Universidad del Bío-Bío

Facultad de Ciencias Empresariales

Escuela de Ingeniería Civil Informática

Uso de técnicas de minería de opinión para apoyar la generación de recomendaciones de productos a partir de reseñas en idioma español

Proyecto de Título Presentado para Obtener el Título de Ingeniero Civil en Informática

Alumno: Carlos Lagos Urbina.

Profesor Guía: Pedro Campos Soto.

Concepción, Chile 2018

Agradecimientos

En primer lugar me gustaría agradecer a mis padres, por todo el esfuerzo y apoyo que han brindado para que yo esté culminando esta etapa de mi vida. A Natalia Rodríguez, por su infinita paciencia, comprensión y ayuda con este proyecto. A mi profesor guía Pedro Campos, por su trabajo, dedicación y orientación en este proyecto de título. A mis amigos, por siempre estar presentes en todo momento y finalmente agradezco el apoyo del proyecto "Contenido Generado por Usuarios: Información relevante para nuevas aplicaciones de Sistemas de Recomendación", código DIUBB 183515 3/R, financiado por la Dirección de Investigación de la Universidad del Bío-Bío.

Resumen

El Proyecto de Título presentado en este informe consiste en la aplicación de técnicas de minería de opinión para la extracción de preferencias de usuarios sobre diferentes de características o aspectos de ítems a partir de reseñas que permitan generar recomendaciones.

En primer lugar, con las herramientas Jsoup y Univocity Parsers se creó un dataset de reseñas de restaurantes en español, el cual se encuentra en formato de valores separados por tabulación.

Para la extracción de características de ítems y análisis de sentimiento de las reseñas, se utilizó un enfoque basado en aprendizaje automático supervisado. Para esto, se etiquetaron manualmente más de 800 oraciones presentes en el dataset con su respectiva polaridad (positivo, negativo o neutral) y aspecto (comida, servicio, precio o ambiente). A partir de las oraciones etiquetadas y utilizando la API de Weka se crearon dos modelos de clasificación, un modelo para determinar los aspectos del ítem de la oración y otro modelo para determinar el sentimiento de la oración. Para mejorar la precisión de los modelos de clasificación, se aplicaron técnicas de limpieza de datos. Los modelos de clasificación creados se utilizaron para extraer los aspectos y sentimiento de todas las oraciones presentes en las reseñas de los usuarios del dataset creado anteriormente.

Finalmente, la información de preferencias de aspectos de usuarios obtenida fue incorporada en un sistema de recomendación multi-criterio utilizando técnicas de filtrado colaborativo implementadas con Apache Mahout. La precisión de recomendación fue comparada con un sistema de recomendación tradicional. El mejor resultado lo obtuvo el sistema de recomendación multi-criterio creado a partir de reseñas.

Índice

1 Introducción y Definición del Problema

En este capítulo se presenta una introducción al tema y se define el problema a resolver.

1.1 Introducción

La creciente popularidad y difusión de redes sociales y de comercio electrónico animan a las personas a escribir reseñas describiendo su evaluación acerca de un ítem. Estas reseñas explican por qué al usuario le gustó o no un ítem basado en su experiencia personal. Dada la cantidad de datos que existe en la web, muchas aplicaciones podrían beneficiarse de esto.

Un ejemplo de esto, son los sistemas de recomendación. Estos sistemas ayudan a los usuarios a tratar con la sobrecarga de información y proporcionan recomendaciones personalizadas de ítems (Adomavicius & Tuzhilin, 2005), por ejemplo qué celular comprar, qué canción escuchar o en cuál restaurant comer. Los sistemas de recomendación tratan de predecir qué ítems son los más apropiados para un usuario basado en sus preferencias (por ejemplo a través de *ratings* o valoraciones otorgadas por el usuario) o acciones realizadas (por ejmeplo compras realizadas, canciones escuchadas, tiempo en una página, etc), de las cuales se deduce su preferencia. Tradicionalmente, los sistemas de recomendación utilizan un rating general por cada par (usuario, ítem) para representar el gusto del usuario por el ítem. En los últimos años este tipo de sistema ha sido considerado como limitado (Adomavicius & Kwon, 2015) dado que la preferencia de un usuario por un ítem puede depender de más de un aspecto de éste. Por ejemplo, al comer en un restaurant, un usuario pudo haber disfrutado de la comida y el ambiente, pero no del servicio.

Los sistemas de recomendación que modelan la preferencia de un usuario sobre un ítem con varios ratings se llaman sistemas de recomendación multi-criterio (Adomavicius & Kwon, 2015). A pesar de que este tipo de sistemas puede proporcionar recomendaciones más precisas (Adomavicius & Kwon, 2007), estos pueden abrumar a los usuarios al pedirle que otorgue muchos ratings por un mismo ítem. Es en este punto donde se debe aprovechar las reseñas escritas por los usuarios, analizándolas automáticamente y obteniendo su opinión o preferencia acerca de las diferentes característica de los ítems, para luego integrarlas en el proceso de generación de recomendaciones.

1.2 Definición del Problema

Uno de los problemas a los que se deben enfrentar los sistemas de recomendación es la escasez de ratings. Este problema ocurre cuando los ratings numéricos son difíciles de obtener por parte del sistema o simplemente los usuarios han otorgado ratings a una pequeña cantidad de ítems disponibles en el sistema. Debido a la escasa cantidad de ratings por usuario y por lo tanto la falta de información de las preferencias de un usuario, el sistema de recomendación será afectado negativamente en el proceso de generación de recomendaciones. Los sistemas de recomendación multi-criterio se ven más afectados por este problema, dado que los usuarios deben entregar más de un rating por ítem.

Para abordar este problema, en este proyecto de título se busca establecer un procedimiento para la extracción de información de reseñas en idioma español acerca de las preferencias de un usuario sobre diferentes aspectos de los ítems que ha consumido, para posteriormente incluir esta información en un sistema de recomendación multi-criterio.

1.3 Objetivos

A continuación, se presentan los objetivos generales y específicos de este proyecto de título.

1.3.1 Objetivo General

El objetivo general del proyecto es la aplicación de herramientas y técnicas de minería de opinión para extraer información de preferencias de usuario sobre diferentes aspectos de ítems a partir de reseñas en idioma español, que permitan apoyar la generación de recomendaciones para dichos ítems.

1.3.2 Objetivos Específicos

Los objetivos específicos del proyecto son los siguientes:

- Crear un dataset (recopilación de un conjunto de datos históricos) de comentarios en español con su respectivo rating sobre productos/servicios.
- Aplicar técnicas de minería de opinión para extraer características de ítems a partir de reseñas de usuarios en idioma español.

- Aplicar y evaluar técnicas de recomendación utilizando la información obtenida mediante el proceso de minería de opinión.
- Implementación de un software con interfaz gráfica de usuario que permita ajustar la ejecución de las técnicas implementadas.

1.4 Presentación de capítulos

En esta sección se presentan los capítulos que componen este proyecto de título.

Capítulo 1 Introducción y Definición del Problema: En este capítulo se presenta la introducción del trabajo, sus objetivos y la estructura de los capítulos.

Capítulo 2 Marco Teórico: En este capítulo se explica detalladamente las técnicas y procedimientos utilizados en este proyecto de título.

Capítulo 3 Herramientas Utilizadas: En este capítulo se presentan las herramientas utilizadas en el desarrollo de este proyecto de título.

Capítulo 4 Creación de Dataset: En este capítulo se presenta el proceso que se realizó para la creación del dataset.

