo de este trabajo de tesis se usan métodos estadísticos y simbólicos (acercamiento híbrido), que analizan el texto de forma numérica pero conservando algún tipo de información semántica, como se verá más adelante.

2.2.1. Etapas de un sistema de minería de textos

En general, los sistemas de minería de textos cuentan con cuatro etapas, las cuales son las mostradas en la figura 2.1.

Figura 2.1: Etapas de un sistema de minería de textos. - La figura muestra las cuatro etapas más comunes en un sistema de minería de textos.

2.2.1.1. Etapa I: Recopilación de documentos

El primer paso para analizar texto es recolectar la información, en este caso los documentos relevantes. Es importante notar que en la minería de textos un documento puede ser un texto de miles de líneas o de solo una línea. En muchos casos esta información ya está disponible y solo basta con asegurarse de su calidad para poder analizarlos.

En otros casos, se necesita una colección de documentos; es decir, un corpus. Para extraer documentos, por ejemplo, de la Web, se ocupan herramientas (por ej., un Web crawler) cuyo propósito es recolectar documentos. Un Web crawler es una aplicación de software que recorre la Web automáticamente descargando las páginas de interés de acuerdo a su objetivo. En la minería de textos es usado para formar grandes colecciones de texto desde la Web, que no están disponibles de ninguna otra forma.

2.2.1.2. Etapa II: Tareas de preprocesamiento

Las técnicas usadas en esta etapa se desprenden del PLN. De acuerdo a [23], el procesamiento del lenguaje natural es el área de la ciencias de la computación que busca que las máquinas o computadoras lleven a cabo tareas útiles que impliquen el uso del lenguaje humano. Tareas tales como permitir la comunicación humano-máquina, mejorar la comunicación humano-humano, o realizar procesamiento de

texto o de voz que resulte útil de alguna forma. Para lograr estas tareas, el PLN requiere varios tipos de conocimiento acerca del lenguaje:

- Fonética y fonología: conocimiento acerca de sonidos lingüísticos.
- Morfología: conocimiento de los componentes significativos de las palabras.
- Sintaxis: conocimiento de las relaciones estructurales entre palabras.
- Semántica: conocimiento del significado.
- Pragmática: conocimiento de la relación del significado con los objetivos e intenciones del hablante.
- Discurso: conocimiento sobre unidades lingüísticas más largas que una simple declaración.

El PLN, al intentar extraer un representación más completa del significado del texto, ayuda a la minería de textos a descubrir conocimiento interesante y útil de texto no estructurado.

Las tareas de preprocesamiento descritas a continuación están basadas en técnicas del PLN.

2.2.1.2.1. Estandarización de los documentos

Una vez obtenidos los documentos que conformarán el corpus, se deben almacenar de manera uniforme que permita manipularlos, leerlos y escribirlos fácilmente.

Para este fin se ha adoptado, en la comunidad de procesamiento de texto, el lenguaje XML (*Extensible Markup Language*) [21]. Este formato estándar permite insertar etiquetas dentro del texto para identificar sus partes. Estas etiquetas forzosamente deben existir en pares de inicio y de finalización. Dentro de cada etiqueta pueden existir más etiquetas, permitiendo especificar aun más cada parte. Los nombres de las etiquetas son arbitrarios, sin embargo existen ya ciertos patrones que se siguen para estandarizar las colecciones de texto. Para trabajar como usuario con este formato, hoy en día existen muchos editores de texto y procesadores de palabras que permiten leer y guardar archivos en este formato.

Como desarrollador de aplicaciones, la mayoría de los lenguajes de programación modernos pueden, por medio de librerías y de interfaces de programación de aplicaciones (APIs), procesar información XML.

El objetivo de estandarizar el texto en un formato como XML es poder utilizar las herramientas de minería de textos con cualquier documento sin importar como fue generado ni su formato original [21].

2.2.1.2.2. Segmentación

Cuando ya se tienen los documentos estandarizados, el siguiente paso es encontrar rasgos que caractericen al texto almacenado. Por lo que es necesario definir fronteras y separar el texto en partes más simples.

Dos tipos de segmentaciones resultan relevantes para esta tesis: la segmentación de enunciados y la segmentación de palabras gráficas o *tokens*.

La segmentación de enunciados es un paso crucial en el procesamiento del texto [24]. Pretende dividir el texto en las oraciones individuales que lo componen. Antes de segmentar en palabras o tokenizar, es necesario segmentar el texto en enunciados. En inglés (la lengua usada en los experimentos de este trabajo), el punto, los signos de exclamación y de interrogación son delimitadores razonables. Sin embargo el punto es ambiguo, ya que existe también en las abreviaturas que lo utilizan y que pueden o no indicar la finalización de un enunciado.

Generalmente los algoritmos de segmentación de oraciones trabajan construyendo un clasificador binario (basado en secuencias de reglas o aprendizaje de máquinas) el cual decide si un signo de puntuación es parte de una palabra o es un delimitador de un enunciado [24].

La segmentación de palabras (tokenización) puede resultar complicada en lenguajes que no tengan una representación visual de las fronteras entre cada palabra [25]. En inglés y en otras lenguas, el espacio en blanco es considerado una frontera para delimitar palabras. También lo son los signos de puntuación, tales como los caracteres ()¡¿!?.^{en}tre otros [21]. Sin embargo, hay situaciones en las que los signos de puntuación están dentro de las palabras, como en Ph.D, AT&T, reddit.com y 444,444 (cuando se consideren los números como tokens). Para segmentar palabras se pueden definir distintas reglas que abarquen cada caso, dependiendo de

qué caracter precede qué caracter o si el caracter es letra mayúscula o no. Sin embargo, estas reglas se pueden volver complicadas de entender y de mantener, por lo que una opción práctica es usar expresiones regulares para definir las fronteras entre palabras y separar el texto por esas fronteras designadas.

Cuando se han identificado los tokens, se pueden encontrar los tipos. Un tipo es un conjunto de tokens; esto es, un token es una instancia de un tipo, por lo que suele haber un número de tokens por cada tipo. Si se tiene la oración "La gata es de la señora", en esta oración existen dos tokens "la", y estos dos tokens son una instancia del tipo "la". O sea un tipo que tiene dos tokens.

2.2.1.2.3. Lematización

El siguiente paso es convertir los tokens a una forma estándar. Esto puede ser útil dependiendo de la aplicación del sistema. Lematizar, en este trabajo de tesis, implica llevar un token a su raíz, removiendo las flexiones de las palabras. Con la raíz, se puede generar una conflación, que significa tratar como sinónimos varias palabras con la misma raíz. Existen diversos métodos para obtener las raíces de los tokens. El usado en este trabajo es el algoritmo Porter, descrito en [23].

Un ejemplo de lematización es el siguiente: para los tokens en inglés *exciting*, *excited* y *excitation*, la raíz encontrada por el algoritmo Porter es *excit*. Esta raíz *excit* aglomera las tres palabras anteriores en una sola palabra. Cuando aparezca en el texto alguna de esas tres palabras, se tratarán como sinónimos y se representarán por la raíz encontrada.

2.2.1.2.4. Generación del vector de rasgos

Para caracterizar un texto es necesario definir un conjunto de rasgos. Estos rasgos pueden ser simplemente las palabras que aparecen en cada texto o las palabras más frecuentes de un corpus. Una vez identificados estos rasgos se procede a formar una tabla o matriz basada en el concepto del modelo de espacio vectorial, es decir, representar cada texto en el corpus por medio de un vector donde cada dimensión de este es el valor del rasgo elegido. Posteriormente, el conjunto de vectores forma la matriz que contiene los rasgos en las columnas, los textos en los renglones y cada celda toma el valor que le corresponde a ese rasgo en ese texto. En la minería de textos, esta matriz tiene la característica de ser de alta

dimensionalidad, debido a que, como ya se explicó, un conjunto de textos puede significar una gran cantidad de palabras. Además, la mayoría de las entradas en la matriz serán cero, dado que los textos generalmente no comparten las mismas palabras. Por lo anterior, un tratamiento especial se aplica a este tipo de matrices llamadas dispersas. Se deben utilizar estructuras de datos y algoritmos diseñados explícitamente para almacenar este tipo de matrices y realizar operaciones con estas.

Para reducir el tamaño del vector se puede utilizar una lista de palabras de paro (*stoplist*), para filtrar las palabras funcionales que la mayoría del tiempo no tienen capacidades predictivas de interés en la minería [21], tales como "el", "la", "a", entre otras. Estas palabras se pueden remover del vector de rasgos para hacer más pequeño dicho vector. También se pueden utilizar técnicas de selección de rasgos que intentan elegir un subconjunto de palabras que puedan tener un mayor potencial para la predicción; aunque generalmente no se utilizan y se confía en la frecuencia de las palabras para colocarlas o no en el vector de rasgos. Lematizar las palabras también ocasiona que el vector se reduzca. Si se almacena solo la raíz de cada palabra podemos aglomerar en esa única raíz todas las variantes presentes en el texto.

2.2.1.2.5. N-gramas

En esta tesis, los n-gramas se definen como secuencias de n palabras consecutivas. Estas secuencias pueden ser de una sola palabra, cuando n es igual a uno, llamados unigramas, de dos palabras, bigramas, de tres palabras, trigramas y así sucesivamente. Estos n-gramas se pueden utilizar como rasgos. De hecho, cuando se usan palabras únicas se podría decir que se usan unigramas. También se pueden utilizar bigramas o trigramas para formar el vector de rasgos.

2.2.1.2.6. Etiquetado PoS (*Part of Speech tagging*)

Como se dijo antes, el objetivo del PLN es analizar y entender el lenguaje. Al estar aún lejos de alcanzar esa meta, el PLN se ha enfocado en tareas intermedias que buscan encontrar sentido de alguna parte de la estructura inherente del lenguaje sin requerir un entendimiento completo de él. El etiquetado PoS (*Part of Speech Tagging*) es una de estas tareas [25].

Etiquetar es asignar a cada palabra de una oración una categoría gramatical basada en el papel que cumple dentro de la oración. Las etiquetas proveen información acerca del contenido semántico de la palabra [26].

La mayoría de las gramáticas en inglés deberían tener como mínimo el sustantivo, el verbo, el adjetivo, el adverbio, la preposición y la conjunción.

El conjunto de etiquetas *Penn Treebank* [27], construido del corpus del periódico *Wall Street Journal*, contiene 36 categorías. Este conjunto de etiquetas es el utilizado en este trabajo.

De acuerdo a [23], la entrada para un algoritmo de etiquetado es una cadena de palabras y un conjunto de etiquetas. La salida es la mejor etiqueta encontrada para cada palabra, tal como se aprecia en la figura 2.2.

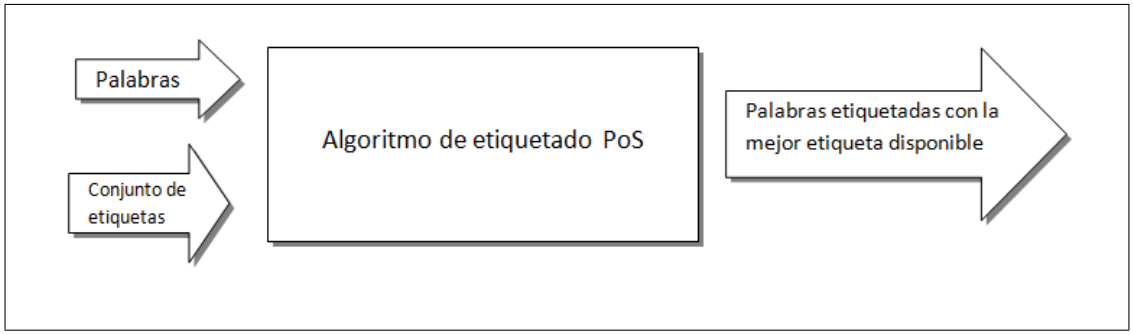

Figura 2.2: Diagrama de un etiquetador PoS - La figura muestra, a grandes rasgos, las entradas y la salida de un etiquetador PoS.

Un ejemplo en inglés de un etiquetado POS es el siguiente:
Oración original:
"Deliver me from Swedish furniture"
Oración etiquetada:
Deliver/VB (Verbo)
me/PRP (Pronombre personal)
from/IN (Preposición)
Swedish/JJ (Adjetivo)
furniture/NN (Sustantivo)

Después de aplicar las tareas de procesamiento del texto, se tiene un corpus estandarizado y etiquetado al cual se le aplicará la siguiente etapa.

2.2.1.3. Etapa III: Operaciones principales de minería

En esta etapa se aplican los algoritmos de minería de datos a los textos preparados para obtener la información deseada por el sistema. Aquí se encontrarían los métodos necesarios para clasificar texto, resumirlo, recuperar información, entre otras tareas.

2.2.1.4. Etapa IV: Presentación

Incluye la Interfaz Gráfica de Usuario (*GUI*) del sistema, herramientas de visualización para los resultados obtenidos, editores de consultas, entre otros instrumentos que faciliten al usuario final la interpretación y la manipulación de la información entregada por el sistema.

2.3. Aprendizaje de Máquinas (*Machine Learning*)

El aprendizaje de máquinas es un área de la inteligencia artificial cuyo objetivo es desarrollar algoritmos y técnicas que permitan a las computadoras resolver problemas mediante datos de ejemplo o experiencias obtenidas con anterioridad. Tiene una amplia gama de aplicaciones tales como procesamiento del lenguaje natural, motores de búsqueda, diagnósticos médicos, bioinformática, análisis de mercados y reconocimiento de patrones, entre otras.

El aprendizaje de máquinas se aplica cuando tenemos un problema y no se cuenta con un algoritmo que lo pueda solucionar. Por ejemplo, para separar correos electrónicos legítimos de correos basura, tenemos una entrada (un correo electrónico) y sabemos cuál debe ser la salida: correo basura o correo legítimo. En este contexto, la pregunta es cómo transformar la entrada en la salida [28]. La idea general del aprendizaje de máquinas es compensar la falta de conocimiento (el algoritmo) con la información disponible. Fácilmente se pueden juntar miles de correos electrónicos de los cuales conocemos su clase, legítimos o basura, y a partir de ellos hacer que la computadora aprenda que es lo que hace al correo

basura diferente del correo legítimo, esto con base en los ejemplos (experiencia) obtenidos.

El aprendizaje de máquinas utiliza la estadística para construir modelos matemáticos[2]. Estos modelos están definidos con ciertos parámetros, y el aprendizaje consiste en programar y ejecutar un programa de computadora que optimice los parámetros de dicho modelo, usando datos de entrenamiento (experiencia pasada). El modelo generalmente es predictivo, es decir, puede hacer predicciones en el futuro (clasificaciones)[28].

2.3.1. Clasificación[3]

Una de las aplicaciones más importantes del aprendizaje de máquinas es la clasificación, que es cuando tenemos dos o más clases a las que debemos asignar un caso no visto antes. La experiencia anterior nos ayuda a entrenar un sistema que encuentre automáticamente la salida (la etiqueta que designa la clase) dada una entrada (un nuevo caso). En cuanto a la clasificación se refiere, la experiencia anterior consiste en un dominio o conjunto de objetos, donde cada uno de ellos pertenece a una clase conocida.

Por ejemplo, en el reconocimiento de rostros, la entrada es la imagen de un rostro y las clases son las personas a ser reconocidas. Los datos de entrenamiento pueden ser miles de imágenes de rostros que son usados para entrenar el sistema. La salida es la asociación de cada rostro con una identidad [28].

Otro ejemplo es la clasificación de textos, donde los documentos pueden ser organizados por los temas que tratan.

Para realizar la clasificación de opiniones, en este trabajo se ha optado por usar el algoritmo de Bayes ingenuo, el cual se explica a continuación.

2.3.1.1. Método de Bayes ingenuo (*naïve Bayes*)

El método de Bayes ingenuo es importante dentro de los métodos usados para la clasificación por diversas razones, entre ellas está que es fácil de construir, fácil

[2]Un modelo matemático es la descripción de un sistema utilizando lenguaje matemático.

[3]También conocido como aprendizaje supervisado. Supervisado porque requiere de datos de entrenamiento etiquetados

de interpretar y a pesar de que asume la independencia condicional de las variables utilizadas (de ahí su nombre de ingenuo), se desempeña sorprendentemente bien. Podrá no ser el clasificador más robusto pero sus resultados suelen ser bastante confiables [29].

El proceso de aprendizaje bayesiano es muy eficiente. Analiza los datos de entrenamiento solo una vez para estimar todas las probabilidades requeridas para la clasificación [30].

A continuación se explicará el algoritmo orientado a la clasificación de textos.

Según [31], la clasificación con Bayes Ingenuo es vista como la estimación de la probabilidad *a posteriori* de una clase c dado un texto d:

$$P\left(c|d\right) \propto P\left(c\right) \prod_{1 \leq k \leq n_d} P\left(t_k|c\right)$$

Donde $P\left(t_k|c\right)$ es la probabilidad condicional del término t_k ocurriendo en un documento de la clase c[4]. $P\left(t_k|c\right)$ es una medida de qué tanto la evidencia t_k contribuye a que c sea la clase correcta. $P\left(c\right)$ es la probabilidad *a priori* de que un documento pertenezca a la clase c. Por otra parte, $(t_1, t_2, t_3, \ldots, t_{nd})$ son los tokens o palabras en d de donde se infiere el vocabulario usado (los tipos) para la clasificación y n_d es el número de tokens en d.

La clase elegida es la mejor clase posible, que es la que tiene una probabilidad mayor a las demás clases. Por lo que para predecir la clase c de un documento d es necesario calcular:

$$c = \arg\max_{c \in \mathbb{C}} P(c|d) = \arg\max_{c \in \mathbb{C}} P(c) \prod_{1 \leq k \leq n_d} P(t_k|c) \tag{2.1}$$

donde $\mathbb{C}$ es el conjunto de las clases posibles: $c_1, c_2, \ldots, c_{|\mathbb{C}|}$.

[4]En [31] se explica porque $P(c|d)$ es proporcional ($\propto$) y no igual al elemento de la derecha.

Se observa que cada parámetro condicional $P(t_k|c)$ es un peso que indica qué tan bueno es t_k como un indicador para la clase c. Así como $P(c_i)$ es un indicador de la frecuencia relativa con la que aparecen las clases. Las clases que cuentan con una mayor proporción en C son las que tienen mayor posibilidad de ser las correctas. Entonces la multiplicación de los indicadores es una medida de qué tanto pertenece un documento a una clase.

Una vez conocida la ecuación, es necesario estimar los parámetros para entrenar el clasificador.