Capítulo 5 Detalles de Implementación: En este capítulo se presentan los detalles de implementación de los pasos realizados en este proyecto de título.

Capítulo 6 Desarrollo de la Aplicación: En este capítulo se presentan los requerimientos del software y su diagrama de casos de uso.

Capítulo 7 Uso de la Aplicación: En este capítulo se presenta como se utiliza la aplicación desarrollada para aplicar diferentes técnicas de minería de opinión y de sistemas de recomendación.

Capítulo 8 Experimento: En este capítulo se presenta el experimento realizado, con el análisis de los resultados obtenidos.

Capítulo 9 Conclusiones: En este capítulo se presentan las conclusiones de este proyecto de título y el cumplimiento de los objetivos específicos.

2 Marco Teórico

En este capítulo se presenta una explicación acerca de minería de texto, análisis de sentimiento y sistemas de recomendación.

2.1 Minería de Texto

La minería de texto se puede definir como la extracción de información útil de fuentes de información a través de la identificación y exploración de patrones interesantes (Zhang, Chen, & Liu, 2015). Esta área de investigación abarca otras áreas como procesamiento del lenguaje natural, aprendizaje automático, recuperación de la información, entre otros. A diferencia de la minería de datos, en la cual la fuente de información son bases de datos con información estructurada, en la minería de texto la fuente de información son conjuntos de documentos de texto no estructurado.

La minería de texto ha sido influenciada por la minería de datos y por lo mismo ambas áreas poseen similitudes (Zhang et al., 2015). Por ejemplo, ambas áreas pueden utilizar métodos de pre procesamiento de datos, algoritmos para el descubrimiento de patrones y métodos para presentar estos patrones.

El proceso general para aplicar minería de texto se puede dividir en 4 fases (Feldman & Sanger, 2006), las cuales se explican a continuación:

- **Pre procesamiento:** Esta fase incluye todos los algoritmos y técnicas para preparar los datos antes de aplicar algoritmos para el descubrimiento de patrones. Generalmente, la fase de pre procesamiento convierte la representación del texto que se encuentra en el conjunto de documentos (Feldman & Sanger, 2006). En el punto 2.1.1 se explican algunas técnicas comúnmente utilizadas en esta fase.

- **Operaciones de minería:** Es la fase principal de la minería de texto la cual incluye la aplicación de algoritmos para el descubrimiento de patrones, análisis de tendencia, entre otros.

- **Capa de presentación:** Se pueden utilizar diversas herramientas para facilitar la exploración y análisis de la información extraída.

- **Post procesamiento:** En esta fase se aplican métodos para evaluar el proceso realizado y eliminar información redundante. Por ejemplo, si se han utilizado algoritmos de aprendizaje automático como arboles de decisión, puede que sea necesario realizar una poda.

Cabe destacar que las fases de pre procesamiento y operaciones de minería son consideradas como críticas dentro de todo el proceso de minería de texto (Feldman & Sanger, 2006).

2.1.1 Técnicas de Pre procesamiento de datos

La etapa de pre procesamiento tiene un rol importante en la minería de texto. Muchas veces el texto que se encuentra en línea trae consigo información no útil, por ejemplo, etiquetas HTML, palabras mal escritas o caracteres innecesarios. En esta etapa se aplican métodos y técnicas para transformar los datos con el fin de mejorar la efectividad del algoritmo clasificador. Algunas de las técnicas comúnmente utilizadas en esta etapa son:

- **Filtro de Stopwords:** Las stopwords o palabras vacías son palabras que no tienen un significado dentro de un texto como preposiciones, pronombres, artículos, entre otros.
- **Stemming:** Es un método para llevar las palabras a su forma raíz. Por ejemplo, niño, niña, niñez su raíz es “niñ”. Cabe destacar que la raíz de las palabras no es necesariamente una palabra válida.
- **Lematización:** Genera el lema de todas las palabras. Por ejemplo, la palabra juega, jugaba y jugando todas poseen el mismo lema “jugar”. En este caso el lema de las palabras si es un término válido.
- **Otros procesamientos:** Otras técnicas que puedan ser aplicadas para mejorar los resultados del clasificador. Por ejemplo, eliminación de signos, números, corrección de errores ortográficos, entre otros.

2.2 Clasificación de Texto

El problema de clasificación de texto se puede definir como: Dado un conjunto de categorías o clases y un conjunto de documentos de texto, se debe asignar la categoría correcta a cada documento (Feldman & Sanger, 2006). Hoy en día la clasificación automática de documentos es aplicada en varios dominios tales como filtrado de spam, análisis de sentimiento, clasificación de páginas web, entre otros.

Como muchos problemas, la clasificación de texto se puede realizar de diferentes formas (Feldman & Sanger, 2006). La primera es conocida como ingeniería del conocimiento, en la cual el conocimiento de expertos acerca de las categorías es codificada directamente en el sistema. La segunda forma es a través de la utilización de algoritmos de aprendizaje automático. En este enfoque se construye un clasificador que permitirá asignar la categoría a un nuevo documento de acuerdo a las características aprendidas de un conjunto de documentos que estén previamente clasificados en una categoría por un experto del dominio (Sebastiani & Fabrizio, 2002). En aprendizaje automático, el segundo enfoque se denomina aprendizaje supervisado dado que el entrenamiento del algoritmo es guiado por los datos previamente etiquetados de forma correcta.

Una vez que se ha construido un clasificador, es necesario evaluar su rendimiento. Como se ha mencionado anteriormente, para realizar clasificación de texto utilizando algoritmos de aprendizaje automático, es necesario un conjunto de documentos etiquetados con sus respectivas categorías. Para evaluar el rendimiento del clasificador, normalmente el conjunto de documentos es dividido en dos partes: un conjunto de entrenamiento y un conjunto de evaluación. El conjunto de entrenamiento es utilizado para entrenar al clasificador y el conjunto de evaluación es utilizado para evaluar su efectividad.

Las métricas comúnmente utilizadas para evaluar los clasificadores son *precision*, *recall* y *accuracy* (Feldman & Sanger, 2006), las cuales se calculan a partir de una matriz de confusión. Una matriz de confusión es una herramienta que permite la visualización del rendimiento de un algoritmo que emplea aprendizaje supervisado. En la Tabla 1 se presenta una matriz de confusión, donde:

- **TP** = verdadero positivo (true positives en inglés): número de muestras clasificadas como positivas que realmente son positivas.
- **FP** = falso positivo (false positives en inglés): número de muestras clasificadas como positivos y que realmente son negativas.
- **TN** = verdadero negativo (true negatives en inglés): número de muestras clasificadas como negativas y que realmente son negativas.
- **FN** = falso negativo (false negatives en inglés): número de muestras clasificadas como negativas y que realmente son positivas.

		Clasificación Automática	
		Positivo	Negativo
Clasificación Real	Positivo	TP	FN
	Negativo	FP	TN

Tabla 1: Matriz de confusión.

Las métricas son definidas a continuación.

- **Precision:** Porcentaje de los documentos correctamente clasificados en una categoría entre todos los documentos clasificados en la misma categoría.

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

Formula 1: Precision.

- **Recall:** Porcentaje de los documentos correctamente clasificados en una categoría en relación al número de documentos que pertenecen a esa categoría.

$$Recall = \frac{TP}{TP + FN}$$

Formula 2: Recall.

- **Accuracy:** Porcentaje de los documentos correctamente clasificados entre todas las clases.

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

Formula 3: Accuracy.

2.3 Análisis de Sentimiento

El análisis de sentimiento es el área que se encarga de analizar opiniones, sentimientos y emociones expresadas texto (Liu, 2012). En general, se puede hacer análisis de sentimiento en tres niveles de detalle.

- **Nivel de documento:** En este caso la tarea consiste en determinar si la opinión general de un documento es negativa o positiva. Por ejemplo, dada la reseña de un producto, se determina si la reseña completa expresa una opinión negativa o positiva sobre el producto.

- **Nivel de oración:** En este nivel se analizan las oraciones dentro de un documento para determinar si expresan una opinión negativa o positiva.

- **Nivel de aspecto:** El análisis en este nivel se basa en que una opinión consiste en un sentimiento (positivo o negativo) y un objetivo (Liu, 2012). En este caso se identifica qué le gustó y qué no le gustó a la persona. Por ejemplo, la frase *"La comida es muy sabrosa pero el servicio es muy lento"* evalúa dos aspectos del

producto, la comida y el servicio. El sentimiento expresado en el primer aspecto es positivo y en el segundo aspecto es negativo. Los aspectos son los objetivos de la opinión. Este problema es conocido como análisis de sentimiento basado en aspecto, donde se busca determinar los aspectos (características) de una entidad y su sentimiento.

2.3.1 Extracción de Aspecto

Explicado en el punto 2.3, el análisis de sentimiento a nivel de aspecto tiene como objetivo identificar la relación entre el sentimiento expresado y el aspecto en una opinión. La tarea de identificar los aspectos en una opinión se denomina extracción de aspecto. Un aspecto de una entidad representa una característica o atributo de ésta y debe ser identificado con un nombre único. Por ejemplo, para un celular se tienen los aspectos de pantalla, duración de batería, cámara, entre otros.

Existen dos tipos de aspectos que se pueden identificar, aspectos explícitos y aspectos implícitos (Liu, 2012). Los aspectos explícitos son palabras en el documento que explícitamente representan un aspecto. En el ejemplo presentado en el punto 2.3, comida y servicio están explícitamente en el texto. Por otro lado, los aspectos implícitos son aspectos que están presentes en el texto pero no explícitamente. Por ejemplo, la frase *"Nos atendieron de la mejor manera"* contiene una opinión implícita acerca del aspecto servicio.

Cabe destacar que Liu (Liu, 2012) diferencia entre categoría de un aspecto y expresión de un aspecto. La categoría de un aspecto está formada por varias expresiones del aspecto. Por ejemplo, menú, plato y postre son expresiones del aspecto y representan la categoría de aspecto comida.

2.4 Sistemas de Recomendación

Los sistemas de recomendación son herramientas software y técnicas que proporcionan recomendaciones personalizadas de diferentes ítems a usuarios (Ricci, Rokach, & Shapira, 2015). Un ítem es lo que el sistema recomiende, lo que significa que un ítem puede ser un objeto (computadores, libros, música) o un servicio (viajes, hoteles, restaurantes). Generalmente, estos sistemas se enfocan en un solo tipo de ítem a recomendar.

Los sistemas de recomendación ayudan al usuario a evitar la sobrecarga de información. Por ejemplo, si un usuario entra a la página web de Amazon para comprar un celular, se encontrará con más de 900.000 productos diferentes. Normalmente, solo una pequeña porción de los productos existentes son del interés del usuario y por lo tanto se enfrentará a la sobrecarga de información. El objetivo de los sistemas de recomendación es apoyar al usuario en el descubrimiento de contenido proporcionando recomendaciones de ítems que sean del interés del usuario.

2.4.1 Tipos de Sistemas de Recomendación

A continuación, se presenta una taxonomía de sistemas de recomendación propuesta por Burke (Burke, 2007). Esta taxonomía clasifica a los sistemas de recomendación en 5 diferentes tipos dependiendo de la técnica de recomendación utilizada.

2.4.1.1 Recomendación Basada en Contenido

Este tipo de sistema recomienda ítems que son similares a los vistos por el usuario anteriormente y que le han gustado. Esto quiere decir que el sistema genera recomendaciones de dos fuentes de datos, las características de los ítems y los ratings del usuario. Por ejemplo, si un usuario ha visto una película que pertenece al género de acción y le ha otorgado un rating positivo, el sistema va a recomendar películas del mismo género. En muchos casos, las características de los ítems son palabras claves extraídas de la descripción de los ítems (Ricci et al., 2015).

2.4.1.2 Filtrado Colaborativo

Filtrado colaborativo se puede realizar de dos formas diferentes, filtrado colaborativo basado en modelo y filtrado colaborativo basado en vecindario. En filtrado colaborativo basado en modelo sea crea un modelo predictivo utilizando ratings de los usuarios. Esto

quiere decir que dado los ratings o preferencias de un usuario, se hace la predicción de rating para un nuevo ítem. Para crear los modelos predictivos se utilizan diversos algoritmos tales como máquina de vectores de soporte, clustering o entropía máxima (Ning, Desrosiers, & Karypis, 2015). Por otro lado, filtrado colaborativo basado en vecindario recomienda ítems a un usuario basado en sus acciones pasadas (por ejemplo, ratings explícitos o compras de ítems) y las acciones pasadas de otros usuarios similares. La similitud entre dos usuarios es calculada en base a los ratings que ambos usuarios han otorgado a ítems en el pasado. Filtrado colaborativo es considerada la técnicas más popular y utilizada en los sistemas de recomendación (Ricci et al., 2015). En la sección 2.4.2 se presenta una explicación más detallada sobre filtrado colaborativo basado en vecindario.

2.4.1.3 Demográfico

Los sistemas de recomendación demográficos asumen que la recomendación de ítems es diferente para cada nicho demográfico. Esto quiere decir que, este tipo de sistema genera recomendaciones en base al perfil demográfico del usuario. Por ejemplo, se pueden personalizar las recomendaciones de acuerdo a la edad del usuario o el país en el que reside.

2.4.1.4 Recomendación Basada en Conocimiento

Los sistemas de recomendación basados en conocimiento, como dice su nombre, utilizan el conocimiento del dominio acerca de las preferencias de los usuarios para generar recomendaciones de ítems. Este tipo de sistemas son apropiados en el contexto de que los ítems no son adquiridos a menudo (Aggarwal, 2016). Algunos ejemplos de estos ítems son automóviles, paquetes de viajes, casas o seguros. Todos estos ítems son comprados pocas veces y por lo tanto es difícil obtener una cantidad suficiente de ratings para generar recomendaciones con técnicas como filtrado colaborativo.

2.4.1.5 Sistemas de Recomendación Híbridos

Las técnicas de recomendación mencionadas anteriormente utilizan distintas fuentes de información. Por ejemplo, filtrado colaborativo utiliza ratings de usuarios, recomendación basada en contenido utiliza la descripción de los ítems y recomendación demográfica utiliza el perfil demográfico del usuario para realizar recomendaciones. Debido a estas diferencias, es que cada una de estas técnicas posee sus ventajas y desventajas. En algunos casos, se

tiene la oportunidad de utilizar más de una técnica para la misma tarea de recomendación. A estos sistemas se les conoce como sistemas de recomendación híbridos, los cuales combinan técnicas de recomendación para lograr mejores recomendaciones.