$P(c_i)$ se estima simplemente dividiendo el número de documentos que pertenecen a esa clase, N_c, entre el número de documentos de entrenamiento disponibles, N.

$$P(c) = \frac{N_c}{N} \tag{2.2}$$

La probabilidad condicional $P(t_k|c)$ se calcula como la frecuencia relativa del término k en los documentos que pertenecen a la clase c:

$$P(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}} \tag{2.3}$$

Donde T_{ct} es el número de ocurrencias del término t en los documentos de entrenamiento de la clase c. V es el conjunto de palabras distintas o vocabulario en los documentos de entrenamiento para la clase c. El denominador es simplemente el número de palabras que ocurren en los datos de entrenamiento para esa clase. Existen dos problemas que hay que resolver al aplicar Bayes ingenuo:

1. En la ecuación 2.1, muchas probabilidades condicionales son multiplicadas, una por cada posición $1 \leq k \leq n_d$. Estas probabilidades son valores menores a la unidad. Por lo que al multiplicarse sucesivamente, los valores tienden a

cero, posiblemente causando un *floating point underflow*[5]. Por lo tanto, se utiliza en lugar de la multiplicación, la suma de logaritmos. La clase con el logaritmo de la probabilidad más grande será todavía la clase más probable.

Dado que:

$$\log(xy) = \log(x) + \log(y)$$

Entonces la predicción de la clase se calcula, con suma de logaritmos, de la siguiente forma:

$$c = \arg\max_{c \in \mathbb{C}} [\log P(c) + \sum_{1 \leq k \leq n_d} P(t_k|c)]$$

2. Si se desea clasificar un documento que contiene alguna palabra que no apareció durante el entrenamiento, se obtendría una probabilidad de cero para esa palabra en esa clase, lo cual provocaría que la probabilidad se hiciera cero o se volviera indeterminada al sumar logaritmos naturales de cero. La solución consiste en suavizar las probabilidades. La forma estándar de hacerlo es aumentando la cuenta de cada palabra distinta con una pequeña cantidad $\lambda\,(0 \leq \lambda <= 1)$, de tal forma que cada palabra tendrá al menos una muy pequeña probabilidad de ocurrencia. Esto es llamado suavizado de Lidstone. Cuando $\lambda = 1$, el suavizado es conocido como suavizado de Laplace [30]. Por lo tanto, la ecuación para obtener la probabilidad condicional de un término dada una clase queda de la siguiente forma:

$$P(t|c) = \frac{T_{ct} + \lambda}{\sum_{t' \in V} T_{ct'} + \lambda |V|}$$

[5]El agotamiento de punto flotante o *floating point underflow* es una condición en un programa de computadora que ocurre cuando el verdadero resultado de una operación es menor en magnitud al mínimo valor representable como punto flotante normal en el tipo de dato usado. El resultado es la transformación automática del valor a cero, alterando completamente el resultado de la operación.

donde $|V|$ es el número de términos en el vocabulario.

Cuando se entrena el sistema se almacenan los valores de probabilidad condicional para cada palabra encontrada en los documentos de entrenamiento. Cuando se clasifica se busca la probabilidad de cada palabra del documento nuevo. Si no se encuentra la palabra, se le asigna el valor determinado por el suavizado.

Dado que solo se analiza una vez la información durante el entrenamiento, el algoritmo es lineal al número de ejemplos de entrenamiento, haciéndolo extremadamente eficiente, siendo esta una de sus grandes fortalezas [30].

2.3.2. Evaluación del clasificador

Después de construir el clasificador es necesario conocer la calidad de los resultados entregados por el sistema. Este paso consiste en evaluar el desempeño de la solución propuesta al problema planteado. ¿Es el sistema mejor que predecir aleatoriamente las clases? ¿Alcanza un desempeño que haga que su futura aplicación valga la pena? [32].

Para hacer esta evaluación es necesario realizar pruebas sobre la muestra de datos con la que se cuenta. Esta muestra contiene ejemplos con sus clases ya conocidas *a priori*. Como se dijo antes, el objetivo del clasificador es generalizar, a partir de esta información, nuevos ejemplos que aparecerán en el futuro.

Ahora, cuando se entrena y se prueba con el mismo conjunto de casos, seguramente se obtendrían resultados muy buenos. Sin embargo, estos resultados no se generalizarían para casos nuevos [32]. A esto se le conoce como *overfitting* y se busca evitar por medio de métodos que separan la muestra total en dos conjuntos separados: conjunto de entrenamiento y conjunto de pruebas. De hecho, la técnica más sencilla usada para la evaluación consiste en separar la muestra en dos, una parte para entrenar y otra parte para probar.

2.3.2.1. Remuestreo (*resampling*)

En lugar de simplemente separar los casos en dos muestras (entrenamiento y pruebas), se podrían elegir aleatoriamente varios de los casos para entrenar y el resto para realizar las pruebas; después, repetir este proceso varias veces. Esto es

el remuestreo y pretende reducir los errores en las estimaciones cuando se pruebe el clasificador con nuevos casos [32].

2.3.2.1.1. Validación cruzada con k pliegues (k - *fold cross validation*)

Este método utiliza toda la información disponible tanto para entrenar como para probar.

Se divide aleatoriamente el conjunto total de muestras en k partes del mismo tamaño. Se generan varios pares de dos elementos: un conjunto de entrenamiento y uno de pruebas. Para generar cada par se deja una de las k partes fuera como conjunto de pruebas y se combinan las $k-1$ partes restantes para el conjunto de entrenamiento. Haciendo esto k veces, cada vez dejando fuera otra de las k partes, se obtienen k pares de conjunto de entrenamiento y conjunto de pruebas[6].

Por ejemplo, si se desea realizar una validación con $k = 3$, con una muestra de 999 casos de la clase A y 999 casos de la clase B, se divide el conjunto total en tres partes: parte 1, parte 2 y parte 3, cada una con 333 casos de la clase A y 333 casos de la clase B. Luego se entrenan tres clasificadores:

- El primero se entrena con las partes 1 y 2 y se prueba con la parte 3.
- El segundo se entrena con las partes 1 y 3 y se prueba con la parte 2.
- El tercero se entrena con las partes 2 y 3 y se prueba con la parte 1.

En cada ocasión se usan 666 casos para entrenar y 333 casos para probar.

Típicamente k es igual a 10 ó 30 [28]. Conforme k crece, se obtienen estimaciones más robustas, sin embargo el tamaño del conjunto de prueba disminuye para una misma muestra, si bien aumenta el conjunto de entrenamiento. Esto es, el costo en tiempo y recursos para el entrenamiento depende también de k: Se debe entrenar un clasificador k veces, y el tamaño del conjunto de entrenamiento crece también con k.

[6]Un caso especial de la validación cruzada es la validación cruzada deja uno afuera (*take one out*). En este método k es igual al número de ejemplos disponibles (no se tiene la opción de cambiar k, sino que se asume que el número de casos es la unidad primordial). Es usado normalmente cuando el número de casos es pequeño, pero obviamente no es eficiente para un conjunto de gran tamaño, ya que se deben entrenar k clasificadores [30].

2.3.3. Medición del desempeño del clasificador

Para un clasificador binario, se genera una matriz de dimensión 2 x 2, la cual contiene todos los posibles resultados de la clasificación.

	Clasificado como positivo	Clasificado como negativo	Total
Realmente positivo	tp	fn	p
Realmente negativo	fp	tn	n
Total	p'	n'	N

Tabla 2.1: Matriz con los únicos posibles resultados de una clasificación binaria.

Esta matriz es llamada matriz de confusión (ver tabla 2.1) y contiene cuatro valores que representan el desempeño del clasificador que se explican en la siguiente sección.

2.3.3.1. Medidas de desempeño

Tomando como referencia los valores de la tabla 2.1, para un caso realmente positivo, si la predicción es también positiva, se le denomina positivo verdadero, **tp** (*true positive*). Si la predicción es negativa, para el mismo caso positivo, se le llama falso negativo, **fn** (*false negative*). Si el caso es realmente negativo y se predice como positivo, es un falso positivo, **fp** (*false positive*). Si con el mismo caso negativo, se predice negativo, se obtiene un negativo verdadero, **tn** (*true negative*).

Además, **p** es el número de casos positivos reales, **n** es el número de casos negativos reales, **p'** es el número de casos clasificados como positivos y **n'** es el número de casos clasificados como negativos.

En la diagonal principal de la matriz se encuentran las clasificaciones correctas (tp y tn), mientras que los otros dos valores (fn y fp) representan los dos tipos de errores.

Por ejemplo, si se tiene una aplicación cuyo fin es autenticar usuarios de un estacionamiento por su huella digital, el **error falso negativo** representaría la ocasión en que un usuario válido quisiera entrar al estacionamiento y fuera

Nombre	Fórmula
error	$(fp + fn)/N$
exactitud	$(tp + tn)/N = 1 - error$
razón-tp	tp/p
razón-fp	fp/n
precisión	tp/p'
exhaustividad	$tp/p =$ razón-tp
sensitividad	$tp/p =$ razón-tp
especificidad	$tn/n = 1-$ razón-fp

Tabla 2.2: Medidas de desempeño para clasificadores

rechazado por la aplicación. El **error falso positivo** sería cuando a un usuario no autorizado se le da acceso erróneamente. Se puede ver que los dos tipos de errores no son igual de graves, el falso positivo es más grave en una aplicación de este tipo. Aunque, dependiendo de la aplicación, estos dos errores pueden variar su grado de importancia.

Las medidas más comunes para medir el desempeño de los clasificadores binarios son las mostradas en la tabla 2.2. Nótese que la razón-tp sirve para estimar tanto la exhaustividad como la sensitividad.

La **exactitud** (*accuracy*) es utilizada frecuentemente para determinar el desempeño de un clasificador. Sin embargo, no es muy útil cuando se está interesado sólo en la clase minoritaria (la clase menos representada en la muestra), ya que, retomando el ejemplo del estacionamiento, asumiendo que el 90 % de los usuarios son auténticos, un clasificador sólo tiene que predecir cada caso como positivo (aceptado) para obtener una exactitud del 90 %, aunque haya existido un solo caso, el más relevante, que fue clasificado como válido cuando en realidad era un usuario no autorizado, es decir, un caso negativo clasificado como positivo. El **error** es el inverso de la exactitud. La mayoría del tiempo nuestro interés no se enfoca en medidas generales como estas dos, sino en los tipos de errores antes mencionados.

En cuanto a la precisión (*precision*) y la exhaustividad (*recall*), estas son también medidas adecuadas para determinar qué tan completa y precisa fue la predicción en la clase positiva. La **precisión** es el número de ejemplos correctamente clasificados como positivos dividido entre el número total de ejemplos

clasificados como positivos. La **exhaustividad** es el número de ejemplos correctamente clasificados como positivos dividido entre el número total de ejemplos positivos reales en el conjunto de pruebas [30].

En aplicaciones de recuperación de información la medida F (F-measure), es comúnmente utilizada como medida única de desempeño. La **medida F** es la media armónica de la precisión y la exhaustividad, y está definida por:

$$Medida\ F = \frac{2}{\frac{1}{precisión} + \frac{1}{exhaustividad}}$$

La exactitud es por mucho la métrica de desempeño más utilizada. Sin embargo, se ha demostrado que las razones para usar la exactitud como medida única son muy cuestionables, siendo el análisis ROC (acrónimo de *Receiver Operating Characteristic*, esto es, característica operativa del receptor), que se describe abajo, una alternativa no tan simple de obtener como la exactitud pero que permite realizar conclusiones firmes y generales [33].

2.3.3.1.1. Espacio ROC

Una gráfica ROC es una técnica para visualizar, organizar y seleccionar clasificadores basados en su desempeño. El espacio ROC se originó en la teoría de detección de señales y tiene por $eje - x$ la razón-fp y por $eje - y$ la razón-tp. El análisis ROC se ha extendido recientemente al aprendizaje de máquinas y ahora es usado con mayor frecuencia en la evaluación y comparación de algoritmos.

Para trazar un punto o una curva en el espacio ROC, se necesita definir el tipo de clasificador que se tiene. Existen generalmente dos tipos de clasificadores por la forma en la que designan la clase predicha:

1. Clasificadores discretos: Determinan la clase de un caso de forma binaria, *sí o no* se pertenece a la clase. Esos clasificadores entregan un solo punto, el cual es trazado en el espacio ROC.

2. Clasificadores probabilísticos: Entregan un valor numérico que representa el grado de pertenencia de un caso a una clase. Si se determina un umbral, y la salida del clasificador es mayor a ese umbral, entonces ese caso se considera

positivo, de lo contrario, el caso es clasificado como negativo. Si se varía ese umbral, se obtendrá un punto para cada valor diferente. Uniendo estos puntos se obtendría una curva ROC.

Para ubicar un punto en el espacio ROC basta con conocer los valores de razón-tp y de razón-fp del clasificador a representar. Estos dos valores se usan como coordenadas en el plano XY y con ellas se traza el punto.

Para trazar la curva se requiere de un procedimiento más complejo. Como se dijo antes, es necesario tener un clasificador probabilístico y contar con un umbral, de tal forma que si este se supera, el clasificador asigna la clase positiva, de lo contrario asinga la clase negativa. Al variar este umbral se obtienen puntos que se grafican en el espacio ROC. Toda curva ROC generada de un conjunto finito de casos producirá una función escalón que se aproxima a una verdadera curva al mismo tiempo que el número de casos tiende a infinito.

En el espacio ROC, entre más arriba y a la izquierda se encuentre un punto o la cresta de la curva, mejor es el desempeño del clasificador. Idealmente un clasificador debería estar en la coordenada $(0, 1)$ donde se tiene la máxima razón-tp y la mínima razón-fp. La línea diagonal $y = x$ corresponde a usar un clasificador aleatorio, por lo que es poco común observar un punto o curva por debajo de esta línea, ya que si se negara el resultado obtenido por ese clasificador, se obtendría automáticamente un mejor desempeño. Un clasificador que se encuentra en la diagonal no ofrece información acerca de la pertenencia a las clases [34]. En la figura 2.3 se ilustra el espacio ROC.

Las curvas y puntos en el espacio ROC representan herramientas útiles para visualizar y evaluar clasificadores. Como se dijo, son capaces de ofrecer información más detallada acerca del desempeño de un clasificador en comparación con medidas como la exactitud o el error (ya que estas medidas dependen directamente de la distribución de las clases). La curva ROC sirve también para conocer el *trade-off*[7] de un clasificador y poderlo ajustar dependiendo de lo que se deseé clasificar con él.

[7]El trade-off se refiere a perder un tipo de calidad, pero ganando otro tipo de calidad. Es decir, se elige entre clasificar mejor los casos realmente positivos a costa de clasificar con mayor frecuencia casos realmente negativos como positivos.

En este trabajo, el espacio ROC (aunado a las medidas convencionales de error y exactitud) es utilizado para determinar el método que ofrece el mejor desempeño.

2.3.4. Agrupamiento[8] (*Clustering*)

Cuando no se conocen las clases en las que queremos separar los datos, no se cuenta con datos de entrenamiento y/o solo se tiene la información de entrada, se aplican las técnicas de agrupamiento para organizar los datos automáticamente.

Todos los problemas de agrupamiento son problemas de optimización. El objetivo es elegir la mejor agrupación de objetos posible de acuerdo a cierta función de calidad.

Un buen resultado de un agrupamiento debería reunir objetos similares y separar los distintos. Por lo tanto la función de calidad se expresa en términos de una función de similitud entre objetos. Una función de similitud toma dos objetos y produce un valor real, el cual indica la proximidad que existe entre esos objetos. Este valor se obtiene con base en los rasgos que representan a cada objeto.

Como se dijo antes, el concepto de representar un objeto como vectores de rasgos multidimensionales es llamado modelo de espacio vectorial. En este modelo, la función de similitud está usualmente basada en la distancia entre vectores de acuerdo a alguna métrica [26].

La función de similitud más popular es la distancia Euclidiana:

$$D(x_i, x_j) = \sqrt{\sum_k (x_{ik} - x_{jk})^2}$$

donde x_i y x_j son dos objetos y k es la dimensión de su vector de rasgos.

La distancia Euclidiana es un caso particular de la métrica de Minkowski, cuando $p = 2$:

$$D_p(x_i, x_j) = \left(\sum_k (x_{ik} - x_{jk})^p \right)^{\frac{1}{p}}$$

[8]También conocido como aprendizaje no supervisado. No requiere o no se cuentan con datos de entrenamiento etiquetados.

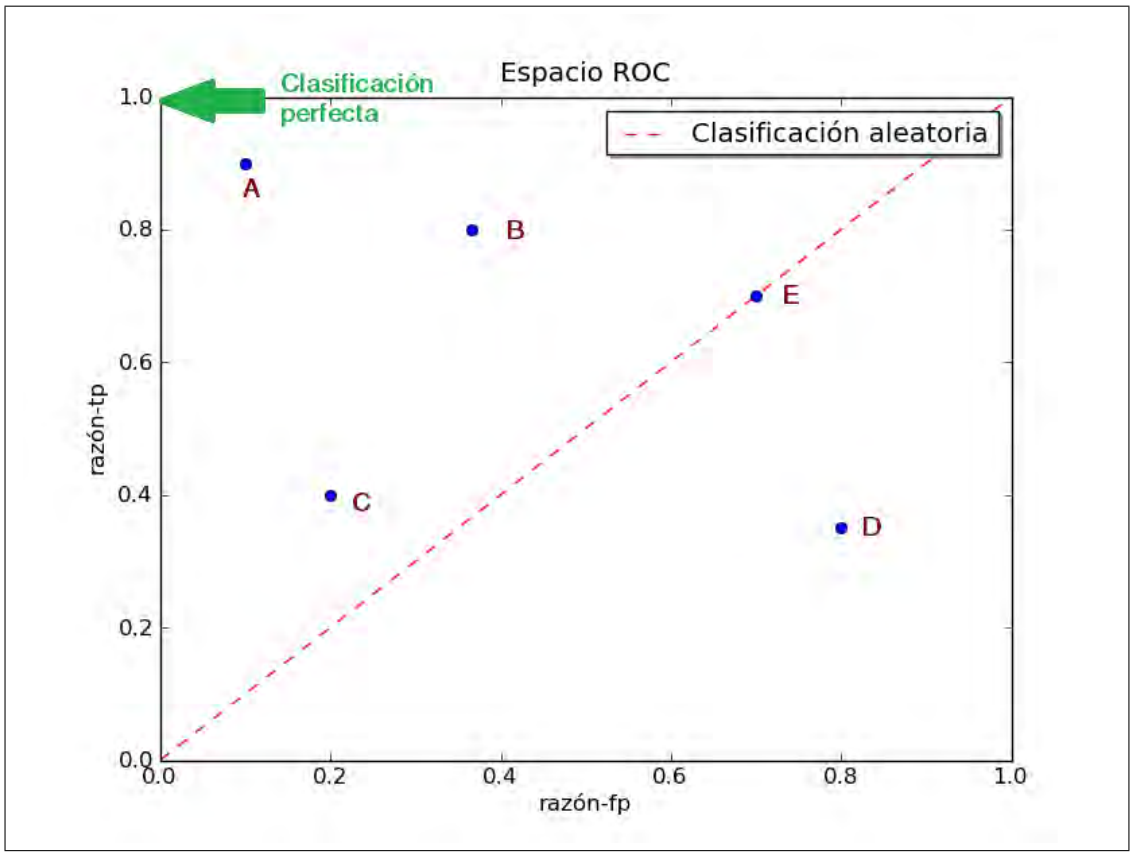

Figura 2.3: Espacio ROC - La figura muestra varios puntos en el espacio ROC. Entre más arriba y a la izquierda se encuentre el punto, el desempeño del clasificador es mejor. La línea diagonal $y = x$ representa una predicción aleatoria. El punto (0.0,1.0) representa el resultado de una clasificación perfecta: sin errores falsos positivos y todos verdaderos positivos. En este ejemplo, el punto A representa al mejor clasificador, el punto E representa a un clasificador con desempeño igual al de un clasificador aleatorio. El punto C representa a un clasificador con un desempeño muy pobre, muy cercano al desempeño de un clasificador aleatorio. Finalmente, el punto B representa el mismo clasificador que el que representa el punto D pero negado.