2.4.2 Filtrado Colaborativo Basado en Vecindario

Mencionado en la sección 2.4.1.2, en filtrado colaborativo basado en vecindario (también conocidos como basados en memoria), se utiliza la información de ratings de usuarios pero a diferencia de filtrado colaborativo basado en modelo no se crea un modelo predictivo. En este caso los ratings de los usuarios son utilizados directamente en la predicción de rating para un nuevo ítem. Filtrado colaborativo basado en vecindario se puede hacer de dos formas:

1. **Filtrado colaborativo basado en usuario:** La idea principal de filtrado colaborativo basado usuario es que el rating de un nuevo ítem para un usuario va a ser similar al rating de otro usuario, si ambos usuarios son similares, es decir, si ambos usuarios han otorgado ratings similares a otros ítems.

2. **Filtrado colaborativo basado en ítem:** En este método se calcula la similitud entre el conjunto de ítems que el usuario ya ha otorgado un rating y el ítem a recomendar. Una vez encontrados y seleccionado los ítems más similares, se procede a calcular el rating.

Una diferencia a destacar entre filtrado colaborativo basado en usuario y filtrado colaborativo basado en ítem es que en el primer caso, los ratings se predicen utilizando los ratings de los usuarios del vecindario, es decir, se utilizan ratings de otros usuarios. En el segundo caso, los ratings de ítems se predicen utilizando los ratings del mismo usuario.

A pesar de que filtrado colaborativo se puede realizar de formas diferentes, en general esta técnica posee dos objetivos principales (Ning et al., 2015):

- **Predicción de Rating:** Uno de los objetivos de filtrado colaborativo es realizar una predicción de rating que un usuario daría a un nuevo ítem. Para esto, los ratings de los usuarios deben estar disponibles.

- **Determinación de top-k ítems:** En estos casos se tiene un conjunto de ítems con los que el usuario ha interactuado y por lo tanto el objetivo de recomendación es recomendar al usuario una lista de K ítems que pueden ser de su interés. En estos casos, la preferencia de un usuario sobre un ítem es representada con un valor binario (0 o 1).

De acuerdo a Buhmann (Buhmann et al., 2011), el proceso de generación de recomendaciones en filtrado colaborativo basado en vecindario puede ser resumido en los siguientes pasos. Los pasos que se explican a continuación corresponden al proceso de generación de recomendaciones de filtrado colaborativo basado en usuario, pero el proceso es análogo a filtrado colaborativo basado en ítem.

1. Asignar un valor a todos los usuarios de acuerdo a la similitud con el usuario activo.
2. Seleccionar al grupo de usuarios que poseen la mayor similitud respecto al usuario activo. El grupo de usuarios seleccionados se conoce como vecindario.
3. Realizar la predicción de rating para un ítem de acuerdo a los ratings de los usuarios del vecindario.

En el primer paso, para calcular la similitud entre usuarios o ítems se puede hacer de diferentes maneras. A continuación se mencionan medidas de similitud utilizadas en sistemas de recomendación:

- **Coeficiente de Correlación de Pearson:** Es una de las medidas de similitud más utilizadas en los sistemas de recomendación (Buhmann et al., 2011). Esta medida calcula el grado de relación lineal entre dos variables y es medida entre los valores de -1 y 1. Un coeficiente de correlación de Pearson de 1 indica el mayor grado de similitud, es decir, las variables están perfectamente correlacionadas, de manera proporcional. Por otro lado, el valor de -1 indica que existe una relación inversamente proporcional perfecta entre las variables. El valor 0 indica que no existe relación lineal entre las variables. En la Formula 4 se presenta como se calcula el coeficiente de correlación de Pearson, donde *"u"* y *"v"* representan dos usuarios diferentes, I_u y I_v son los ítems a los que los usuarios *"u"* y *"v"* han

otorgado un rating, p_{ui} es el rating del usuario *"u"* para el ítem *"i"* y $\bar{p}_u$ es el promedio de ratings otorgados por el usuario *"u"*.

$$sim(u,v) = \frac{\sum_{i \in I_u \cap I_v}(p_{ui} - \bar{p}_u)(p_{vi} - \bar{p}_v)}{\sqrt{\sum_{i \in I_u \cap I_v}(p_{ui} - \bar{p}_u)^2}\sqrt{\sum_{i \in I_u \cap I_v}(p_{vi} - \bar{p}_v)^2}}$$

Formula 4: Coeficiente de Correlación de Pearson para Usuarios.

Análogamente, la similitud entre dos ítems *"i"* y *"j"* se calcula de la siguiente manera:

$$sim(i,j) = \frac{\sum_{u \in U}(r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in U}(r_{ui} - \bar{r}_i)^2}\sqrt{\sum_{u \in U}(r_{uj} - \bar{r}_j)^2}}$$

Formula 5: Coeficiente de Correlación de Pearson para Ítems.

Donde U es el conjunto de usuarios que han entregado un rating para los ítems *"i"* y *"j"*, r_{ui} es el rating del usuario *"u"* para el ítem *"i"* y $\bar{r}_i$ es el promedio de ratings para el ítem *"i"* de todos los usuarios.

- **Distancia Euclidiana:** La distancia euclidiana entre dos vectores de dos dimensiones x= (x_1, x_2) e y=(y_1, y_2) se define como:

$$d(x,y) = \sqrt{(X_1 - Y_1)^2 + (X_2 - Y_2)^2} = \sqrt{\sum_{i=1}^{2}(X_i - Y_i)^2}$$

Formula 6: Distancia Euclidiana.

Donde x_i e y_i son los ratings otorgados por dos usuarios diferentes para los ítems que tienen en común. A diferencia del coeficiente de correlación de Pearson, en esta medida un mayor valor (distancia) entre los vectores señala menor similitud y una menor distancia representa una mayor similitud.

En el paso 2, para seleccionar al grupo de usuarios a utilizar en la predicción de rating se puede hacer de dos maneras:

- **K-vecinos más cercanos:** Se selecciona a los K usuarios más similares al usuario activo. El valor de K debe ser seleccionado de forma cuidadosa. Si K es muy grande, se tendrán problemas de eficiencia debido a la cantidad de memoria requerida para guardar los ratings de los usuarios (Ning et al., 2015). Por otro lado, si K es muy pequeño, el número de ítems a recomendar se puede ver reducido considerablemente (Ning et al., 2015).

- **Umbral de similitud:** En este caso se elige un umbral de similitud y se selecciona a todos los usuarios que posean una similitud igual o mayor a esta.

En el paso 3, una vez seleccionado el grupo de usuarios similares al usuario activo, se procede a realizar la predicción de rating. Esto se puede calcular como el promedio de ratings de los usuarios en el vecindario, pero este método no toma en cuenta los niveles de similitud que poseen los diferentes usuarios (Ning et al., 2015). Generalmente las predicciones de rating se calculan de la siguiente manera:

$$p_{u,i} = \bar{r}_u + \frac{\sum_{v \in V} sim_{u,v} \times (r_{v,i} - \bar{r}_v)}{\sum_{v \in V} |sim_{u,v}|}$$

Formula 7: Predicción de Rating.

Donde $p_{u,i}$ es la predicción de rating del usuario *"u"* para el ítem *"i"*, $sim_{u,v}$ es la similitud entre el usuario *"u"* y el usuario *"v"* y *V* es el vecindario seleccionado previamente.

2.4.2.1 Evaluación de Sistemas de Recomendación

Como se ha explicado anteriormente, los sistemas de recomendación tienen como objetivo predecir el rating que un usuario le daría a un ítem. Por lo tanto, una de las forma para evaluar un sistema de recomendación es medir la precisión de sus predicciones de ratings. Error cuadrático medio (Root Mean Squared Error - RMSE) y error absoluto medio (Mean Absolute Error - MAE) son dos de las métricas más utilizadas para evaluar la precisión de los sistemas de recomendación (Gunawardana & Shani, 2015). A continuación se presenta como calcular el error cuadrático medio y error absoluto medio entre los ratings que el sistema de recomendación calcula y los ratings reales.