Existe un gran número de medidas de similitud disponibles, cada una sirve a un propósito particular [26].

2.3.4.1. Tipos de algoritmos de agrupamiento

Los algoritmos de agrupamiento se pueden clasificar en algunos tipos básicos. Por la estructura que producen, existen dos tipos: algoritmos jerárquicos y algoritmos particionales o planos.

Un **algoritmo jerárquico**, como su nombre lo indica, entrega como resultado una estructura de datos de tipo árbol, jerárquica, donde cada nodo representa una subclase de su nodo padre. Las hojas del árbol son los objetos individuales del conjunto agrupado. Cada nodo representa el grupo que contiene a todos los objetos de sus descendientes. Un ejemplo de los resultados de este tipo de algoritmo se observa en la figura 2.4

Un **algoritmo particional** entrega un cierto número de grupos. La relación entre los grupos es pocas veces determinada. La mayoría de estos algoritmos son iterativos: inician con un conjunto inicial de grupos y los reubican en cada iteración, mejorando su distribución. En la figura 2.5 se muestra un ejemplo de este tipo de agrupamiento.

Otra diferencia importante de los algoritmos de clasificación depende de la membresía de cada objeto. La membresía de un objeto indica a qué grupo pertenece ese objeto. Si cada objeto es asignado a uno y solo un grupo, entonces es **agrupamiento duro**. Por otro lado, el **agrupamiento suave** permite varios grados de membresía y permite tener membresía en múltiples grupos.

Existen ciertas técnicas de agrupamiento estándar, tales como árboles de decisión, bosques de árboles de decisión, máquinas de vectores de soporte, el algoritmo de k - medias, el algoritmo de esperanza - maximización, entre otras [36]. Estas técnicas se desempeñan bien con conjuntos de datos comunes, sin embargo, para conjuntos de datos complejos, son usadas técnicas de análisis más complejas.

La **descomposición de matrices**, descrita a continuación, puede ofrecer un análisis más completo, o puede también producir datos más limpios para después ser usados por los métodos estándar.

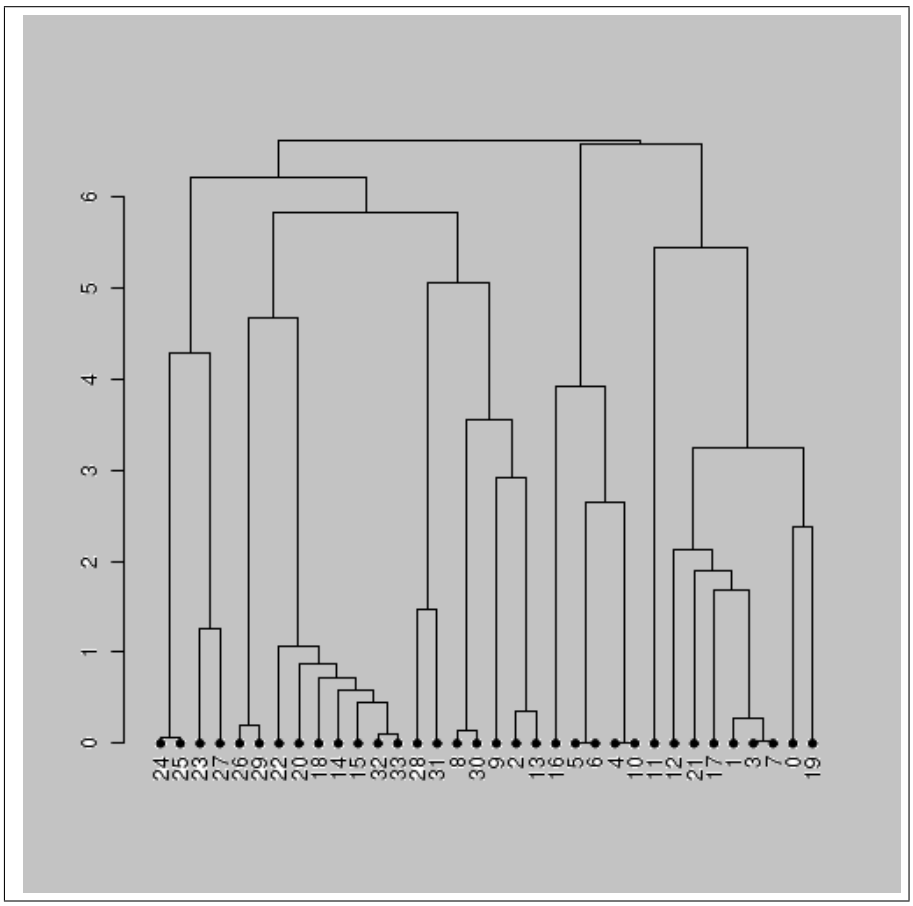

Figura 2.4: Ejemplo de un agrupamiento jerárquico - La figura muestra los resultados de la aplicación de un agrupamiento jerárquico. Este tipo de diagramas es llamado dendrograma y representa el ordenamiento de los grupos producidos por este método de agrupamiento. Esta imagen fue tomada de [35].

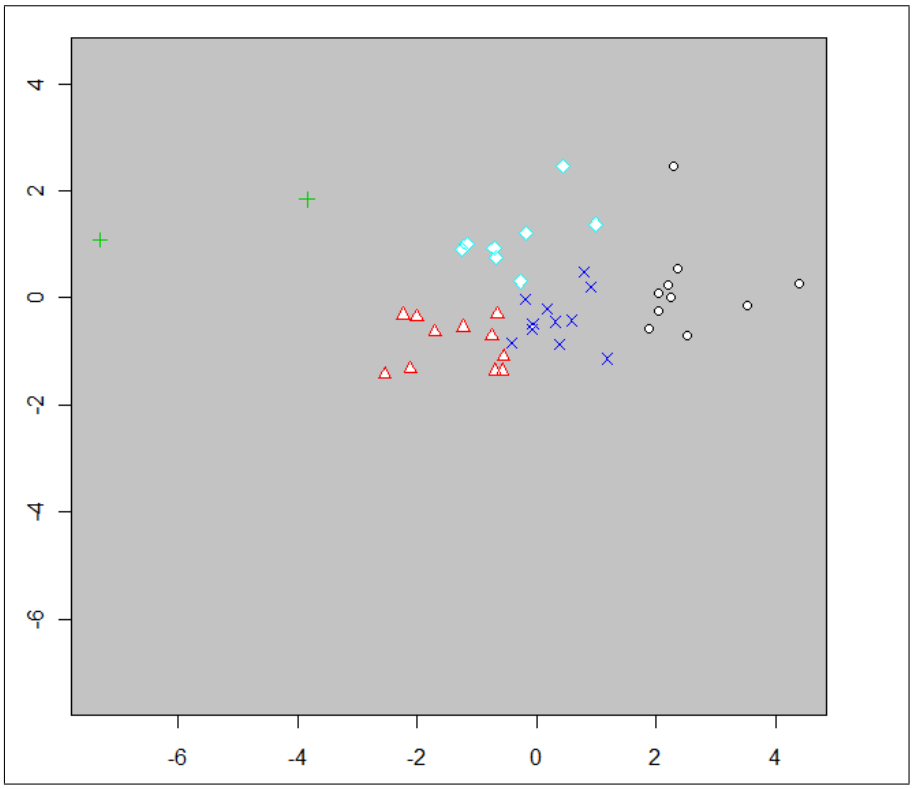

Figura 2.5: Ejemplo de un agrupamiento particional - La figura muestra los resultados de la aplicación de un agrupamiento particional. Cada objeto es miembro de un grupo encontrado.

2.3.4.2. Descomposición de matrices

De acuerdo a [36], la descomposición de matrices es usada principalmente para dos tareas en el análisis de datos:

- Es capaz de separar los datos basura, producto de procesos no controlados o de errores, de los datos disponibles. Esto es útil al aplicar técnicas convencionales ya que de esta forma producirán resultados mejores. Esta tarea podría llamarse limpieza de datos.
- Agrupa objetos de un conjunto de datos, ya sea usando alguna técnica estándar o interpretando el resultado de la descomposición.

Los algoritmos más comunes de descomposición de matrices son:

- Descomposición en Valores Singulares (*Singular Value Decomposition SVD*) y Análisis de Componentes Principales (*Principal Component Analysis PCA*);
- Descomposición semidiscreta (*SemiDiscrete Decomposition SDD*);
- Análisis de Componentes Independientes (*Independent Component Analysis ICA*);
- Factorización no negativa de matrices (*Non-Negative Matrix Factorizarion NMF*).

Descripción

Si consideramos un conjunto de datos como una matriz, con n renglones, cada uno representando a un objeto y m columnas, cada una representando un rasgo, entonces la celda ij representa el valor del rasgo j para el objeto i. En los algoritmos de descomposición de matrices, se busca expresar una matriz de datos, A, como el producto de un conjunto de nuevas matrices que sacan a la luz las estructuras y relaciones implícitas en A.

Formalmente, una descomposición de matrices puede ser descrita por una ecuación de la forma:

$$A = WCH \tag{2.4}$$

donde las dimensiones de las matrices son:

- La dimensión de A es $n \times m$ con $n \gg m$ (esto es, n es mucho mayor que m)
- W es $m \times r$ para una r que es usualmente menor a m
- C es $r \times r$
- H es $r \times m$

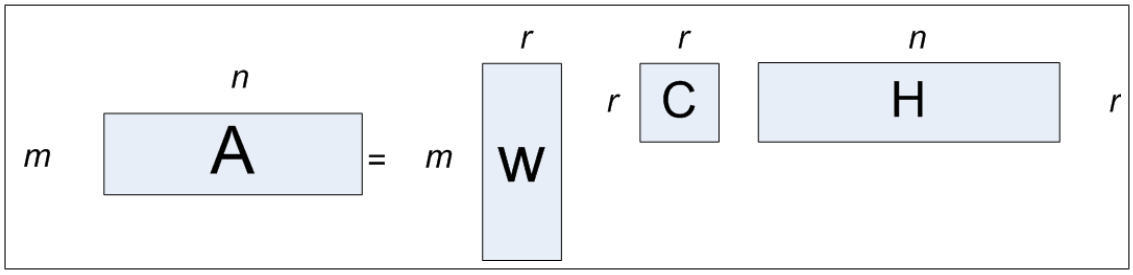

Figura 2.6: Descomposición de la matriz de datos A en las matrices W, C y H - La figura muestra las matrices y sus dimensiones después de una descomposición de matrices.

Específicamente, de la ecuación 2.4 se observa que un elemento de A, por ejemplo el elemento a_{11} es generado por la multiplicación del primer renglón de W, el primer elemento de C, y la primera columna de H. Por lo que cada valor en A es una combinación de partes de F, combinados de ciertas formas descritas por W y C.

La matriz W tiene el mismo número de renglones que A. El $i-ésimo$ renglón de W proporciona r piezas de información que juntas dan una nueva interpretación al $i-ésimo$ objeto; mientras que A provee m piezas de información acerca del $i-ésimo$ objeto.

La matriz H tiene siempre el mismo número de columnas que A. Cada columna de H da una nueva interpretación del atributo descrito por la columna correspondiente de A, en términos de r piezas de información.

El papel de r es el de forzar una representación más compacta con respecto de la forma original. Se asume que una matriz más pequeña (que represente a A), capturará las regularidades latentes que puedan existir dentro de A. Generalmente el valor de r es más pequeño que el de m [36].

La matriz C contiene valores que reflejan las relaciones entre los factores latentes: el $ij-ésimo$ valor (c_{ij}) ofrece la conexión que existe entre el factor latente capturado por la $i-ésima$ columna de W y el factor latente capturado por el $j-ésimo$ renglón de H. Algunas descomposiciones no generan esta matriz intermedia, tal es el caso de la factorización no negativa.

La mayoría de los métodos para descomponer matrices pueden ser expresados como problemas de optimización limitada.

En esta tesis se ocupó el algoritmo de factorización no negativa de matrices, el cual se explica a continuación.

2.3.4.2.1. Factorización no negativa de matrices

La factorización no negativa de matrices (FNM) representa una alternativa al análisis de componentes principales. Este método es particional duro y está diseñado para conjuntos de datos cuyos valores de sus atributos nunca son negativos. Asimismo, los elementos de las matrices resultantes no contienen valores negativos tampoco. Por ejemplo, los documentos con texto no pueden contener frecuencias negativas de palabras así como las imágenes no pueden tener cantidades negativas de colores.

La factorización no negativa de matrices produce solo dos matrices, como se observa en la figura 2.7, la matriz W y la matriz H. Por lo que la definición de la factorización no negativa es:

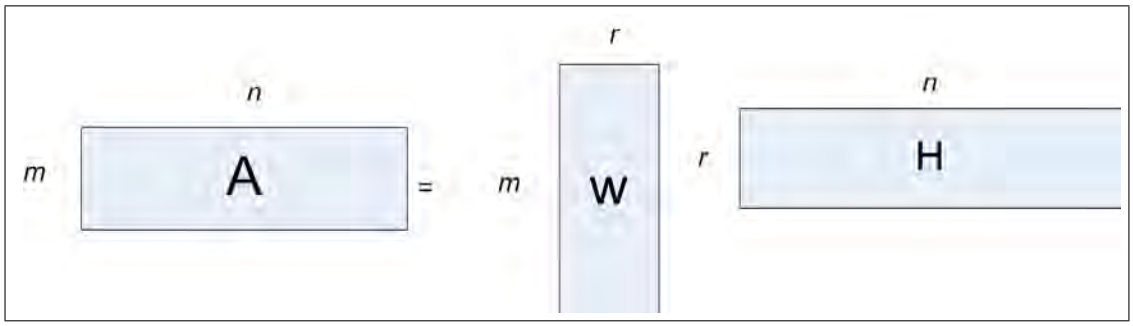

Figura 2.7: Matrices obtenidas después de la factorizacion con FNM. - La figura muestra los resultados obtenidos al aplicar la factorización no negativa de matrices.

$$A \approx WH$$

donde:

- A es $m \times n$
- W es $m \times r$
- H es $r \times n$
- $r \leq m$

Ambas matrices, W y H, deben contener solo valores no negativos. W es la matriz de factores y H es la matriz de mezcla.

El enfoque convencional para encontrar W y H es minimizar la distancia entre A y el producto WH:

$$\min_{W,H} f(W,H) \equiv \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} (A_{ij} - (WH)_{ij})^2$$

$$\text{sujeto a } W_{ia} \geq 0, H_{bj} \geq 0, \forall i, a, b, j.$$

Esta descomposición no es única, depende del método para inicializar W y H, del algoritmo con el que se generen finalmente las mismas matrices y de la métrica de error usada para comprobar la convergencia. Algunos de los algoritmos de optimización usados para FNM son:

- Actualización multiplicativa, descrita por [37].
- Codificación dispersa, descrita por [38].
- Descenso de gradientes con mínimos cuadrados limitados, descrito por [39].
- Mínimos cuadrados alternantes, descrito en [40].
- Mínimos cuadrados alternantes usando gradientes proyectados, descrito por [41].

Los algoritmos usados durante el desarrollo de este trabajo, se explican a continuación.

Actualización multiplicativa (*Multiplicative Method MM*)

Usando la norma de Frobenius[9], la función objetivo (o problema de minimización) se define como:

$$\min_{W,H} ||V - WH||_F^2$$

con W y H siempre no negativas.

El algoritmo, usando la norma de Frobenius, se describe como:

1. Inicializar W y H con valores aleatorios no negativos.
2. Iterar por cada c, j, i (índices de cada una de las matrices) hasta que el error de convergencia sea mínimo o después de n iteraciones:

   a) $H_{cj} \leftarrow H_{cj} \frac{(W^T V)_{cj}}{(W^T W H)_{cj} + \epsilon}$

   b) $W_{ic} \leftarrow W_{ic} \frac{(V H^T)_{ic}}{(W H H^T)_{ic} + \epsilon}$

   c) $W \leftarrow |W|, H \leftarrow |H|$

Epsilon (ϵ) se agrega al denominador para evitar divisiones entre cero, su valor es muy pequeño (10^{-9}).

Mínimos cuadrados alternantes usando gradientes proyectados (*Alternating Least Squares using Projected Gradients*)

Además del método de actualización multiplicativa, en esta tesis (como se explicará más adelante, en el capítulo tres de metodología) también se utilizó el algoritmo de mínimos cuadrados alternantes usando gradientes proyectados, el cual difiere de MM en la forma de darle solución al problema de optimización (esto es, lo realiza con mínimos cuadrados alternantes). Se ha demostrado que este método converge más rápido (requiriendo menos iteraciones) [41].

[9]También llamada norma de Hilbert - Schmidt, se define como: $\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$

El problema de mínimos cuadrados alternantes se expresa con las siguientes reglas, en sustitución de las reglas del paso 2 del algoritmo descrito arriba:

$$W^{k+1} = \arg\min_{W \geq 0} f(W, H^k),$$

$$H^{k+1} = \arg\min_{H \geq 0} f(W^{k+1}, H)$$

Por la necesidad de brevedad, en esta tesis no se ahondará en la explicación de la aplicación de gradientes proyectados para resolver estas reglas. Véase [41] para una explicación más detallada. De hecho, la implementación en Python de este método utilizada en esta tesis se debe a A. Di Franco, cuyo código puede consultarse en [42].

Obtenidas las matrices W y H, estas se pueden utilizar para realizar el agrupamiento: el vector a_j es asignado al grupo i si h_{ij} es el elemento más grande en la columna j de H.

2.4. Minería de Opiniones

2.4.1. Introducción

Conocer las opiniones de otras personas sobre algún tema en específico siempre ha sido importante para formar un juicio propio. El área de minería de opiniones se encarga del tratamiento computacional de la opinión, sentimiento y subjetividad en el texto. A continuación se esbozan algunas características de este campo de investigación (ver [22]).

La información contenida en textos se puede dividir, a grandes rasgos, en dos categorías: descripciones y opiniones. Las descripciones son expresiones objetivas acerca de cosas, eventos, etc., y de sus propiedades. Las opiniones son usualmente expresiones subjetivas que describen los sentimientos, aprecio o juicio acerca de un objeto, un evento, etc., y de sus propiedades [4].

Dado que el concepto de opinión, y por ende de sentimientos, es muy amplio, en este trabajo se trabajará solamente con opiniones negativas y positivas. Entendiendo como opinión negativa a aquella que habla mal acerca de algo y como opinión positiva a aquella que habla bien acerca de algo.

En las últimas dos décadas, gracias al desarrollo de la Web, se han generado cientos de sitios que permiten a millones de usuarios expresar sus opiniones acerca de casi cualquier tema. Estas opiniones cada vez se vuelven más importantes a la hora de tomar decisiones, ya sea al realizar la compra de un producto o servicio, o en la elección de candidatos políticos. Al existir una cantidad tan grande de opiniones, es necesario construir sistemas de acceso a la información con la finalidad de ayudar a los consumidores a tomar decisiones, ya que la información puede presentarse de forma tramposa, confusa, excesiva o difícil de localizar.

Si se conocen las tendencias de opinión acerca de un producto, es posible, para la empresa que lo produce, ajustar sus campañas publicitarias, el posicionamiento de la marca, y otras posibles estrategias de promoción.

Para realizar un sistema de minería de opiniones se deben resolver los siguientes problemas:

1. Determinar si el usuario desea o no una reseña u opinión acerca de algo. Esto se puede resolver al usar palabras como “opinión” o “reseña” al momento

de realizar una petición y realizando una clasificación dentro de la consulta hecha.

2. Encontrar, dentro de un documento que hable sobre el objeto deseado, las oraciones que contengan material de opinión. Si se sabe de antemano que la información es una opinión (dado que el sitio de donde se extrajo la información está dedicado a ofrecer opiniones), entonces es un problema relativamente fácil de resolver. En cambio, cuando se extrae información de sitios como blogs o páginas personales, se debe identificar la porción de la reseña que contiene algún tipo de opinión.

3. Una vez encontrada la opinión, se debe determinar el parecer o sentido que en ella se expresa, ya sea una opinión negativa o positiva y en ciertos casos una opinión neutral[10].

4. Finalmente, el sistema debe expresar la información encontrada de manera resumida y clara, ya sea con un valor numérico representando la opinión general, o con las frases más representativas de cada tipo de opiniones (positiva o negativa).

El desarrollo de esta tesis se enfoca a resolver los problemas dos y tres.

La investigación sobre minería de opiniones ha crecido rápidamente en la última década gracias a diversos factores:

- El mayor uso de métodos de aprendizaje de máquinas dentro del área de PLN y de recuperación de información.

- La disponibilidad de información, gracias al crecimiento de la Web específicamente de sitios donde se pueden verter opiniones.

- Los retos intelectuales y las aplicaciones comerciales que el área ofrece.

[10]En este trabajo, una opinión neutral se refiere a la clase de enunciados objetivos, es decir, aquellos que no contienen una opinión o juicio

2.4.2. Aplicaciones

Aplicación en sitios orientados a reseñas

Esta aplicación consiste en obtener información acerca de algún tema desde un sitio de la Web enfocado a recolectar reseñas y opiniones de los usuarios. Estas reseñas pueden ser después resumidas automáticamente e incluso se pueden corregir las calificaciones cuando están equivocadas (por ejemplo, cuando un usuario ha dado una calificación baja cuando su reseña es positiva).

Aplicación como una etapa dentro de otro sistema

La minería de opiniones puede ser usada como una etapa de otro sistema de minería de textos:

- En sistemas de recomendaciones, puede evitar que se ofrezcan artículos que tienen muy mala reputación.
- En anuncios en páginas web, convendría presentar solo marcas que tienen una buena impresión en los usuarios e incluso aumentar su frecuencia de aparición cuando se detecte que están siendo calificadas positivamente.
- En la recuperación de información, al eliminar oraciones subjetivas, se podrían mejorar los resultados.
- En los sistemas de pregunta respuesta, cuando se tiene una consulta orientada a una opinión, se le necesita dar un tratamiento especial: una respuesta basada no en hechos factuales sino en información subjetiva, pues es eso lo que el usuario necesita.

Aplicación en los negocios

La aplicación en los negocios es una de las razones más importantes dentro de las empresas para desarrollar este campo.

Cuando se desean conocer las razones del éxito o fracaso de un producto, es de gran utilidad contar con las opiniones que dieron fruto al prestigio actual de ese artículo, ya sea para continuar con su campaña publicitaria o hacer algún cambio de estrategia o en el producto.

Capítulo 3

Metodología

En el capítulo 2 se presentaron algunas generalidades de los sistemas de minería de opiniones. En este capítulo, se describen los métodos usados en este proyecto para la creación del sistema de clasificación de opiniones.

Antes de describir el sistema, es necesario presentar las herramientas de programación utilizadas para su creación.

3.1. Herramientas de programación utilizadas

3.1.1. Python

Se eligió el lenguaje de programación Python[1] debido a que es un lenguaje simple con excelente funcionalidad para procesar información lingüística [24].

Las fortalezas más importantes de Python, según [43], son:

- La librería estándar de Python es suficientemente amplia para que sea considerado apto para la resolución de cualquier tipo de problema informático (desarrollo Web, acceso a bases de datos, aplicaciones de escritorio, desarrollo científico y numérico, entre otros).

[1] http://www.python.org/

- Es compatible con otros lenguajes de programación. Se puede integrar con librerías de Java, a través de Jython[2]. También es compatible con .NET (lenguajes como C#, Visual Basic o F#) por medio de IronPython[3]. Incluso puede acoplar módulos desarrollados en C o C++, cuando no se encuentra una librería adecuada o se requiere del desempeño del código de bajo nivel.
- Python puede ser ejecutado en cualquier sistema operativo moderno, por ejemplo Windows, Unix/Linux, OS/2, Mac, Amiga, entre otros.
- Python está implementado bajo una licencia tipo *open source*. Esto hace que sea utilizado y redistribuido libremente.

La versión de Python utilizada fue la 2.6.4.

Existen tres módulos para Python que fueron fundamentales para el desarrollo de este trabajo, a continuación se describen brevemente cada uno de estos módulos:

- NLTK (*Natural Language Toolkit*)[4]: conjunto de librerías enfocadas al PLN. Es ideal para la investigación y desarrollo en esta área y también en las relacionadas como minería de textos, inteligencia artificial y aprendizaje de máquinas. NLTK ofrece clases básicas para representar datos relevantes al PLN, así como interfaces para realizar tareas tales como etiquetado PoS (*Part of speech*), búsqueda de n-gramas, obtención de frecuencias de aparición, entre otras muchas tareas útiles. También cuenta con una amplia documentación y una actividad bastante grande por parte de sus usuarios en foros y en listas de correos.
- Numpy[5]: es un módulo para Python que incluye un gran soporte para arreglos, vectores y matrices multidimensionales así como las funciones matemáticas (del álgebra lineal) para este tipo de objetos. También incluye sus propios tipos de datos, usados en los elementos de las matrices.
- Scipy[6]: Es una librería de algoritmos y herramientas matemáticas para Pyt-

[2]Jython es una implementación de Python para la máquina virtual de Java.
[3]IronPython es una implementación de Python orientado a la tecnología .NET.
[4]http://www.nltk.org
[5]http://numpy.scipy.org
[6]http://www.scipy.org

hon. Contiene el modulo de álgebra lineal, que resulta indispensable para el algoritmo de agrupamiento ya que incluye al objeto que representa a una matriz dispersa, útil para crear y manipular matrices con una gran cantidad de ceros de forma eficiente. Scipy le permite a Python ser un lenguaje muy utilizado en la investigación científica actualmente.

También se usaron los módulos IMDBpy[7] y Beautiful Soup[8]. IMDb (*Internet Movie Database*) es una base de datos en línea con información acerca de películas, series de televisión, actores, equipos de producción, videojuegos, entre otros temas. Permite a cualquier usuario registrado en el sitio escribir una reseña y asignar una calificación de 1 al 10 a cualquiera de los temas mencionados. La calificación general de una película, por ejemplo, es el promedio de todas las calificaciones vertidas por los usuarios sobre esa película. IMDBpy es utilizado para recuperar y administrar la información de IMDb . Sin embargo, al momento de programar el sistema, IMDBpy no ofrecía forma de recuperar los comentarios de cada película. Es por esto que fue necesario programar un Web crawler, cuya explicación está más abajo. Para programar este crawler se necesita el módulo Beautiful Soup, el cual realiza la función de analizar código HTML y XML. Tiene la ventaja de poder trabajar con lenguaje de marcado mal formado y provee métodos para navegar, buscar y modificar el árbol de análisis de un documento.

3.1.2. Eclipse IDE (*Integrated Development Environment*)[9]

Eclipse es un ambiente de desarrollo de software conformado por un entorno de desarrollo integrado y un sistema de *plugins* o complementos. Está escrito principalmente en Java. Puede ser usado para desarrollar aplicaciones en Java, por medio de complementos en Ada, C, C++, COBOL, Perl, PHP, Python, Ruby y Scheme, entre otros.

La versión Galileo 2.5 fue la utilizada para la programación en Python.

[7] http://imdbpy.sourceforge.net/
[8] http://www.crummy.com/software/BeautifulSoup/
[9] http://www.eclipse.org

3.1.3. Pydev[10]

Pydev es el complemento que permite programar en Python, Jython e IronPython en el ambiente de desarrollo Eclipse. Ofrece refactorización de código, depuración gráfica y análisis de código automático, entre otras útiles funciones. La versión utilizada fue la 1.6.4. Una vez instaladas y configuradas las tres herramientas, se comenzó con la programación del sistema.

3.2. Procesos del sistema

En esta sección se describen los procesos que comprenden al sistema. Así, en la figura 3.1 se pueden apreciar los procesos principales.

Dentro del sistema se pueden identificar ocho procesos principales, con sus respectivos datos o archivos de entrada y de salida. A continuación se detallará cada uno de ellos.

3.2.1. Obtención de artículos sobre películas desde Wikipedia[11]

Por medio de la opción Special:Export[12] que ofrece Wikipedia, se pueden exportar los artículos que cumplan con cierta categoría elegida por el usuario. De esta forma se extrajeron dos archivos XML, uno para películas de 2009 y otro para películas de 2010 (*2009 films* y *2010 films*, respectivamente).

Un fragmento del contenido de este tipo de archivos aparece en la figura 3.2. Estos archivos están bien formados de acuerdo a los lineamientos de XML. Sin embargo, contienen un lenguaje de marcado propio de Wikipedia, el cual hay que analizar para poder extraer los nombres y fechas. Las fechas son necesarias ya que, para limitar la cantidad de información, se usaron solo las películas estrenadas en noviembre y diciembre de 2009 y las estrenadas en enero de 2010.

[10] http://pydev.org/

[11] Enciclopedia libre, en línea y poliglota de la fundación Wikimedia. Cuenta con más de 3.5 millones de artículos en inglés, mismo idioma de las reseñas y opiniones sobre películas usadas en esta tesis.

[12] http://en.wikipedia.org/wiki/Special:Export

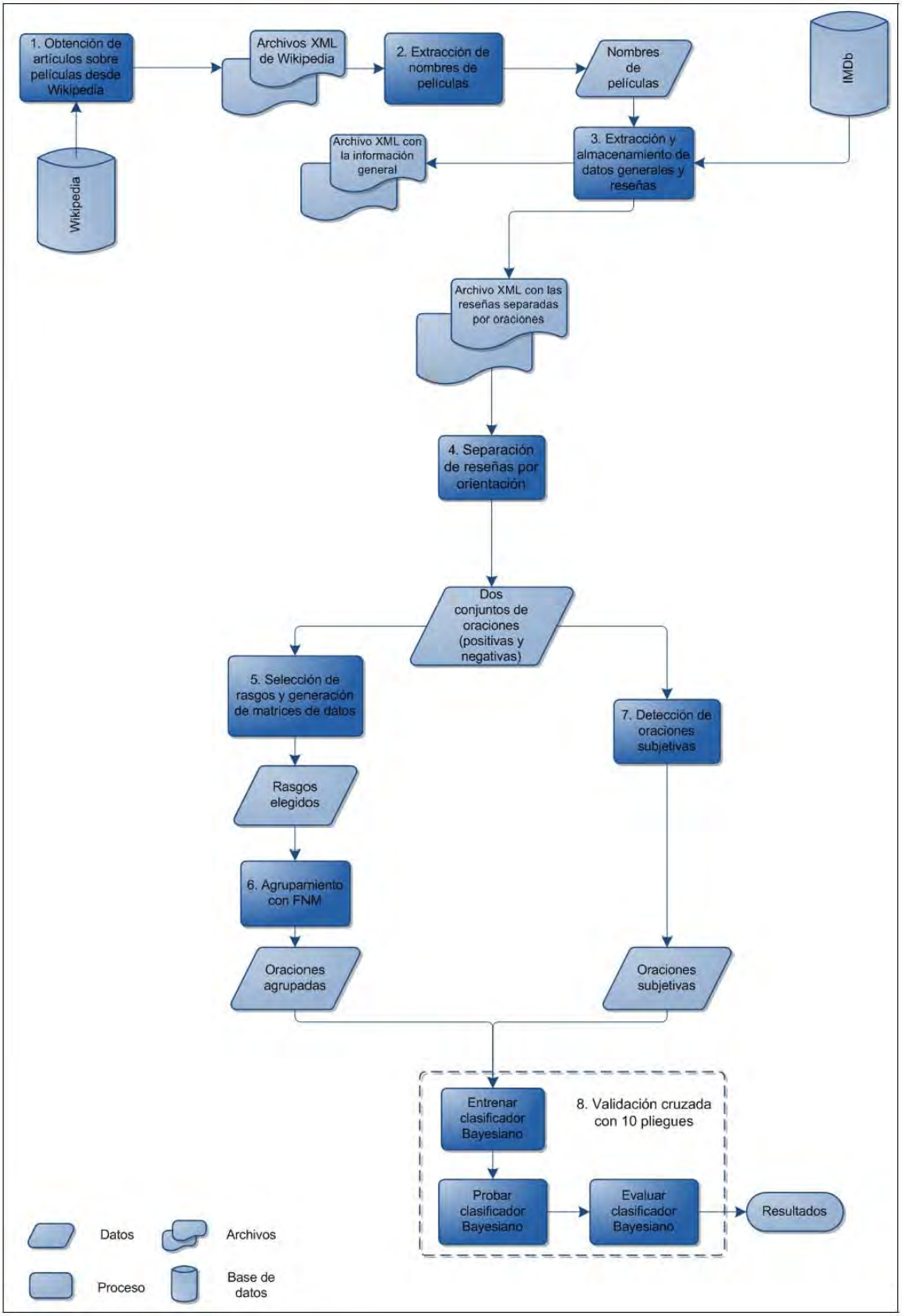

Figura 3.1: Procesos que comprenden al sistema de clasificación - La figura muestra los procesos que se realizaron para clasificar los enunciados dependiendo de su opinión

```
[[Category:2009  in India]]
[[Category:2009 films | ]]
[[Category:Lists of 2009 films by country or language|Tamil]]</text>
    </revision>
  </page>
  <page>
    <title>Category:Lists of 2009 films by country or language</title>
    <id>23835047</id>
    <revision>
      <id>306315314</id>
      <timestamp>2009-08-06T01:11:30Z</timestamp>
      <contributor>
        <username>Rich Farmbrough</username>
        <id>82835</id>
      </contributor>
      <text xml:space="preserve">{{Films by country or language|2009}}</text>
    </revision>
  </page>
  <page>
    <title>(500) Days of Summer</title>
    <id>18057739</id>
    <revision>
      <id>341125687</id>
      <timestamp>2010-01-31T18:43:13Z</timestamp>
      <contributor>
        <ip>68.4.102.67</ip>
      </contributor>
      <comment>/* Home media */</comment>
      <text xml:space="preserve">{{Infobox film
| name           = (500) Days of Summer
| image          = Five hundred days of summer.jpg
| caption        = Promotional film poster
| director       = [[Marc Webb]]
| producer       = Mason Novick &lt;br /&gt;Jessica Tuchinsky &lt;br /&gt;[[Mark Waters (director)|Mark Waters]]
&lt;br /&gt;Steven J. Wolfe
| writer         = Scott Neustadter &lt;br /&gt;Michael H. Weber
| narrator       = [[Richard McGonagle]]
| starring       = [[Joseph Gordon-Levitt]] &lt;br /&gt;[[Zooey Deschanel]] &lt;!-- top listed only (see Poster).
see Cast for more details. --&gt;
| music          = [[Mychael Danna]] &lt;br /&gt;Rob Simonsen
```

Figura 3.2: Archivo `2009_films.xml` - La figura muestra un fragmento del contenido de uno de los archivos XML extraídos desde Wikipedia.

La información buscada está contenida en estructuras llamadas *infoboxes* (o cajas de información) donde aparecen organizados los datos de cada película. Sin embargo, existen ocasiones en las que los datos no están completos, contienen caracteres erróneos o cambian su formato de una película a otra. Estos problemas son solucionados en gran medida durante la siguiente etapa.

3.2.2. Extracción de los títulos de las películas

Con los archivos de Wikipedia obtenidos, se procedió a extraer los títulos o nombres de las películas que en ellos aparecen. Esta tarea se hizo con ayuda de expresiones regulares. Se extrajo el título o nombre de la película y su fecha de estreno (por ejemplo, del archivo que aparece en la figura 3.2, se extrajo el nombre de la película *(500) Days of Summer*). Como se dijo anteriormente, la fecha sirvió para limitar la cantidad de títulos de películas (y por ende de películas) a procesar.