- **Error cuadrático medio:**

$$RMSE = \sqrt{\frac{1}{|K|} \sum_{(u,i) \in K} (\hat{r}_{ui} - r_{ui})^2}$$

Formula 8: Error Cuadrático Medio.

- **Error absoluto medio:**

$$MAE = \frac{1}{|K|} \sum_{(u,i) \in K} |\hat{r}_{ui} - r_{ui}|$$

Formula 9: Error Absoluto Medio.

Donde *"u"* e *"i"* representan a un usuario y un ítem del sistema respectivamente, $\hat{r}_{ui}$ es el rating del ítem *"i"* que el sistema ha calculado para el usuario *"u"* y *K* es un conjunto de pares de (usuario, ítem) para los cuales el rating real que el usuario *"u"* le ha otorgado al ítem *"i"* es conocido y representado por r_{ui}.

2.5 Sistemas de Recomendación Multi-criterio

A pesar de que los sistemas de recomendación ayudan a los usuarios a encontrar ítems que sean de su interés a través de las recomendaciones, muchos de estos sistemas permiten al usuario entregar solo un rating general por ítem. Basándose en este rating general, es que los sistemas de recomendación generan sus recomendaciones. En los últimos años, este tipo de sistema ha sido considerado como limitado (Adomavicius & Kwon, 2015). Esto es debido a que, en el momento de que un usuario decide comprar un ítem, el usuario tendrá en consideración más de una característica (atributo o criterio) del ítem. Por ejemplo, en un sistema de recomendación multi-criterio de restaurantes donde se permite a los usuarios entregar un rating de 1 a 5 por cada atributo que el sistema permita (por ejemplo servicio y ambiente). Si un usuario entrega a un restaurant un rating de 1 y 5 respectivamente, el promedio de ambos ratings del usuario para el restaurant será de 3. Si otro usuario, para el mismo restaurant entrega los ratings de 5 y 1, su promedio también será de 3. En un sistema de recomendación tradicional estos usuarios se consideran similares, a pesar de que poseen diferentes gustos para los atributos de un mismo ítem. Es por esto que tener esta información acerca de los usuarios, ayudan a conocer de mejor manera sus preferencias y por lo tanto generar recomendaciones más precisas.

Los sistemas de recomendación tradicionales (de un rating) trabajan con solo un rating general por cada usuario e ítem. Una vez que el sistema posee suficientes ratings del usuario, va a proceder a estimar los ratings de ítems que el usuario aún no ha visto. El problema de recomendación asume que existe un conjunto de usuarios y un conjunto de ítems que pueden ser recomendados. Por lo tanto, formalmente la función que estima el rating de un ítem para un usuario se define como:

$$R: Usuario \times Ítem \rightarrow Rating$$

Formula 10: Función de Sistemas de Recomendación.

Para cada par de (usuario, ítem), donde *Usuario* e *Ítem* son los usuarios e ítems del sistema y Rating es un rating dentro de un cierto rango de valores. En los sistemas de recomendación multi-criterio, los usuarios entregan más de un rating por ítem. Por lo tanto, la función que estima el rating de un ítem para un usuario es definida como:

$$R: Usuario \times Ítem \rightarrow R_0 \times R_1 \times \ldots R_k$$

Formula 11: Función de Sistemas de Recomendación Multi-criterio.

Donde R_0 es el rating general y R_1 hasta R_k representan los ratings de los criterios sobre el ítem, donde al igual que el rating general, son ratings dentro de un rango de valores previamente definidos.

2.5.1 Incorporación de Ratings Multi-criterio en el Cálculo de Similitud para Filtrado Colaborativo

Los sistemas de recomendación multi-criterio proporcionan información adicional acerca de las preferencias de un usuario para un ítem. Esto quiere decir que, además de un rating general estos sistemas proporcionan un rating por cada aspecto o característica de un ítem. Por lo tanto, la incorporación de esta información en el proceso de generación de recomendaciones puede mejorar la experiencia del usuario proporcionando recomendaciones más precisas. El objetivo de los sistemas de recomendación multi-criterio es el mismo que el de los sistemas de recomendación tradicionales, es decir, buscan ítems que maximicen la utilidad del usuario (Adomavicius & Kwon, 2007). Por lo tanto, en los sistemas de recomendación multi-criterio también se busca predecir un rating general para cada ítem para cada usuario.

Como se puede observar en la Formula 4, en sistemas de recomendación multi-criterio no se puede emplear el Coeficiente de Correlación de Pearson dado que al momento de calcular la similitud entre dos usuarios, este considera que existe un rating por cada par (usuario, ítem). Para incorporar ratings multi-criterio en el cálculo de similitud entre usuarios, Adomavicius y Kwon (Adomavicius & Kwon, 2007) propusieron dos técnicas llamadas similitud promedio y similitud del peor caso. A continuación se presentan ambas técnicas.

- **Similitud promedio:**

$$sim_{prom}(u, v) = \frac{1}{K+1} \sum_{i=0}^{k} sim_i(u, v)$$

Formula 12: Similitud Promedio.

Donde *"u"* y *"v"* son dos usuarios del sistema y *k* es el número de criterios del ítem. En similitud promedio, la similitud entre dos usuarios se realiza de la siguiente manera: utilizando una técnica de similitud mencionada en la sección 2.4.2, se calcula la similitud de los criterios por separado para luego sumarlas y calcular el promedio. El valor obtenido es la similitud entre ambos usuarios.

- **Similitud del peor caso:**

$$sim_{min}(u, v) = min_{i=0\ldots k} sim_i(u, v)$$

Formula 13: Similitud del peor caso.

En esta técnica también se calcula la similitud de los criterios por separado pero en este caso, la similitud entre usuarios es la menor similitud calculada entre los criterios.

3 Herramientas Utilizadas

En este capítulo se presentan las herramientas software utilizadas en el desarrollo de este proyecto de título.

3.1 Jsoup

Jsoup es una librería de Java de código abierto para trabajar con documentos HTML. Esta librería proporciona una API para extraer y manipular datos de páginas web, utilizando métodos de CSS y JQuery. A pesar de que Jsoup trabaja con especificaciones de HTML5, esta librería está diseñada para tratar con diferentes versiones de HTML que se pueda encontrar en la web y automáticamente crear su respectivo DOM. Jsoup fue creado en el año 2009 por Jonathan Hedley y hasta el momento sigue actualizándose con nuevas mejoras.

En resumen, Jsoup proporciona las siguientes funcionalidades:

- Extraer y leer páginas HTML desde una URL, archivo o un string de texto.
- Encontrar y extraer datos de páginas web utilizando el DOM y métodos de CSS y JQuery.
- Manipular datos: agregar/modificar/eliminar elementos HTML, atributos o texto.

3.2 Univocity Parsers

Univocity Parsers es una librería Java de código abierto para leer y escribir archivos en formato CSV y TSV. Esta librería fue creada por Univocity Software y presenta una detallada documentación y varios tutoriales para facilitar su uso.

Las características que proporciona Univocity Parsers para escribir archivos son las siguientes:

- Escribir con selección de columnas.
- Escribir cabeceras y manejo de columnas vacías

Las características para leer archivos son las siguientes:

- Lectura de archivos concurrente.

- Extracción de cabeceras, límite de filas a leer, lectura de columnas específicas y manejo de columnas vacías.