Así, se filtraron los nombres de películas de acuerdo a su fecha de estreno. Los nombres de las que se encontraban dentro del rango usado (noviembre-diciembre 2009 y enero 2010) fueron almacenados en un objeto tipo lista para después buscar su información en la siguiente etapa.

En ciertas ocasiones los meses de las fechas de estreno aparecen con letras (January 2010) y en otras ocasiones, con número (09 - 2009). Esto obligó a generar expresiones regulares que atiendan cada uno de estos casos.

También, algunos títulos contenían caracteres especiales[13], los cuales fueron removidos para evitar problemas al almacenar estos títulos en nuevos archivos XML.

3.2.3. Extracción y almacenamiento de datos generales y reseñas

Con los nombres de las películas se buscaron y extrajeron los datos adicionales y las reseñas de cada película. Como ya se dijo, esta etapa fue realizada con la

[13]En http://meta.wikimedia.org/wiki/Help:Special_characters hay una lista con estos caracteres especiales.

información contenida en el sitio IMDb. De esta manera, se buscó cada nombre de película en IMDb, se extrajo su identificador único dentro de la base y a partir de este se adquirió la siguiente información:

- Titulo en inglés[14].
- Títulos en otros idiomas, en esta base conocidos como *aliases*.
- Calificación o *rating*. Su rango es de 1 a 10 y es la calificación[15] que le dieron los usuarios por medio de sus votos a cada película.
- Número de votos.
- El identificador único, el cual es un número de siete cifras o más.

Después, con el identificador único, se procedió a extraer los comentarios desde las páginas Web que los contienen por medio del Web crawler. Este se encarga de recorrer página por página y recuperar cada uno de los comentarios. Al momento de desarrollar el sistema, las direcciones de las páginas de comentarios tenían la forma:

http://www.imdb.com/title/tt`XXXXXXX`/usercomments?filter=best;start= donde `XXXXXXX` es el identificador único de la película.

El crawler se encargó de obtener el número de páginas de comentarios, de descargar cada página, analizar el código HTML y obtener la información deseada de cada comentario. La información obtenida es la siguiente:

- Utilidad: cada reseña puede ser calificada por otros usuarios, quienes determinan si es útil o no.
- Título del comentario.

[14] Algunas veces fue necesario buscar el título de la película por medio de la fecha de estreno.

[15] Según http://www.imdb.com/help/show_leaf?ratingsexplanation, el promedio está ponderado (*weighted*) con el fin de reducir o eliminar los votos de individuos cuyo fin es cambiar el rating actual de la película en lugar de dar una opinión sincera acerca de ella. El procedimiento para obtener este promedio no es revelado.

- Rating: calificación que el usuario le dio a la película; en un rango de 1 a 10; expresado en el número de estrellas marcadas de diez posibles.

En la figura 3.3 se muestran en su forma original los datos extraídos.

Después de extraer los datos y reseñas, se procedió a almacenarlos de una forma estándar para poder ser ocupados después por el clasificador u otro sistema. Así, ya no es necesario realizar todo el proceso de consultar IMDb, ahorrando tiempo y recursos. Las reseñas se dividieron en oraciones por medio del segmen-

Figura 3.3: Forma en la que se encuentran los datos en la página original de IMDb - La figura muestra una parte de la información, como aparece en el sitio, que se extrajo con el crawler.

tador de oraciones Punkt, que ofrece[16] NLTK. También se eliminaron caracteres especiales. Esto para poder analizarlas y alimentarlas a los algoritmos de clasificación y agrupamiento.

Se generaron dos archivos XML. Un archivo para los datos de cada película, `peliculas.xml` y otro exclusivamente con las reseñas de cada película, `reseñas.xml`.

En el archivo XML de los datos de cada película, `peliculas.xml`, existe el elemento padre `movies`, el cual solo tiene un elemento hijo, `title`, cuyo contenido es el nombre de la película, en inglés. Este elemento `title` tiene tres atributos:

[16]El cual se detalla en: http://nltk.googlecode.com/svn/trunk/doc/api/nltk.tokenize.punkt.PunktSentenceTokenizer-class.html

1. `IMDBid`: el identificador único de la película.

2. `ratingG`: el rating promediado de la película.

3. `votes`: el número de votos que tiene la película.

En la figura 3.4 se observa un fragmento del archivo `peliculas.xml` (datos de cada película).

```
<?xml version="1.0" encoding="UTF-8"?>
 <movies>
    <title IMDbID="1333093" ratingG="3.9" votes="15">Love Me Again (Land Down Under)</title>
    <title IMDbID="0235158" ratingG="7.5" votes="16">Aanaval Mothiram</title>
    <title IMDbID="0233469" ratingG="5.2" votes="23172">Collateral Damage</title>
    <title IMDbID="1343046" ratingG="6.7" votes="97">Meitantei Conan: Shikkoku no chaser</title>
    <title IMDbID="1292655" ratingG="4.8" votes="75">The River Within</title>
    <title IMDbID="1571728" ratingG="NA" votes="NA">First Ink</title>
    <title IMDbID="0289226" ratingG="8.3" votes="18">House Full</title>
    <title IMDbID="1202230" ratingG="7.6" votes="23">Courting Condi</title>
    <title IMDbID="1392996" ratingG="7.5" votes="31">Sarah's Choice</title>
    <title IMDbID="0448182" ratingG="6.0" votes="119">Yesterday Was a Lie</title>
    <title IMDbID="1459219" ratingG="7.0" votes="14">Actresses</title>
    <title IMDbID="1397502" ratingG="8.0" votes="21">Grown Up Movie Star</title>
    <title IMDbID="1059925" ratingG="6.5" votes="1394">Greta</title>
    <title IMDbID="1341167" ratingG="9.0" votes="62">Four Lions</title>
    <title IMDbID="0115195" ratingG="6.9" votes="2407">Gulliver's Travels</title>
    <title IMDbID="1488591" ratingG="7.6" votes="10">Reiton kyôju to eien no utahime</title>
    <title IMDbID="1345777" ratingG="7.9" votes="843">Ishqiya</title>
    <title IMDbID="0085776" ratingG="8.0" votes="201">Katha</title>
    <title IMDbID="0860906" ratingG="8.7" votes="217">Evangerion shin gekijôban: Ha</title>
    <title IMDbID="1066327" ratingG="6.1" votes="17">London Betty</title>
    <title IMDbID="1148165" ratingG="6.9" votes="287">Bran Nue Dae</title>
    <title IMDbID="0120255" ratingG="7.8" votes="15807">The Sweet Hereafter</title>
    <title IMDbID="0878804" ratingG="7.7" votes="25484">The Blind Side</title>
    <title IMDbID="0367942" ratingG="7.3" votes="10">Kutty</title>
```

Figura 3.4: Archivo `peliculas.xml` - La figura muestra un fragmento del archivo XML que contiene la información de cada película. Cuando no se pudo obtener una calificación y/o el número de votos (porque IMDb aún no contaba con esos datos), el valor fue ocupado por las letras "NA" (*Not Available*).

En el archivo XML con las reseñas, `reseñas.xml`, el elemento padre es `todas_reviews`, y su elemento hijo es `comment`, cuyo contenido es toda la reseña de un usuario sobre una película. Este elemento `comment` tiene tres atributos:

1. `IMDBid`: el identificador único de la película.

2. `u`: la utilidad de la reseña que se expresa como x/y. Donde x es el número de usuarios que encontraron útil el comentario de y usuarios que lo revisaron.

3. `r`: rating que el autor de la reseña le dio a la película.

Además, comment tiene dos elementos hijos:

1. `titleC`: es el título de la reseña.

2. `s`: que marca los enunciados de la reseña y ocurre tantas veces como enunciados haya en la reseña en cuestión (recordar que las reseñas fueron segmentadas en oraciones en la sección 3.2.3).

En la imagen 3.5 se observa un fragmento de `reseñas.xml` (con las reseñas de cada película).

Figura 3.5: Archivo `reseñas.xml` - La figura muestra un fragmento del archivo XML que contiene las reseñas de cada película.

Al final, se recuperaron 171 películas y 7,089 reseñas. Las reseñas contaron con 123,878 enunciados en total.

El archivo `reseñas.xml` es el que se utilizó para realizar los experimentos que se describirán a continuación.

En la siguiente etapa, los elementos <s/> (enunciados de los comentarios) se separaron por orientación positiva o negativa.

3.2.4. Separación de reseñas por orientación

En esta etapa se separaron los enunciados de las reseñas en dos conjuntos diferentes. La separación se hizo de acuerdo a la calificación que el comentario dio a la película reseñada. Se consideró que los comentarios que exhibieron un rating (del autor de la reseña) menor a cinco, son de orientación negativa y los que tuvieron un rating igual o mayor a cinco eran de orientación positiva. Solo se trabajó con las primeras 50 reseñas de cada película, para evitar la dominación del corpus por parte de un número pequeño de películas populares.

No se utilizaron los comentarios que no asignaron una calificación a la película. Asimismo, se utilizaron solo las reseñas cuya utilidad fue superior al 50 %. Esto para tratar de eliminar comentarios sin lógica o comentarios basura (anuncios comerciales, letras aleatorias, entre otros).

Los conjuntos de oraciones se almacenaron en memoria para continuar con el proceso del sistema. Las cantidades de oraciones, en los conjuntos de orientación negativa y positiva, son:

- 12,998 enunciados para la orientación negativa.
- 48,323 enunciados para la orientación positiva.

Como se puede apreciar, la cantidad de enunciados positivos es casi cuatro veces mayor a la cantidad de enunciados negativos. Esto se debe simplemente a que existen un mayor número de reseñas positivas que negativas en las películas utilizadas.

Una vez separadas las oraciones, se siguieron dos líneas de experimentación: la agrupación de los comentarios por medio de FNM y la detección de oraciones subjetivas[17]. Después, con las oraciones agrupadas o con las oraciones identificadas como subjetivas, se entrenó un clasificador bayesiano ingenuo.

Primero se explicará el método usado para agrupar por medio de FNM y posteriormente se explicará la identificación de oraciones subjetivas.

3.2.5. Selección de rasgos y generación de matrices de datos

Antes de aplicar el método de agrupación, se generaron las matrices de datos necesarias como entrada para el algoritmo FNM.

Esto es, se crearon dos matrices de datos, una para cada una de las orientaciones (una para la positiva y otra para la negativa). Cada matriz requirió de un vector de rasgos. Estos rasgos pueden ser cada una de las palabras de las oraciones de cada orientación. Sin embargo, para limitar el tamaño del vector de rasgos, se usaron aquellos que, en teoría, ofrecerían mayor información acerca de un comentario positivo o negativo.

Los rasgos que se utilizaron para cada una de las matrices fueron elegidos con base en la experimentación. Los que resultaron más exitosos y, por ende, fueron utilizados se enumeran a continuación:

- Trigramas que aparecen más de 16 veces. Estos trigramas además no deben contener un sustantivo en la segunda posición del trigrama. La razón fue que, con base en la experiencia, los trigramas que contienen sustantivos en esa posición generalmente eran nombres propios, ya sea de películas, actores, directores, etc.
- Adjetivos que aparecen más de cinco veces.
- Palabras (unigramas) que aparecen más de 40 ocasiones y que no se encuentren en una lista de paro.

[17]Como se vio antes, una oración subjetiva es aquella que describe el aprecio o jucio acerca de algo.

Estos rasgos son los que conformaron las columnas de las matrices de datos.

Fue necesario tokenizar las oraciones con el fin de buscar trigramas, etiquetarlos y obtener frecuencias de aparición. Estas tareas se llevaron a cabo con la ayuda del NLTK, que ofrece varios métodos[18] para tokenizar. El utilizado fue el basado en expresiones regulares, que conserva palabras, palabras con guiones, palabras con apóstrofes y números. Estas expresiones fueron construidas con base en las propuestas por [24, 44]. El etiquetado PoS se realizó también con la ayuda del NLTK, usando, como se dijo antes, el conjunto de etiquetas Penn Treebank. Las etiquetas buscadas fueron JJ (adjetivo), JJR (adjetivo comparativo), JJS (adjetivo superlativo) y VBG (adverbio) y NN (sustantivo).

Se generaron entonces las dos matrices, donde los renglones eran los enunciados de las reseñas y las columnas los rasgos seleccionados. El valor de cada celda es la frecuencia absoluta de cada palabra o trigrama en el texto. Se probó también con el peso tf-idf[19], pero los resultados fueron mejores con la frecuencia.

Las matrices creadas fueron del tipo dispersas, ya que la gran mayoría de sus elementos son cero.

Las características de las matrices son:

- La matriz de orientación negativa tuvo una dimensión de $351 \times 12,998$, con 27,878 elementos distintos a cero.
- La matriz de orientación positiva tuvo una dimensión de $1,470 \times 48,323$, con 195,699 elementos distintos a cero.

La figura 3.6 muestra un fragmento de la matriz de orientación negativa representada como una imagen. Esta imagen, en su totalidad, tiene por dimensiones $12,998 \times 351$ pixeles, por lo tanto, cada valor de la matriz está representado por un pixel. Los valores distintos a cero se muestran en color negro.

Con estas matrices se procedió a realizar el agrupamiento FNM.

[18] Estos métodos se pueden revisar en: http://nltk.googlecode.com/svn/trunk/doc/howto/tokenize.html.

[19] El peso tf-idf es una medida estadística usada para determinar la importancia de una palabra en un documento de un corpus. El peso incrementa proporcionalmente a la frecuencia relativa (o absoluta) de la palabra en el documento y disminuye proporcionalmente a la cantidad de veces que aparece esa palabra en otros documentos del corpus.

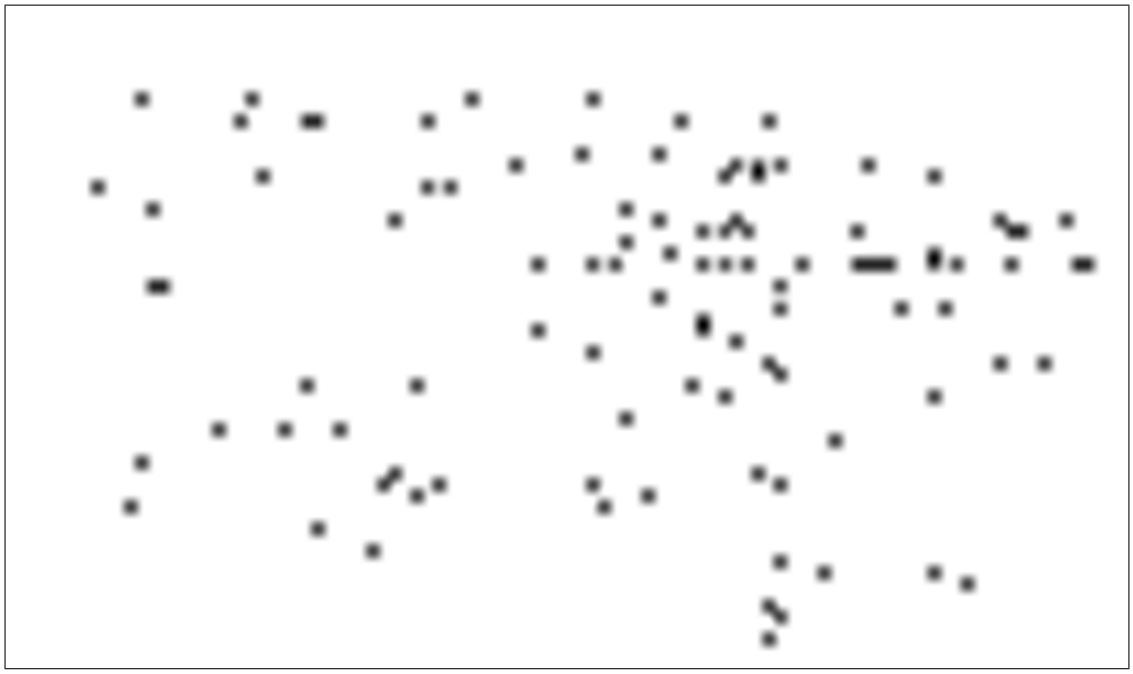

Figura 3.6: Matriz dispersa - Fragmento de una imagen que representa la matriz dispersa de orientación negativa. Los elementos en negro son aquellos mayores a cero.

3.2.6. Agrupamiento con factorización no negativa de matrices (FNM)

El objetivo de agrupar las oraciones es obtener aquellas que contengan las palabras que mejor definen a cada orientación; obteniendo los enunciados más representativos. De esta forma, en los rasgos de las matrices, se cuenta no solo con los unigramas y trigramas más comunes, sino también con las palabras que rodean a estos n-gramas. Con esto se pretende reducir la cantidad de información necesaria para entrenar el clasificador. Otra razón para realizar el agrupamiento es entrenar el clasificador solo con oraciones subjetivas, esto es, dejando de lado las que no ofrecen una opinión.

Como se vio en el capítulo dos sección 2.3.4.2.1, la agrupación por medio de FNM consiste en factorizar la matriz de datos de entrada en dos matrices de menor dimensión. Ambas matrices después se interpretan para obtener los grupos a los cuales pertenece cada oración.

Las matrices de entrada, para cada agrupamiento, fueron: la matriz de orientación negativa y la de orientación positiva, creadas en la etapa anterior. Estas matrices se descompusieron en dos matrices cada una.

Al ejecutar el algoritmo de FNM, es necesario indicar el valor r, que en términos prácticos es el número de grupos que se desean obtener. Se eligió $r = 15$ para los dos procesos de agrupamiento.

Los resultados se almacenaron en memoria para ser procesados en la siguiente etapa. También fueron guardados en archivos de texto plano.

En la figura 3.7 se aprecia un fragmento de los resultados entregados por el agrupamiento:

Cuando se encuentran los grupos, a cada enunciado miembro le corresponde un valor numérico, llamado valor de pertenencia al grupo, el cual es obtenido de las matrices factorizadas. Esta cantidad indica qué tanto pertenece ese enunciado al grupo, por lo que los que tienen un valor mayor, son los más representativos del grupo. Por esta razón, para entrenar el clasificador bayesiano ingenuo, se tomaron los enunciados cuyo valor de pertenencia al grupo fue mayor a la unidad.