3.3 WEKA

Weka es una colección de algoritmos de aprendizaje automático y herramientas para el pre procesamiento de datos desarrollado en la Universidad de Waikato en Nueva Zelanda (Witten & Frank, 2005). Esta herramienta incluye técnicas y algoritmos para los problemas de minería de datos de regresión, clasificación, clustering, reglas de asociación y selección de atributos (Witten & Frank, 2005).

Respecto a la clasificación, Weka proporciona diversos algoritmos tales como *support vector machines*, *naïve bayes*, árboles de decisión, entre otros. Cabe destacar que Weka proporciona soporte para todo el proceso de minería de datos, desde la preparación de los datos de entrada hasta la evaluación y visualización de los modelos generados.

Una característica de Weka es que los datos de entrada deben estar en formato ARFF (Attribute-Relation File Format). Los datos en este formato son archivos de texto que describen los datos y sus respectivos atributos. Un ejemplo de un archivo ARFF se presenta en la Figura 1.

```
@relation Categoria

@attribute sentece string
@attribute class {precio,comida,servicio,ambiente}

@data
'Con el descuento de Atrapalo relacion calidad/precio de 10, si no me pareceria caro.',precio
'Tanto el trato com Restaurante acogedor, con comida de calidad y buen servicio.',comida
'Tanto el trato como la comida, fueron de primera categoria, un lugar sin duda para repetir.',servicio
'Comida correcta pero un poco desangelado el local ya que habia muy pocas mesas.',comida
'Menu degustacion correcto, sin alardes.',comida
'A destacar el servicio, muy agradables y pendientes de los detalles.',servicio
'de postre coulant de chocolate con helado de mango tb muy rico.',comida
'Carta de vinos con varias referencias interesante.',comida
```

Figura 1: Ejemplo de archivo ARFF

Los archivos ARFF presentan dos secciones. La primera sección es la cabecera, en la cual se encuentran diferentes declaraciones. La declaración @relation se define en la primera línea del archivo y representa el nombre del conjunto de datos. Las declaraciones de @attribute identifica el nombre del atributo y el tipo de dato. Como se puede observar en la Figura 1, la primera declaración de @attribute identifica a un atributo llamado "sentence" y es del tipo "string". La segunda sección es la de datos, la cual contiene la declaración de @data. Esta declaración denota el inicio de los datos en el archivo.

Weka puede ser utilizado desde su interfaz gráfica, disponible de forma gratuita en su sitio web[1] o puede ser utilizado en código Java.

3.4 Standford CoreNLP

Standford CoreNLP es un conjunto de herramientas para el procesamiento de textos en idioma inglés, español, chino, alemán, entre otros. Standford CoreNLP puede ser utilizada a través de línea de comando, mediante una API o a través de su servicio web. Esta librería está escrita en Java pero puede ser utilizada en otros lenguajes tales como C#, Scala, Python, JavaScript, entre otros. Las herramientas que proporciona Standford CoreNLP son las siguientes:

- **Tokenizador:** Es el proceso de separar el texto en unidades llamadas tokens. Los tokens pueden ser palabras, números o conjunto de palabras.
- **Divisor de frase:** Separa una secuencia de tokens en frases. La separación de frases se puede realizar a través de espacios en blanco y/o puntuaciones.
- **Etiquetador gramatical (POS tagging en inglés):** Etiqueta tokens con su respectiva categoría gramatical dentro del texto.
- **Lemma:** Genera el lema (forma canónica o palabra representativa) de cada uno de los tokens.
- **Reconocedor de entidad (NER en inglés):** Reconoce entidades dentro de un texto. Alguna de las entidades que la herramienta reconoce por defecto son personas, ubicaciones, organizaciones, fechas, dinero, entre otras.

[1] www.cs.waikato.ac.nz/ml/weka/

- **RegexNER:** Es un reconocedor de entidades basado en patrones. A través de esta herramienta se pueden reconocer entidades que no están por defecto.
- **Parse:** Proporciona los analizadores sintácticos de circunscripción (constituent parser en inglés) y representación (representation parser en inglés).
- **Análisis de sentimiento:** Realiza análisis de sentimiento utilizando aprendizaje profundo.
- **Coref:** Esta herramienta encuentra referencias de la misma entidad dentro de un texto. Por ejemplo, si se habla de "Natalia" y luego se le menciona como "ella".

A pesar de que por defecto esta herramienta trabaja con textos en idioma inglés, también es posible utilizar esta herramienta en otros idiomas, incluyendo el español. En la Tabla 2 se presenta un resumen de las herramientas proporcionadas por Standford CoreNLP y su compatibilidad con otros idiomas.

Anotador	Árabe	Chino	Inglés	Español	Francés	Alemán
Tokenizador	X	X	X	X	X	
Divisor de frase	X	X	X	X	X	X
Etiquetador POS	X	X	X	X	X	X
Lemma			X			
NER		X	X	X		X
RegexNER	X	X	X	X	X	X
Parse	X	X	X	X	X	X
Análisis de sentimiento			X			
Coref			X			

Tabla 2: Resumen de herramientas e idiomas soportados por Standford CoreNLP (Manning et al., 2014)

3.5 Apache Mahout

Apache Mahout es una librería Java de aprendizaje automático de código abierto desarrollado por Apache Software Foundation. Esta librería es gratuita y puede ser descargada desde su sitio web[2].

A pesar de que muchas técnicas y algoritmos de esta librería aún están en desarrollo, Mahout principalmente proporciona soporte para realizar recomendaciones (filtrado colaborativo), clustering y clasificación (Owen, Anil, Dunning, & Friedman, 2012).

Con respecto a la generación de recomendaciones, Mahout proporciona diferentes métodos para el cálculo de similitud y creación de vecindarios explicados en la sección 2.2.2. Además, proporciona diferentes técnicas de recomendación tales como filtrado colaborativo basado en usuario o basado en ítem, recomendaciones Slope-One y descomposición en valores singulares (Singular Value Decomposition en inglés). Respecto a la evaluación de sistemas de recomendación, Mahout provee las métricas de error cuadrático medio y error absoluto medio.

Finalmente, cabe destacar que Apache Mahout es fácilmente extensible y ofrece una variedad de clases Java que pueden ser personalizadas.

[2] https://mahout.apache.org/

4 Creación de Dataset

En este capítulo se presenta el proceso que se realizó para la creación del dataset de comentarios en español.

4.1 Selección de Sitio Web

El primer paso de este proyecto de título fue la selección del ítem con el que se van a generar recomendaciones. Se decidió trabajar con restaurantes debido a la variedad de páginas web que poseen reseñas de este tipo y además es un ítem recurrente en el ámbito de los sistemas de recomendación. Una vez seleccionado el ítem, se comenzó la búsqueda de una página web de restaurantes que tuviese reseñas en español. Para seleccionar el sitio web, este debía cumplir con los siguientes requisitos:

Criterio 1: Alto número de restaurantes con reseñas en español.
Criterio 2: Las reseñas deben estar con su respectiva puntuación o rating.
Criterio 3: El sitio web debe permitir la extracción del contenido de sus páginas.

Los sitios web encontrados fueron los siguientes:

- **Yelp[3]:** Sitio web fundado en 2004 el cual tiene como objetivo conectar a la gente con diferentes tipos de comercios, los que incluyen restaurantes, bares, servicios de limpieza, entre otros. Hasta el año 2017, los usuarios han escrito más de 148 millones de reseñas en la página.