Al terminar el proceso de agrupamiento, se obtuvieron las siguientes cantidades de oraciones, cada una con valores de pertenencia superior a uno:

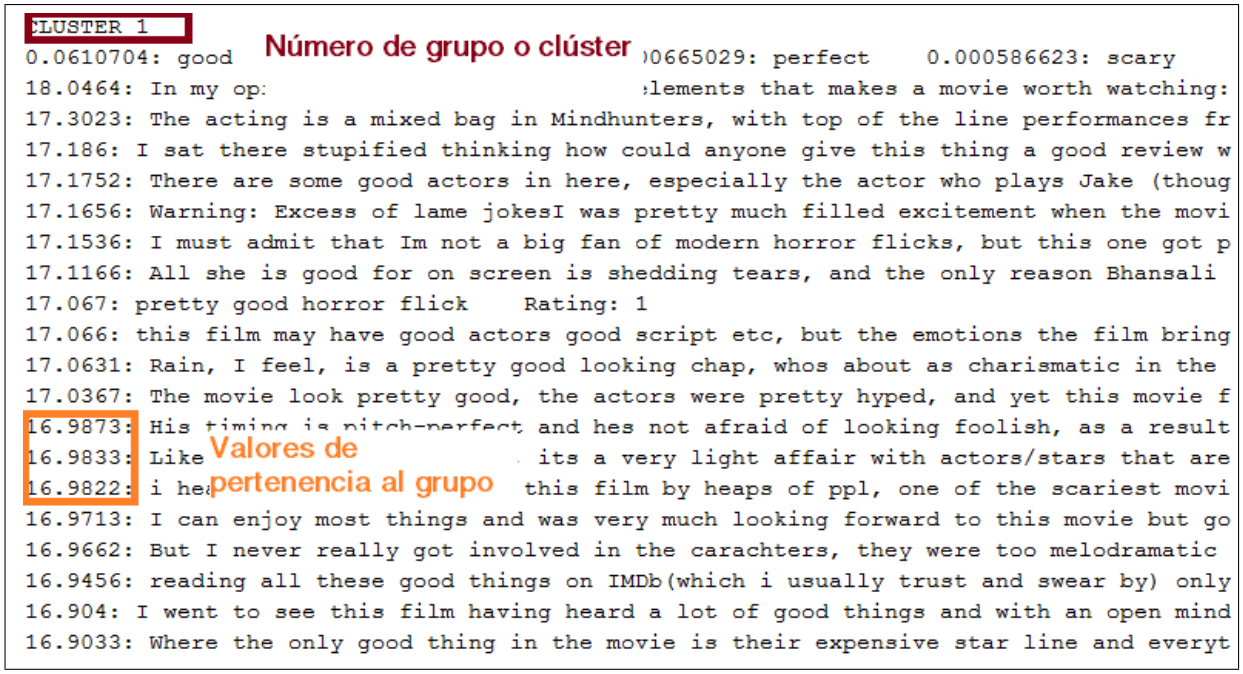

Figura 3.7: Archivo `NMFpositivos.txt` - La figura muestra un fragmento del archivo de texto con el resultado del agrupamiento. Se observan el número del grupo y el valor de pertenencia al grupo de cada enunciado miembro.

- Para la orientación negativa: 5,740 oraciones.
- Para la orientación positiva: 30,058 oraciones.

Con estos enunciados se entrenó el clasificador bayesiano ingenuo, en el marco de la primera línea de experimentación.

La siguiente sección describirá el procedimiento para detectar oraciones subjetivas. Este procedimiento comprende la segunda línea de experimentación.

3.2.7. Detección de oraciones subjetivas

También con los dos conjuntos de enunciados ya separados por calificación como entrada, y como segunda línea de experimentación, se detectaron y utilizaron solo aquellas oraciones subjetivas del texto para entrenar el clasificador.

Se tomaron los tres enfoques descritos en las siguientes secciones. Cada uno de los procedimientos tuvo como entrada los mismos dos conjuntos mencionados y cada uno entregó igualmente dos conjuntos de oraciones, separadas por la orientación de la opinión que en ellas existe.

3.2.7.1. Oraciones con adjetivos o adverbios

De acuerdo con trabajo previo citado en [2] y descrito en [3], los adjetivos y adverbios son indicadores de subjetividad. Por lo tanto, las oraciones que tuvieron al menos uno de estos fueron seleccionadas para entrenar el clasificador.

Se tokenizó cada oración de cada grupo y posteriormente se les aplicó un etiquetado PoS.

Una vez etiquetados los tokens, se filtraron dependiendo de si contenían o no un adjetivo o un adverbio (permanecieron solo las oraciones que sí contenían alguno). Las etiquetas buscadas fueron JJ, JJR, JJS y VBG.

Después del filtrado, se obtuvieron las siguientes cantidades de oraciones:

- Conjunto de oraciones negativas: 6,387.
- Conjunto de oraciones positivas: 8,024.

3.2.7.2. Oraciones con disparadores de presuposición

Una presuposición es el conocimiento implícito de que una declaración debe ser mutuamente conocida o asumida por el hablante y el destinatario, para que la declaración sea considerada apropiada dentro del contexto [45, 46]

Un disparador de presuposición es una construcción que señala la existencia de una presuposición en una declaración.

En [45] se ejemplifican los tipos de disparadores usados para filtrar las oraciones que los contienen de las que no. Esto se hizo de nueva cuenta con la ayuda de expresiones regulares.

Después de realizar este filtrado se obtuvieron las siguientes cantidades:

- Conjunto de oraciones negativas: 1,049.
- Conjunto de oraciones positivas: 2,758.

3.2.7.3. Oraciones con disparadores o adjetivos o adverbios

Como tercer y último enfoque, se tomaron los enunciados que tenían disparadores o adjetivos y adverbios.

Los conjuntos obtenidos fueron:

- Conjunto de oraciones negativas: 7,436.
- Conjunto de oraciones positivas: 10,782.

3.2.8. Validación cruzada con 10 pliegues

Cada línea de experimentación entregó a esta etapa los dos conjuntos de oraciones, uno clasificado como de orientación positiva y otro clasificado como de orientación positiva. Estas clases, positiva y negativa, son las que el clasificador bayesiano ingenuo asignará después de ser entrenado.

En total, se tuvieron cinco pares de conjuntos (diez conjuntos en total). Uno por cada tipo de experimento:

1. Enunciados agrupados con FNM.

2. Enunciados subjetivos elegidos por presencia de adjetivos o adverbios.

3. Enunciados subjetivos elegidos por presencia de disparadores de presuposición.

4. Enunciados subjetivos elegidos por presencia de disparadores de presuposición o adjetivos o adverbios. Este experimento es la unión de los dos anteriores.

5. Todos los enunciados obtenidos en la etapa Separación de reseñas por orientación. Este entrenamiento fue realizado como método de control.

Los subprocesos de entrenamiento, pruebas y evaluación se engloban en el proceso de validación cruzada, ya que esta afecta a los tres procesos.

Como se explicó con anterioridad, en el capítulo dos sección 2.3.2.1.1, durante la validación cruzada con k - pliegues (en este caso $k = 10$) se dividen los datos disponibles (las oraciones) en k pliegues y se entrenan k clasificadores, cada uno con $k - 1$ (en este caso 9) diferentes pliegues. Posteriormente, se prueba cada clasificador con el pliegue con el que no se entrenó.

De esta forma, cada uno de los conjuntos de los cinco pares se dividió en 10 pliegues del mismo tamaño. Se tomaron solo 6,000 enunciados por polaridad y en cada experimento estos se seleccionaron aleatoriamente.

En la figura 3.8 se ilustra lo descrito para un solo conjunto de oraciones.

Recordando el capítulo dos sección 2.3.1.1, el clasificador bayesiano ingenuo requiere de información ya etiquetada con anterioridad para poder predecir la clase de un ejemplo nuevo no visto antes.

El clasificador recibe nueve pliegues en cada entrenamiento, obtiene frecuencias relativas de las palabras de los enunciados de cada uno de estos pliegues y un valor suavizado. Este valor se asigna durante las pruebas a las palabras que no se hayan encontrado en el entrenamiento y que por lo tanto no tienen frecuencia relativa conocida.

Cuando ya se ha entrenado el clasificador, entonces se prueba. La prueba consiste en predecir la clase para los enunciados del pliegue de pruebas (enunciados no vistos antes por el clasificador) y obtener las medidas de desempeño que evalúan el comportamiento del clasificador.

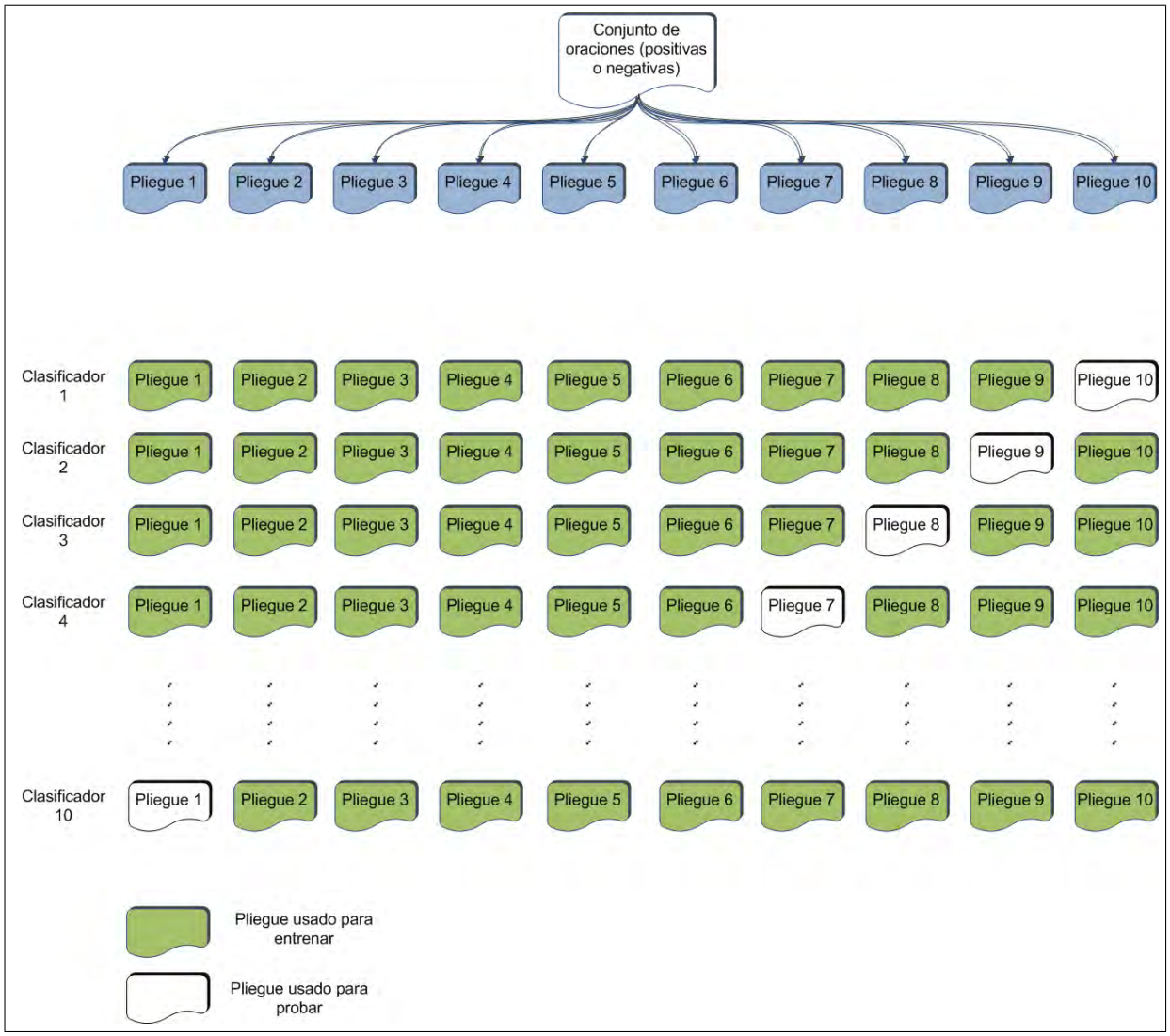

Figura 3.8: Validación cruzada con 10 pliegues - La figura ejemplifica como se realizó el entrenamiento y prueba del clasificador. Se toman solo $k-1$ pliegues para entrenar por cada clasificador y se prueba con los sobrantes. Todos los enunciados son utilizados.

Las medidas de precisión y error y también las medidas de la matriz de confusión sirven para evaluar el desempeño del clasificador y así determinar la efectividad de las técnicas usadas, o saber si cambiando algún parámetro o agregando alguna característica el sistema mejora su desempeño.

Capítulo 4

Resultados

4.1. Resultados generales

A continuación se presentan y analizan los resultados obtenidos por el clasificador creado.

En la tabla 4.1, se presentan los resultados promediados de la validación cruzada con 10 pliegues para cada uno de los cinco experimentos efectuados en este trabajo. En esa tabla se observa que todos los experimentos superaron el baseline de la simple predicción aleatoria de clases. Esta predicción implica una exactitud de 50 %. Como se ve en el primer renglón de la tabla, el mejor resultado de exactitud se obtuvo usando los enunciados con adjetivos o adverbios y el resultado más pobre se obtuvo usando los enunciados con disparadores de presuposición. Como era de esperarse, el error más alto se registró usando este último método, mientras que el error más bajo fue para el método usando los enunciados con adjetivos o adverbios. También se puede notar que los resultados de razón-tp y razón-fp son coherentes con los resultados de precisión y error obtenidos, es decir, los mejores resultados se aprecian en el primer renglón y los peores en el quinto (enunciados con adjetivos o adverbios y enunciados con disparadores de presuposición, respectivamente). Con respecto a la última columna, número de rasgos, se podría inferir que existe un umbral en el número de rasgos que afecta directamente el desempeño del clasificador, ya que el experimento con dispara-

dores de presuposición fue el que tuvo el menor número de rasgos y es posible que esta haya sido la causa del bajo desempeño del clasificador. Sin embargo, el resto de los experimentos (renglones 1 a 4) tuvieron un número de rasgos similar (alrededor de 20,000) y aún así sus desempeños cambiaron considerablemente, es decir, a pesar de tener un número similar de rasgos, los experimentos de los renglones 2, 3 y 4 no alcanzaron la exactitud del experimento del primer renglón.

	Método	**Exactitud**	**Error**	**Razón-tp**	**Razón-fp**	**Número de rasgos**
(1)	enunciados con adjetivos o adverbios	72.60	27.40	72.65	27.45	20,016.70
(2)	enunciados agrupados con FNM	69.30	30.70	69.08	30.48	19,332.20
(3)	enunciados con disparadores o adjetivos o adverbios	67.56	32.44	66.45	31.33	20,434.40
(4)	todos los enunciados	66.63	33.37	65.87	32.60	19,101.80
(5)	enunciados con disparadores de presuposición	63.84	36.16	64.50	36.83	9,277.80

Tabla 4.1: Resultados promediados de la validación con 10 pliegues del clasificador creado.

En la figura 4.1 se observan cinco puntos graficados en el espacio ROC, cada uno de ellos correspondiente a los cinco métodos usados para entrenar el clasificador. El punto correspondiente al uso de oraciones con adjetivos o adverbios es el que se encuentra más cercano a la clasificación perfecta (0.0,1.0), lo que indica que obtuvo el menor error falso positivo y clasificó mejor que los demás los casos positivos. En la figura 4.2 se presentan las curvas ROC, una para cada método utilizado. En ellas se confirman los resultados presentados anteriormente: el mejor resultado se obtiene entrenando el clasificador con enunciados que contenga adjetivos o adverbios. En ellas también se puede apreciar el trade-off que se obtendría si se deseara clasificar con mayor efectividad los casos realmente positivos. Por ejemplo, en el caso de los enunciados con adjetivos o adverbios, si se deseara una razon-tp de 90 %, se tendría que aceptar una razon-fp de poco más del 50 %.

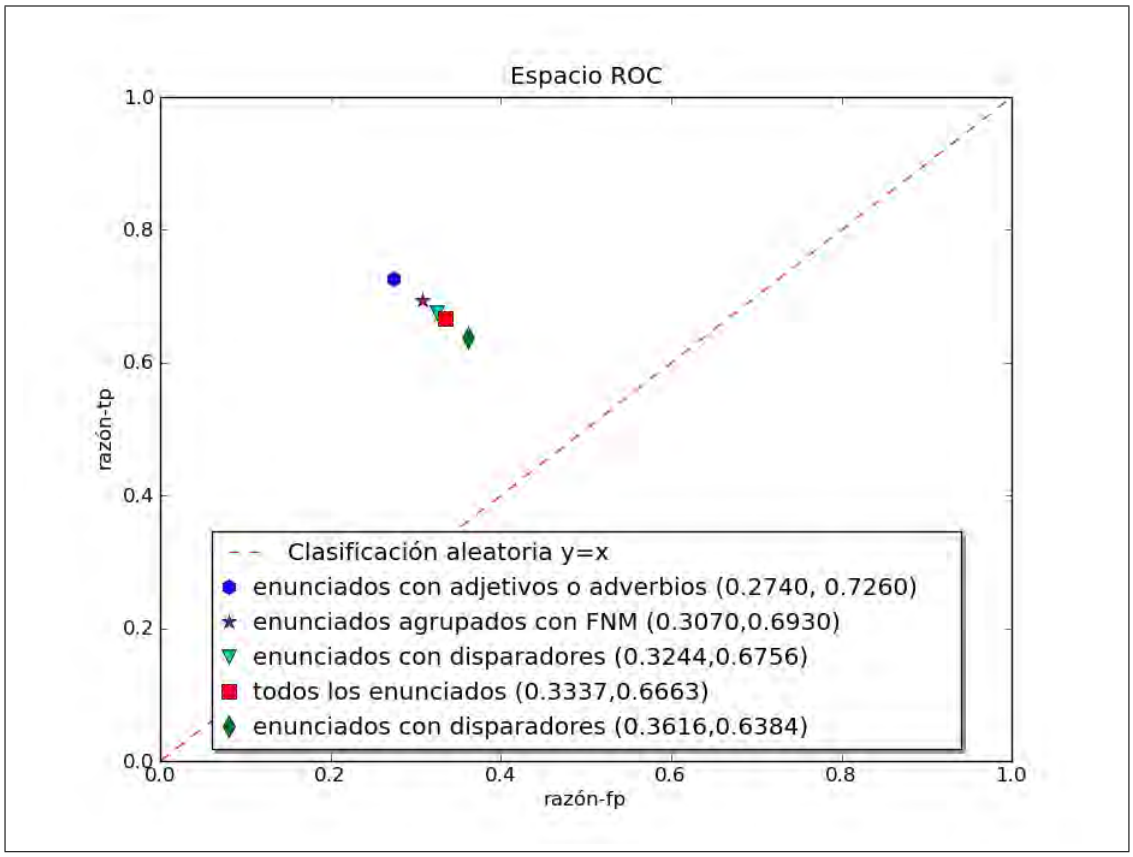

Figura 4.1: Ubicación de los clasificadores en el espacio ROC - La figura muestra los puntos que representan a cada una de las técnicas usadas para entrenar el clasificador. Cuando se usan enunciados con adjetivos o adverbios se logra la mejor posición.

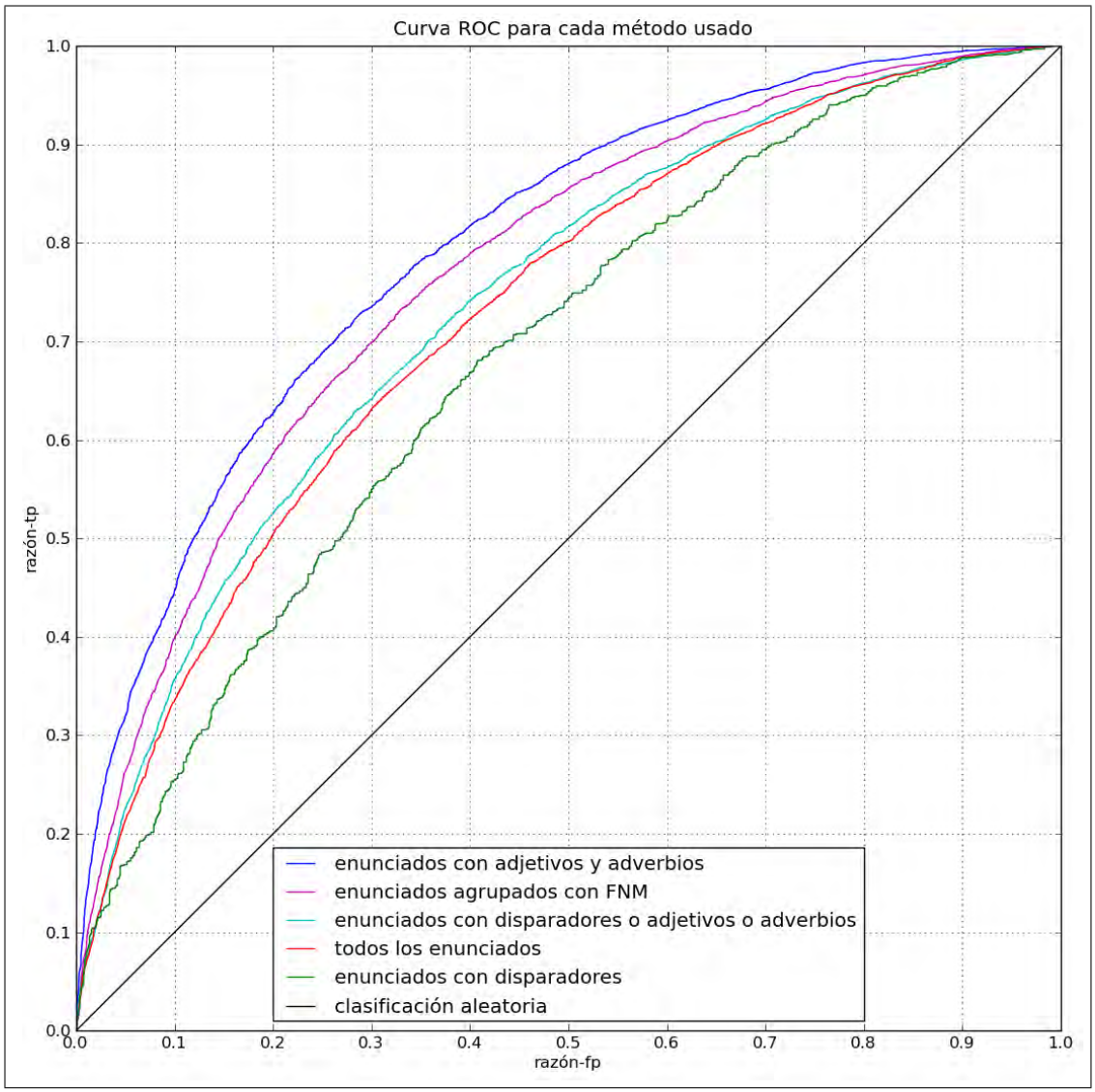

Figura 4.2: Curva ROC para cada método utilizado - La figura muestra la curva ROC obtenida por cada método usado para entrenar cada clasificador. Las curvas son congruentes con los otros resultados presentados: cuando se usan enunciados con adjetivos o adverbios se logra el mejor desempeño.

		Método	Exactitud	Error	Número de rasgos
Resultados con los datos de [47]	(1)	unigramas	79.63	20.36	23,486
	(2)	adjetivos	76.37	23.62	4,434
Resultados de [47]	(3)	unigramas	78.70	21.30	16,165
	(4)	adjetivos	77.00	23.00	2,633

Tabla 4.2: Resultados promediados de la validación cruzada con 3 pliegues del clasificador creado para cada experimento. Se observa que los valores de exactitud para cada experimento (unigramas y adjetivos) son muy similares para el clasificador creado en esta tesis y el creado en [47].

A pesar de que no son comparables los resultados obtenidos en este trabajo con los obtenidos en [47], debido a la diferente metodología seguida, se decidió realizar dos experimentos con el corpus de comentarios usado en [47] para conocer el desempeño del clasificador bayesiano programado en este trabajo. Los resultados obtenidos en estos experimentos se encuentran en la tabla 4.2.

La prueba consistió en reproducir dos de los experimentos reportados en ese trabajo. El primer experimento en cuestión es el que hace uso exclusivamente de los unigramas más frecuentes, así como el uso de palabras negadas[1]. Durante la reproducción del experimento no se hizo uso de las palabras negadas, porque se consideró innecesario; esto es, se utilizaron únicamente los unigramas más frecuentes.

Como se ve en el tercer renglón de la tabla 4.2, en ese trabajo se reportó una exactitud de 78.70 % para el primer experimento. Como se observa en el primer renglón, el resultado obtenido con el clasificador de este trabajo fue de 79.63 %.

En el segundo y cuarto renglón están los resultados para la segunda prueba (adjetivos más frecuentes). En esta prueba se obtuvo una exactitud de 76.37 % mientras que en el trabajo de comparación se obtuvo una exactitud de 77 %. Como se observa, la exactitud disminuye tal como en el trabajo citado.

Es importante notar que, como en [47], en ambos casos (con unigramas y con adjetivos) se usaron solo aquellos cuya frecuencia absoluta era mayor o igual a cuatro.

[1]El uso de palabras negadas consiste en concatenar la etiqueta "NOT_" a todas las palabras que siguieron, en una oración, a las negaciones *not, isn't, didn't*, etc.)

Con esta prueba se pudo concluir que los experimentos de este trabajo y los de [47] tienen una exactitud similar, aun sin considerar pasos especiales como el uso de palabras negadas. También se observa que a pesar de que la exactitud es menor usando solo adjetivos, la cantidad de rasgos utilizados también es mucho menor a la cantidad usada con los unigramas. Esto podría significar la importancia de los adjetivos y adverbios en la emisión de un juicio.

4.2. Enunciados con adjetivos o adverbios

Respecto a los experimentos llevados a cabo en este trabajo, los resultados pueden apreciarse en la tabla 4.1. A continuación se analizará cada uno de los experimentos realizados.

Las oraciones subjetivas, detectadas por medio de aparición de adjetivos o adverbios, resultó el método más exitoso, alcanzando 72.6 % de exactitud. Sus valores de razón-tp y razón-fp son considerablemente mejores que los de los métodos restantes. Esto resulta contradictorio con los resultados de [47], donde los mejores resultados son obtenidos con los unigramas únicamente (sin distinguir si son o no adjetivos u otra parte del discurso) y confirma que los adjetivos y adverbios tienen información que determina la opinión sobre algo, como se muestra también en [2].

Aunque cabe mencionar que además de los adjetivos y adverbios, se incluyeron todas las palabras de esas oraciones donde ocurrieron, intentando así incluir palabras que formaran parte del contexto de la opinión.

4.3. Enunciados agrupados con FNM

Este método resultó el segundo mejor con una exactitud de 69.3 %. Cabe recordar que este método usó los adjetivos y adverbios, los unigramas y los trigramas (sin sustantivo en la segunda palabra del trigrama) más frecuentes.

Si no se seleccionan los rasgos de las matrices de datos, los enunciados se agrupan generalmente, según lo experimentado, no por sentimiento sino por tema,

en este caso por película, lo cual no ayuda para determinar la orientación de la opinión.

4.4. Enunciados con disparadores de presuposición o adjetivos o adverbios

Se obtuvo una exactitud de 67.56 % con estos enunciados, elegidos solo si contenían algún disparador considerado o algún adjetivo o adverbio. Su desempeño fue mejor que si se tomaran solo los disparadores pero inferior a tomar solo los adjetivos o adverbios. Esto podría indicar que los disparadores no ofrecen información acerca de la opinión de una declaración.

4.5. Todos los enunciados

Este método sirvió como control, ya que se tomaron todas las palabras de todas las oraciones seleccionadas (aleatoriamente) de los datos de entrenamiento. Todos los métodos usados, excepto el que usa solo enunciados con disparadores de presuposición, superaron la exactitud de 66.63 % obtenida con este enfoque.

4.6. Enunciados con disparadores de presuposición

Con la menor exactitud reportada, de 63.84 %, este método entregó los resultados más pobres de toda la experimentación. Esto puede implicar que los disparadores no son más útiles para la detección de opiniones que simplemente usar las palabras de todos los enunciados, sin embargo usando una lista más amplia de disparadores o buscando otro tipo de características relacionadas a ellos, se podrían obtener mejores resultados.

Este bajo desempeño se podría deber a que el número de enunciados con disparadores es muy reducido, y los clasificadores requieren de una cantidad considerable de ejemplos (conocimiento previo) para poder predecir exitosamente nuevos casos.

Capítulo 5

Conclusiones

En este trabajo se desarrolló un sistema capaz de clasificar enunciados dependiendo de la opinión que cada uno expresa acerca de una película. Se cumplió entonces con el objetivo principal y con los objetivos específicos:

- Se extrajo la información desde la Web y se preprocesó para ser usada posteriormente en los procesos de agrupamiento, búsqueda de oraciones subjetivas y clasificación.
- Se agruparon automáticamente los enunciados usando un vector de rasgos constituido de unigramas, trigramas y adjetivos. Esto ocasionó que los enunciados se agruparan por compartir una opinión similar y no por el tema del que hablan (la película comentada).
- Se hizo un esfuerzo por separar las oraciones objetivas de las subjetivas mediante la detección de adjetivos y adverbios. También mediante la detección de disparadores de presuposición y mediante la detección de adjetivos, adverbios o disparadores.
- Se entrenó un clasificador bayesiano ingenuo con las oraciones cuya clase ya era conocida previamente (provenientes del agrupamiento FNM y de la detección de oraciones subjetivas). Después se probó con oraciones no vistas antes por el clasificador. Finalmente se evaluaron los resultados arrojados por las pruebas de cada clasificador.

Los resultados entregados por el clasificador superaron, como se mencionó antes, el baseline de una clasificación aleatoria.

Los resultados de exactitud obtenidos en este trabajo se podrían comparar con los obtenidos en [47] para un clasificador de tipo Bayes ingenuo. Sin embargo, los métodos seguidos en ese trabajo difieren de los seguidos en esta tesis. En ese trabajo la mejor exactitud obtenida para este tipo de clasificadores fue de 81.6 %, trabajando, como se dijo antes, con unigramas concatenados a su etiqueta PoS.

Aunado a la diferente metodología usada en [47], otras diferencias importantes entre este trabajo y aquel son:

- No se utilizó el mismo corpus de entrenamiento. Se extrajo uno desde el sitio web de IMDb y se procesó de forma que no es igual a la forma usada en el trabajo de referencia.

- Se clasificaron oraciones y no reseñas completas. A pesar de que, computacionalmente, una oración se podría considerar igual a una reseña completa (considerando una reseña completa como una sola línea de texto), una reseña completa contiene, obviamente, más palabras. Esta mayor cantidad de palabras podría ofrecer más información acerca de la orientación de la película, mientras que la cantidad reducida de palabras dentro de una oración podría ofrecer menos información acerca de la opinión de esa oración.

- No se realizaron algunos pasos tomados en el trabajo referido, como es el uso de palabras negadas y el uso de signos de puntuación como palabras separadas. También se evito el uso de listas de paro.

A lo largo de la metodología de esta tesis se realizaron pasos que podrían mejorarse en el futuro.

El primero de estos casos se encuentra en el proceso de extracción de información desde IMDb mediante el Web crawler. El Web crawler, al no ser una implementación propia de IMDb, está a merced de la continuidad del estilo y del formato HTML del sitio web. De hecho, en diciembre y enero de 2010 y 2011, el sitio de IMDb cambió considerablemente. De esta manera, si se quisiera repetir este experimento ahora, se tendrían que realizar modificaciones al Web crawler

creado en esta tesis. Sin embargo este problema es común en todos los módulos de este tipo y realmente se escapa del control del programador.

Durante el proceso de agrupamiento con FNM, se tomó un paso poco convencional: el uso exclusivo de trigramas y más aun, de aquellos sin sustantivo en la segunda posición. Como se dijo en la metodología, esta medida parece arbitraria, pero surge de la etapa de exploración del corpus, ya que durante la experimentación se vio que los trigramas, con esa limitación, son los que ofrecen mayor cantidad de frases juiciosas. De hecho, en los resultados experimentales, se observó una ligera mejoría en la exactitud con el uso de estas limitantes. Aun así, convendría hacer un análisis con mayor profundidad acerca de las diferencias existentes en el uso de diferentes n-gramas.

También en el mismo proceso, se decidieron experimentalmente los parámetros del algoritmo de factorización FNM. Estos valores son:

- `r`: Indica el número de grupos deseados.
- `tolerancia`: Indica qué tanto pueden ser diferentes el producto de las dos matrices encontradas y la matriz original.
- `Número máximo de iteraciones`: Indica cuántas iteraciones el algoritmo podrá realizar antes de detenerse.
- `Límite de tiempo`: Indica cuánto tiempo podrá llevarse a cabo la ejecución del algoritmo antes de detenerse.

Sería conveniente realizar pruebas más extensas acerca de la selección de parámetros. Sin embargo, para realizar estas pruebas es necesario dividir los datos de entrenamiento no en dos, entrenamiento y pruebas, sino en tres partes: entrenamiento, estimación de parámetros y pruebas. Este paso resultó imposible de realizar debido al poco tiempo disponible.

Se intentó detectar oraciones subjetivas por medio de disparadores de presuposición, lo cual no funcionó como se esperaba.

A pesar de estas limitantes, el sistema creado en este trabajo posee varias ventajas.

Como se mencionó, la detección de oraciones subjetivas por medio de disparadores de presuposición no funcionó como se esperaba, pero con este experimento se podría descartar su uso en la detección de opiniones. De todas maneras, antes de descartarlo valdría la pena explorar otros tipos de disparadores o si alguno de los usados es mejor que los demás.

Se usaron los adjetivos y adverbios y se confirmó que sí contienen información importante acerca de la orientación de una opinión. Aunque los adjetivos ciertamente dependen del contexto de la oración en la que se encuentran, ya que, como se observo en los resultados de [47], solos no son igual de efectivos.

La información generada y analizada durante la ejecución de los procesos del sistema puede ser reutilizada para otro tipo de proyectos o sistemas ya que a lo largo de las etapas del sistema, se almacena en archivos XML la información procesada, como es el caso de los archivos `peliculas.xml` y `reseñas.xml`. También se utilizaron archivos planos para almacenar las oraciones ya separadas por orientación.

El desarrollo de este sistema requirió de la elaboración de dos procesos que podrían ser utilizados en trabajos futuros, estos procesos son el agrupamiento mediante FNM y el clasificador bayesiano ingenuo. Cabe destacar que el método de FNM es de reciente aplicación con documentos y se ha demostrado ser más útil que otras técnicas de factorización de matrices [39]. Estos dos procesos son módulos independientes, es decir, no requieren de otras partes del sistema para funcionar y se pueden alimentar directamente con otros datos que necesiten ser agrupados o clasificados. En los apéndices C y D se explican los detalles de estos dos módulos de Python.

Este trabajo sirvió también como una introducción a los procesos de pruebas y validación, necesarios en casi cualquier tipo de sistema de minería de textos.

Asimismo, existen varias mejoras que se pueden realizar al sistema: cambiar el algoritmo de clasificación, utilizar uno más robusto como una máquina de vectores de soporte, el de esperanza-maximización, entre otros. También sería ideal ahondar mucho más en la identificación de las oraciones subjetivas dentro de un texto, y también encontrar, dentro de una oración subjetiva, aquellas que realmente hablen del tema del cual se ofrece la opinión. La detección de ironía también sería de gran utilidad para la clasificación de opiniones, ya que ayudaría a

asignar la clase “negativo” a comentarios negativos que podrían parecer positivos (irónicos). Realizar una clasificación de opiniones no solo binaria (negativa o positiva), sino agregando también la clase neutral, podría aumentar la exactitud en la prediccíon de casos positivos y negativos y también permitiría identificar casos neutrales.

Finalmente, la minería de opiniones es un área de investigación relativamente nueva que hoy en día goza de alta popularidad. Es usada ya en múltiples sitios en la Web dado que existen intereses personales, empresariales y hasta gubernamentales por conocer lo que se opina sobre algún tema. Sin embargo, la minería de opiniones está lejos de ser resuelta, aún representa múltiples retos para el procesamiento del lenguaje natural y para la minería de textos.