- **TripAdvisor[4]:** Sitio web que proporciona reseñas relacionadas con restaurantes, hoteles, paquetes de viajes y más contenido relacionado con viajes. Además de reseñas, a través de la página es posible realizar reservas de hoteles y vuelos. La página posee más de 500 millones de reseñas y comentarios.

- **Foursquare[5]:** Este sitio puede ser definido como una red social basada en servicios de localización que incorpora elementos de juego. La idea de este sitio es que a medida que los usuarios se encuentren en nuevos lugares, estos puedan marcarlos

[3] www.yelp.es/
[4] www.tripadvisor.es/
[5] www.es.foursquare.com/

para recibir recompensas. En este sitio se encuentran reseñas de diferentes servicios tales como vida nocturna, comida, compras, entre otros.

- **Atrápalo[6]:** Es una agencia de viajes la cual promociona desde actividades a realizar en las ciudades hasta paquetes de viajes y cruceros. Además, posee reseñas de hoteles y restaurantes.

Una vez encontradas los sitios web, se analizaron de acuerdo a los criterios definidos previamente. A continuación se presenta la tabla de resumen respecto a los criterios cumplidos por los sitios.

Sitio Web	Criterio 1	Criterio 2	Criterio 3
Yelp	X	X	
TripAdvisor	X	X	
Foursquare			
Atrápalo	X	X	X

Tabla 3: Aprobación de criterios

Como se puede observar en la Tabla 3, Foursquare fue descartada debido a que no cumplía con ninguno de los requisitos. Por otro lado, tanto Yelp como TripAdvisor cumplen con los requisitos de un alto número de reseñas en español con su respectiva puntuación, pero ambas páginas solo permiten la extracción de su contenido con la previa autorización de ellos. Debido a lo difícil de obtener el permiso y a la duración del proceso para obtenerlo, ambas páginas también fueron descartadas. Finalmente se decidió extraer el contenido del sitio web Atrápalo ya que posee un alto número de opiniones y restaurantes, las opiniones están vinculadas a una puntuación y el sitio web permite la explotación de su contenido, siempre que no sea utilizado para fines comerciales.

4.2 Extracción de Datos

Para la extracción de datos del sitio web Atrápalo, se realizaron dos programas en Java utilizando la librería Jsoup para la manipulación de páginas HTML y la librería Univocity parsers para dar un formato al archivo creado. Fue necesario realizar dos programas debido

[6] www.atrapalo.com/

a que el dataset obtenido del primer programa desarrollado sufría de escasez de datos. Este problema consiste en la incapacidad de encontrar vecinos en filtrado colaborativo debido a la poca cantidad de valoraciones que poseen los usuarios del sistema. Por lo tanto, para resolver este problema y obtener mayor cantidad de valoraciones, se desarrolló un segundo programa que extrae reseñas por usuario.

El primer programa funciona de la siguiente manera: En primer lugar el programa lee desde un archivo de texto las URL de los restaurantes seleccionados para la extracción de datos. Por cada restaurant puede existir más de una página de comentarios, por lo que es necesario iterar para recorrer las páginas de un mismo restaurant. Una vez en una página, se obtiene el contenido HTML de ésta y se extraen los comentarios con el respectivo identificador de usuario que lo realizó, fecha del comentario, identificador del restaurante y puntuación asociada. En la Figura 2 se presenta el pseudocódigo del primer programa.

```
INICIO
    Abrir archivo //archivo que contiene las URL de los restaurantes
        PARA cada URL del archivo
            Para 1 hasta N //el valor de N depende del número de páginas de comentarios
                Obtener el HTML de la URL
                Seleccionar ID_usuarios, ID_restaurante, comentarios, fechas y ratings de la página HTML
                Remover etiquetas HTML de los datos obtenidos
                Concatenar datos por ID_usuario
            FIN Para
            Guardar datos en archivo
        FIN Para
FIN
```

Figura 2: Pseudocódigo de primer programa

Una vez realizada la extracción de datos de una página, es necesario realizar una limpieza debido a que estos aún conservan las etiquetas HTML. Por lo tanto, se eliminan las etiquetas HTML de los datos y los datos son concatenados de acuerdo al identificador de usuario para que finalmente sean guardados en un archivo de texto en un formato de valores separados por tabulación (conocido como tab-separated values o TSV en inglés).

Respecto al segundo programa en Java creado para la extracción de datos, este funciona de forma similar al primer programa creado. En primer lugar lee un archivo que contiene identificadores de usuarios. Seguido de esto se crea una URL con el identificador para ingresar a la página del usuario con todos los comentarios realizados por este. Una vez en esta página, se extraen los comentarios realizados con su identificador de restaurante, fecha del comentario y puntuación asociada. A continuación se presenta el pseudocódigo del segundo programa creado.

```
INICIO
    Abrir archivo //archivo que contiene los ID de usuarios
    Crear URL con el ID de usuario
        PARA cada URL del archivo
            Obtener el HTML de su página
            Seleccionar ID_restaurante, comentarios, fechas y ratings de la página HTML
            Remover etiquetas HTML de los datos obtenidos
            Concatenar datos por comentario
        FIN Para
    Guardar datos en archivo
FIN
```

Figura 3: Pseudocódigo de segundo programa

Finalmente, el dataset creado posee 15 358 comentarios distribuidos entre 7 436 usuarios diferentes y 504 restaurantes. En la Tabla 4 se presenta un ejemplo del archivo obtenido. Los identificadores de usuario e identificadores de restaurantes fueron cambiados para mantener la privacidad de los usuarios del sitio web.

ID Usuario	ID Restaurante	Valoración	Fecha	Reseña
0	5355	4.0	19/12/2017	Buen restaurante con impecable servicio.
1	5355	5.0	18/12/2017	Excelente relación precio calidad.
2	5355	3.5	13/12/2017	Un lugar agradable.
3	5355	4.0	13/12/2017	Muy buena comida a buen precio.
4	5355	5.0	12/12/2017	Extraordinario el trato.

Tabla 4: Ejemplo de Dataset

5 Detalles de Implementación

En este capítulo se presentan los detalles de los pasos realizados en este proyecto de título.

5.1 Construcción de Conjunto de Entrenamiento

Como se ha mencionado anteriormente, el objetivo de este proyecto de título es la extracción de información de reseñas acerca de las preferencias de un usuario sobre diferentes aspectos de los ítems que ha consumido y posteriormente incluir esta información en un sistema de recomendación multi-criterio.

Por lo tanto, una vez creado el dataset de reseñas de restaurantes en idioma español, el problema a resolver es como extraer el aspecto y preferencia (sentimiento) de una reseña escrita por un usuario. Este problema se conoce como análisis de sentimiento basado en aspecto. Para resolver ambas tareas, se decidió utilizar algoritmos de clasificación supervisada. Explicado en la sección 2.2, estos algoritmos necesitan un conjunto de documentos etiquetados para clasificar nuevos documentos.