Referencias

[1] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis," *Found. Trends Inf. Retr.*, vol. 2, pp. 1–135, January 2008. 1, 2

[2] 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*, ACL '02, (Stroudsburg, PA, USA), pp. 417–424, Association for Computational Linguistics, 2002. 3, 59, 69

[3] 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*, COLING '00, (Stroudsburg, PA, USA), pp. 299–305, Association for Computational Linguistics, 2000. 3, 59

[4] B. Liu, "Sentiment Analysis and Subjectivity," in *Handbook of Natural Language Processing, Second Edition* (N. Indurkhya and F. J. Damerau, eds.), Boca Raton, FL: CRC Press, Taylor and Francis Group, 2010. ISBN 978-1420085921. 3, 39

[5] H. Binali, V. Potdar, and C. Wu, "A state of the art opinion mining and its application domains," in *Proceedings of the 2009 IEEE International Conference on Industrial Technology*, (Washington, DC, USA), pp. 1–6, IEEE Computer Society, 2009. 3

[6] H. Tang, S. Tan, and X. Cheng, "A survey on sentiment detection of reviews," *Expert Syst. Appl.*, vol. 36, pp. 10760–10773, September 2009. 3

[7] B. Liu, M. Hu, and J. Cheng, "Opinion observer: analyzing and comparing opinions on the Web," in *Proceedings of the 14th international conference on World Wide Web*, WWW '05, (New York, NY, USA), pp. 342–351, ACM, 2005. 4

[8] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, (New York, NY, USA), pp. 168–177, ACM, 2004. 4

[9] S. M. Kim and E. Hovy, "Automatic Detection of Opinion Bearing Words and Sentences," in *Companion Volume to the Proceedings of IJCNLP-05, the Second International Joint Conference on Natural Language Processing*, (Jeju Island, KR), pp. 61–66, 2005. 4

[10] V. L. Rubin, J. M. Stanton, and E. D. Liddy, "Discerning Emotions in Texts," in *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*, (Stanford, US), 2004. 5

[11] C. Strapparava and R. Mihalcea, "Learning to identify emotions in text," in *Proceedings of the 2008 ACM symposium on Applied computing*, SAC '08, (New York, NY, USA), pp. 1556–1560, ACM, 2008. 5

[12] M. Jönsson, "Irony in online reviews: A linguistic approach to identifying irony," 2010. Gothenburg University Publications Electronic Archive. 5

[13] P. Carvalho, L. Sarmento, M. J. Silva, and E. de Oliveira, "Clues for detecting irony in user-generated contents: oh...!! it's "so easy";-)," in *Proceeding of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, TSA '09, (New York, NY, USA), pp. 53–56, ACM, 2009. 5

[14] T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," in *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, (Stroudsburg, PA, USA), pp. 347–354, Association for Computational Linguistics, 2005. 5

[15] E. Riloff and J. Wiebe, "Learning extraction patterns for subjective expressions," in *Proceedings of the 2003 conference on Empirical methods in natural language processing*, EMNLP '03, (Stroudsburg, PA, USA), pp. 105–112, Association for Computational Linguistics, 2003. 5

[16] E. Riloff, J. Wiebe, and T. Wilson, "Learning subjective nouns using extraction pattern bootstrapping," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, (Stroudsburg, PA, USA), pp. 25–32, Association for Computational Linguistics, 2003. 5

[17] H. Yu and V. Hatzivassiloglou, "Towards answering opinion questions: separating facts from opinions and identifying the polarity of opinion sentences," in *Proceedings of the 2003 conference on Empirical methods in natural language processing*, EMNLP '03, (Stroudsburg, PA, USA), pp. 129–136, Association for Computational Linguistics, 2003. 6

[18] 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*, ACL '98, (Stroudsburg, PA, USA), pp. 174–181, Association for Computational Linguistics, 1997. 6

[19] P. D. Turney and M. L. Littman, "Measuring praise and criticism: Inference of semantic orientation from association," *ACM Trans. Inf. Syst.*, vol. 21, pp. 315–346, October 2003. 6

[20] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005. 8

[21] S. Weiss, N. Indurkhya, T. Zhang, and F. Damerau, *Text Mining: Predictive Methods for Analyzing Unstructured Information*. Springer, 2004. 9, 12, 13, 15

[22] A. Srivastava and M. Sahami, eds., *Text Mining: Classification, Clustering, and Applications*. Chapman & Hall/CRC, 1st ed., 2009. 9, 10, 39

[23] D. Jurafsky and J. H. Martin, *Speech and Language Processing*. Prentice Hall, 2008. 11, 14, 16

[24] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st ed., 2009. 13, 42, 55

[25] C. D. Manning and H. Schütze, *Foundations of statistical natural language processing*. Cambridge, MA, USA: MIT Press, 1999. 13, 15

[26] R. Feldman and J. Sanger, *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2006. 16, 28, 30

[27] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of English: the penn treebank," *Comput. Linguist.*, vol. 19, pp. 313–330, June 1993. 16

[28] E. Alpaydin, *Introduction to Machine Learning*. The MIT Press, 2nd ed., 2010. 17, 18, 23

[29] X. Wu and V. Kumar, *The Top Ten Algorithms in Data Mining*. Chapman & Hall/CRC, 1st ed., 2009. 19

[30] B. Liu, *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. 19, 21, 22, 23, 26

[31] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008. 19

[32] N. Ye, *The Handbook of Data Mining (Human Factors and Ergonomics)*. Lawrence Erlbaum Associates, 2004. 22, 23

[33] F. Provost, T. Fawcett, and R. Kohavi, "The Case Against Accuracy Estimation for Comparing Induction Algorithms," in *Proceedings of the Fifteenth International Conference on Machine Learning*, 1997. 26

[34] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861 – 874, 2006. ROC Analysis in Pattern Recognition. 27

[35] "dendogram.png," Jan. 2011. Internet: `http://igraph.sourceforge.net/images/screenshots/dendrogram.png`. 31

[36] D. Skillicorn, *Understanding Complex Datasets: Data Mining with Matrix Decompositions*. Boca Raton, Florida: Chapman & Hall/CRC, 2007. 30, 33, 34

[37] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization.," *Nature*, vol. 401, pp. 788–791, Oct. 1999. 36

[38] P. O. Hoyer, "Non-negative sparse coding," *CoRR*, vol. cs.NE/0202009, 2002. 36

[39] V. Pauca, F. Shahnaz, M. W. Berry, and P. R. J., "Text Mining Using Non-Negative Matrix Factorizations," in *Proceedings of the Fourth SIAM International Conference on Data Mining*, pp. 452–457, 2004. 36, 75

[40] P. Paatero, "The multilinear engine – A table-driven, least squares program for solving multilinear problems, including the n-way parallel factor analysis model," in *Journal of Computational and Graphical Statistics*, pp. 854–888, 1999. 36

[41] C.-J. Lin, "Projected Gradient Methods for Nonnegative Matrix Factorization," *Neural Comput.*, vol. 19, pp. 2756–2779, October 2007. 36, 37, 38

[42] L. Chih-Jen, "Non-negative Matrix Factorization (NMF)," Jan. 2011. Internet: `http://www.csie.ntu.edu.tw/~cjlin/nmf/`. 38

[43] "About Python," Jan. 2011. Internet: `http://www.python.org/about/`. 42

[44] "Intro to Python and NLTK," Jan. 2011. Internet: `http://www.cs.oberlin.edu/~jdonalds/333/lecture03.html`. 55

[45] "What is a presupposition trigger?," Nov. 2010. Internet: `http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/WhatIsAPresuppositionTrigger.htm`. 60

[46] "What is a presupposition?," Nov. 2010. Internet: `http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/WhatIsAPresupposition.htm`. 60

[47] 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*, EMNLP '02, (Stroudsburg, PA, USA), pp. 79–86, Association for Computational Linguistics, 2002. 68, 69, 73, 75, 88

Apéndices

Apéndice A: Matrices de confusión y resultados de precisión, exhaustividad y medida F

En este apéndice se presentan seis tablas con otros resultados obtenidos.

En cada una de las primeras cinco (tablas 1, 2, 3, 4 y 5), se encuentra la matriz de confusión, con sus valores promediados de verdadero positivo (tp), falso negativo (fn), falso positivo (fp), verdadero negativo (tn) y los números de casos, para los experimentos realizados. La sexta y última tabla (tabla 6) presenta los resultados promediados de precisión, exhaustividad y medida F.

	Clasificado como positivo	Clasificado como negativo	Total
Realmente positivo	435.8	164.1	599.9
Realmente negativo	164.7	435.3	600
Total	600.5	599.4	1200

Tabla 1: Matriz de confusión con los valores promedio (promedio de la validación cruzada con 10 pliegues) usando enunciados con adjetivos o adverbios. El total, N, se redondeó.

	Clasificado como positivo	Clasificado como negativo	Total
Realmente positivo	395.5	204.9	600.4
Realmente negativo	195.5	404.1	599.6
Total	591	609	1200

Tabla 2: Matriz de confusión con los valores promedio (promedio de la validación cruzada con 10 pliegues) usando todos los enunciados. El total, N, se redondeó.

	Clasificado como positivo	Clasificado como negativo	Total
Realmente positivo	415.1	185.8	600.9
Realmente negativo	183.8	419.3	603.1
Total	598.9	605.1	1200

Tabla 3: Matriz de confusión con los valores promedio (promedio de la validación cruzada con 10 pliegues) usando los enunciados agrupados con FNM. El total, N, se redondeó.

	Clasificado como positivo	Clasificado como negativo	Total
Realmente positivo	398.7	201.3	600
Realmente negativo	188	412	600
Total	586.7	613.3	1200

Tabla 4: Matriz de confusión con los valores promedio (promedio de la validación cruzada con 10 pliegues) usando enunciados con adjetivos o adverbios o disparadores de presuposición. El total, N, se redondeó.

	Clasificado como positivo	Clasificado como negativo	Total
Realmente positivo	67.2	37	104.2
Realmente negativo	38.3	65.7	104
Total	105.5	102.7	208

Tabla 5: Matriz de confusión con los valores promedio (promedio de la validación cruzada con 10 pliegues) usando enunciados con disparadores de presuposición. El total, N, se redondeó.

	Método	Precisión	Exhaustividad	Medida F
(1)	enunciados con adjetivos o adverbios	72.57	72.65	72.61
(2)	enunciados agrupados con FNM	69.31	69.08	69.19
(3)	enunciados con disparadores o adjetivos o adverbios	67.96	66.45	67.19
(4)	todos los enunciados	66.92	65.87	66.39
(5)	enunciados con disparadores de presuposición	63.70	64.50	64.10

Tabla 6: Resultados promediados de precisión, exhaustividad y medida F de la validación con 10 pliegues del clasificador creado.

Apéndice B: Descripción de los módulos del sistema

En este apéndice se describen brevemente los módulos de Python que componen al sistema. También se presenta el diagrama con las relaciones existentes entre estos módulos. En total son 14 módulos:

1. `analizador_info_imdb.py`: módulo principal. Encargado de comenzar el proceso, desde la recopilación de comentarios y datos hasta la validación del clasificador,

2. `infoIMDB.py`: módulo que extrae los nombres de las películas desde Wikipedia. También adquiere los datos generales desde IMDb,

3. `ObtenComentarios.py`: módulo que contiene al Web crawler, consigue los comentarios de las películas encontradas en el módulo anterior,

4. `matriz_documento_termino.py`: módulo que genera la matriz de datos, con los enunciados segmentados de las comentarios, para la aplicación del algoritmo de FNM,

5. `clusternmf.py`: módulo que recibe la matriz de datos y se encarga, mediante los dos módulos siguientes, de aplicar FNM e interpretar los resultados para generar la agrupación,

6. `gpnmf.py`: módulo que aplica el algoritmo de FNM a la matriz de datos recibida,

7. `datos_entrenamiento.py`: módulo que interpreta las dos matrices resultantes de la aplicación de FNM y entrega los enunciados agrupados,

8. `enunciados_adjetivos_adverbios.py`: módulo que entrega una lista con las oraciones que contienen adjetivos o adverbios,

9. `oraciones_con_pres_triggers.py`: módulo que entrega una lista con las oraciones que contienen disparadores de presuposición,

10. `oraciones_con_triggers_y_adjetivos.py`: módulo que entrega una lista con las oraciones que contienen adjetivos o adverbios o disparadores de presuposición,

11. `oraciones_normales.py`: módulo que entrega una lista con todas las oraciones,

12. `validacion.py`: módulo que aplica la validación cruzada con k pliegues. Entrena y prueba el clasificador, mediante el siguiente módulo, y calcula las medidas de desempeño,

13. `bayes_ingenuo.py`: módulo que entrena el clasificador bayesiano ingenuo y lo prueba con los pliegues adecuados correspondientes,

14. `prueba_bopang.py`: módulo que lleva a cabo las pruebas con los datos encontrados en [47].

En la figura 1 se presentan las llamadas que hacen los módulos entre ellos.

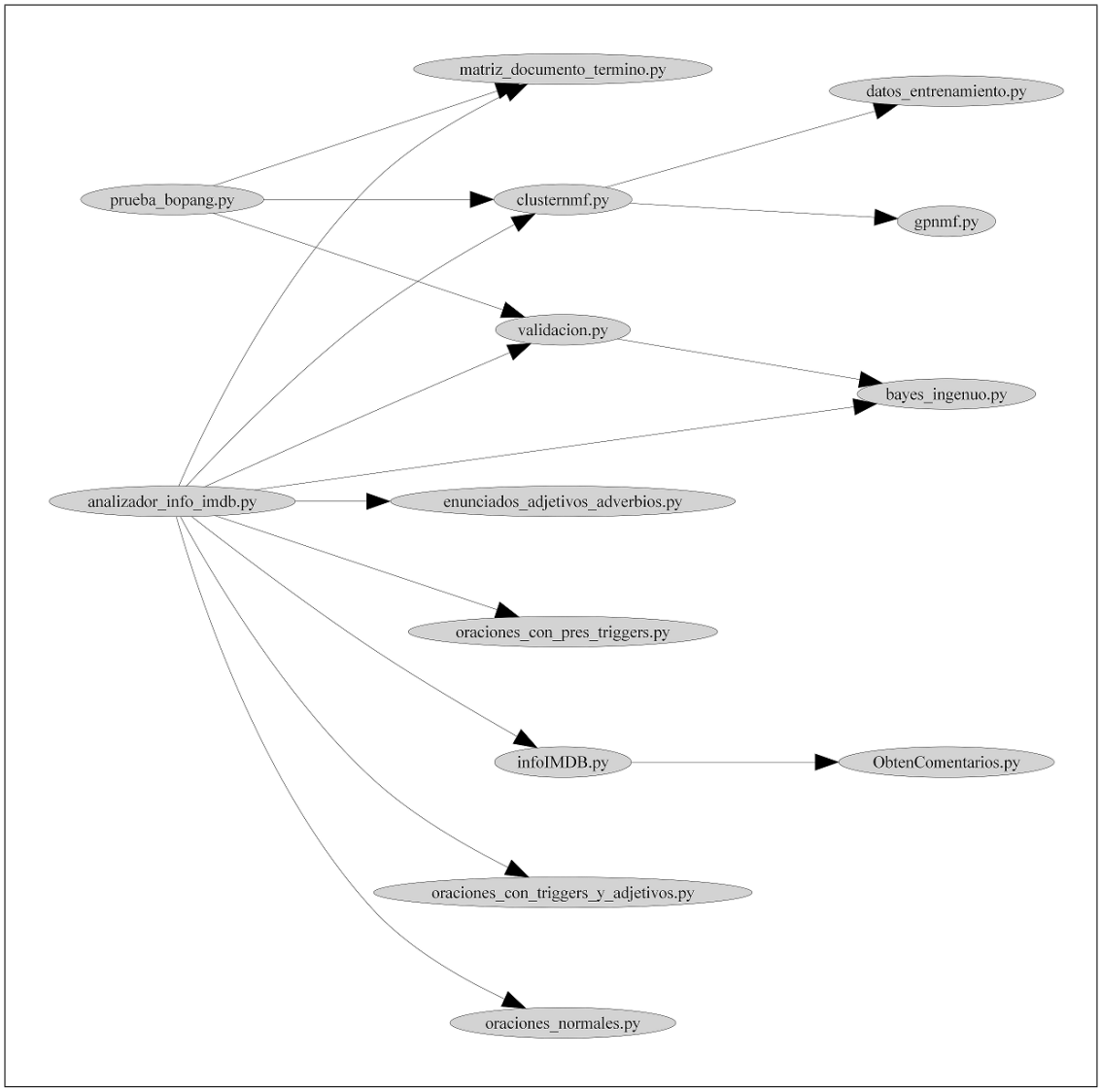

Figura 1: Relación de llamadas entre los módulos - La figura muestra las llamadas hechas entre cada uno de los módulos creados. Por ejemplo, el módulo `analizador_info_imdb.py` llama al módulo `infoIMDB.py`, y este a su vez llama al módulo `ObtenComentarios.py`

Apéndice C: Descripción del módulo de agrupamiento automático con FNM

El módulo `clusterFNM.py` contiene la función para agrupar automáticamente (por similitud) textos de acuerdo a una de dos tipos de métricas, por distancia Euclidiana o por el valor del coseno entre los documentos.

Parámetros de entrada:

- `archivo_documentos:` cadena con la ubicación (*path*) de un archivo de texto plano con un documento por línea.
- `r:` entero que indica el número de grupos deseados.
- `tol:` doble que indica la máxima diferencia entre la norma de la matriz de datos y la norma del producto de las dos matrices resultantes de la factorización.
- `timelimit:` entero que indica el límite de tiempo de ejecución, en segundos.
- `maxiter:` entero que indica el número máximo de iteraciones.

Regresa:

Una lista con `r` objetos tipo `Grupo`. Cada `Grupo` contiene los documentos que pertenecen a cada uno de los grupos: también incluye su nivel de pertenencia al grupo.

Guarda en disco:

- `clusterFNM.txt:` archivo de texto plano con los grupos encontrados y los documentos que pertenecen a cada uno de ellos.
- `matriz_datos.txt:` archivo de texto plano con la matriz de datos generada para el algoritmo de agrupamiento.

Apéndice D: Descripción del módulo de clasificación binaria mediante Bayes ingenuo

El módulo `clasificacionBayes.py` contiene la función para predecir dos tipos diferentes de clases de documentos mediante el algoritmo de Bayes ingenuo.

Parámetros de entrada:

- `archivo_documentos_clase1`: cadena con la ubicación (*path*) de un archivo de texto plano con un documento por línea. La primera línea del archivo deberá contener el nombre de la clase. Estos documentos pertenecen a la primera clase.
- `archivo_documentos_clase2`: cadena con la ubicación de un archivo de texto plano con un documento por línea. La primera línea del archivo deberá contener el nombre de la clase. Estos documentos pertenecen a la segunda clase.
- `archivo_documentos_a_clasificar`: cadena con la ubicación de un archivo de texto plano con un documento por línea. Estos documentos son los que se van a clasificar.
- `frecuencia_absoluta_minima`: entero que indica la frecuencia absoluta mínima con la que deben aparecer los tokens. Si no se indica, se usan todos los tokens.

Regresa:

Dos listas, la primera con los documentos de `archivo_documentos_a_clasificar` que pertenecen a la primera clase, la segunda con los documentos que pertenecen a la segunda clase.

Guarda en disco:

- `documentos_clase_`*nombre de la primera clase*`.txt`: archivo de texto plano con los documentos a los que se les asignó la primera clase.
- `documentos_clase_`*nombre de la segunda clase*`.txt`: archivo de texto plano con los documentos a los que se les asignó la segunda clase.