Para la construcción de ambos conjuntos de entrenamiento (Para la extracción de aspecto y análisis de sentimiento), se extrajeron las reseñas del dataset creado anteriormente y con la ayuda del divisor de frase de Standford CoreNLP, se separaron las reseñas por oraciones. En la Figura 4 se muestra el método desarrollado para separar oraciones. En primer lugar en el método se crea un objeto de Standford CoreNLP y se configura para el idioma español el tokenizador y el divisor de frase. Luego se recorre una lista de strings donde se encuentran almacenadas las reseñas, y para cada reseña se aplica el divisor de frase. Las oraciones obtenidas de una reseña son almacenadas en una lista, por lo que es necesario recorrer esta lista para guardar cada oración en otra lista donde se almacenan todas las oraciones. Una vez que ya no existen reseñas por separar, las oraciones son guardadas en un archivo para su etiquetado.

```
private static List<Object[]> SentenceSplit(List<String[]> reseñas) {
    List<Object[]> datos = new ArrayList<>();
    Properties props = new Properties();
    props.put("annotators", "tokenize, ssplit");
    props.setProperty("tokenize.language", "es");
    StanfordCoreNLP pipeline = new StanfordCoreNLP(props);
    for (String[] strings : reseñas) {
        Annotation document = new Annotation(strings[4]);
        pipeline.annotate(document);
        List<CoreMap> sentences = document
           .get(CoreAnnotations.SentencesAnnotation.class);
        for (CoreMap sentence : sentences) {
            String[] oracion = {sentence.toString()};
            datos.add(oracion);
        }
    }
    return datos;
}
```

Figura 4: Método para separar oraciones.

5.1.1 Proceso de Etiquetado

Por cada oración, los etiquetadores entregaban dos tipos de información: el aspecto presente en la oración y su sentimiento (positivo, negativo o neutral). Los aspectos a considerar fueron previamente determinados de acuerdo a trabajos desarrollados en este dominio. Por lo tanto, en base a lo señalado en los artículos de Musto (Musto, de Gemmis, Semeraro, & Lops, 2017), Adomavicius (Adomavicius & Kwon, 2015) y Chen (Chen, Chen, & Wang, 2015) los aspectos considerados son Comida, Precio, Ambiente y Servicio.

Las oraciones fueron clasificadas por dos etiquetadores humanos (Etiquetador A y Etiquetador B). En primer lugar, cada etiquetador clasificó un número determinado de oraciones por sí solo. Luego, las oraciones clasificadas por el Etiquetador A fueron examinadas y validadas por el Etiquetador B y viceversa. En caso de existir un desacuerdo entre los etiquetadores, la oración era eliminada del conjunto de entrenamiento.

La mayoría de los desacuerdos fue por las siguientes razones:

- **Ambigüedad del sentimiento expresado:** En algunos casos es poco claro si la oración expresa un sentimiento positivo, negativo o si expresa un hecho (neutral). Por ejemplo, en la oración *"Y el postre de locura"* no es claro si manifiesta una opinión negativa o positiva respecto al aspecto de comida.

- **Múltiples aspectos:** En muchos casos, la oración expresaba opinión respecto a múltiples aspectos del ítem. Por ejemplo, la oración *"La comida esta buena y el trato es agradable"* expresa un sentimiento positivo hacia los aspectos de comida y servicio. En estos casos, los etiquetadores clasificaban la oración en un aspecto.

Finalmente, el balance de los conjuntos de entrenamiento creados se presenta en la Tabla 5 y Tabla 6.

Aspecto	**Cantidad**
Comida	385
Servicio	246
Ambiente	187
Precio	86
Total	904

Tabla 5: Balance de Aspecto.

Polaridad	**Cantidad**
Positivo	738
Negativo	88
Neutral	152
Total	978

Tabla 6: Balance de Polaridad.

Ambos conjuntos creados están disponibles por separado en archivos de texto en formato de valores separados por tabulación. Esto quiere decir que, en la primera columna de los archivos se encuentra la oración y en la segunda columna su respectiva clasificación. En las tablas Tabla 7 y Tabla 8 se presenta un ejemplo de ambos conjuntos.

Oración	**Etiqueta**
Con el descuento de Atrápalo relación calidad/precio de 10 si no me parecería caro.	Neutral
Tanto el trato como Restaurante acogedor, con comida de calidad y buen servicio.	Positivo
Mal servicio, comida fría y nada apetecible.	Negativo

Tabla 7: Conjunto de entrenamiento de sentimiento.

Oración	**Etiqueta**
Con el descuento de Atrápalo relación calidad/precio de 10 si no me parecería caro.	Precio
Tanto el trato como Restaurante acogedor, con comida de calidad y buen servicio.	Comida
Buena comida y servicio fantástico	Servicio

Tabla 8: Conjunto de entrenamiento de aspecto.

5.2 Etapa de Pre procesamiento

Como se ha explicado anteriormente, para mejorar los resultados del algoritmo clasificador se pueden aplicar diferentes técnicas. Las técnicas de limpieza de datos que fueron aplicadas son eliminación de stopwords y eliminación de caracteres (tildes, puntuación, signos, etc.). La limpieza se realizó de la siguiente manera: se realiza la lectura de uno de los archivos en formato TSV, donde se encuentran las oraciones y su etiqueta (polaridad o aspecto dependiendo del archivo). Ubicándose en la primera columna del archivo, se comienza a realizar la lectura por fila (oración) y se aplican los métodos. Las técnicas se aplicaron a ambos conjuntos de entrenamiento. Los métodos desarrollados para aplicar las técnicas anteriormente mencionadas se presentan a continuación:

- **stopWords:** Realiza la eliminación de stopwords de una oración. Las stopwords son leídas desde un archivo y agregadas a una lista. Luego, para la oración que recibe como parámetro se recorre la lista para eliminar las stopwords de la oración. La lista de stopwords utilizada está disponible en Github[7].

```
private static String stopWords(String oracion, File archivo) {
    String frase = " " + oracion + " ";
    frase = frase.toLowerCase();
    try {
        BufferedReader br = new BufferedReader(new FileReader(archivo));
        List<String> stop = new ArrayList<>();
        String linea;
        while ((linea = br.readLine()) != null) {
            stop.add(linea);
        }
        for (String string : stop) {
            frase = frase.replaceAll(" " + string + " ", " ");
        }
    } catch (IOException ex) {
        System.out.println(ex);
    }
    return frase;
}
```

Figura 5: Método para eliminar stopwords.

- **noCaracteres:** Permite la eliminación de los caracteres que no sean palabras de una oración. Además, reemplaza la letra “ñ” por “gn”.

[7] https://github.com/stopwords-iso/stopwords-es

```
private static String noCaracteres(String oracion) {
    String numeros = oracion.replaceAll("\\d", "")
            .replaceAll("[[^\\w]&&[^\\s]]", "")
            .replaceAll("ñ", "gn");
    return numeros;
}
```

Figura 6: Método para eliminar caracteres.

- **noTildes:** Elimina tildes de una oración. Por ejemplo, la palabra "oración" es reemplazada por "oracion".

```
private static String noTildes(String oracion) {
    String sinTilde = StringUtils.stripAccents(oracion);
    return sinTilde;
}
```

Figura 7: Método para eliminar tildes.

Una vez aplicados todos los métodos, las oraciones son guardadas en un archivo de texto en formato TSV, conservando su estructura, es decir, en la primera columna se encuentra la oración y en la segunda columna su etiqueta. En este punto se obtienen dos archivos con oraciones pre procesadas y etiquetadas en formato TSV.

A modo de ejemplo, en la Tabla 9 se presentan oraciones y su respectiva transformación después de aplicar todos los métodos anteriormente mencionados.

Oración	Transformación
“A destacar el servicio, muy agradables y pendientes de los detalles.”	“destacar servicio agradables pendientes detalles”
“Platos deliciosos, con una mezcla de sabores apasionante y un ambiente único.”	“platos deliciosos mezcla sabores apasionante ambiente unico”
“Además el servicio fue excelente, nos atendieron varios camareros y todos fueron super atentos.”	“servicio excelente atendieron camareros super atentos”

Tabla 9: Transformación de oraciones.

5.3 Aplicación de Algoritmo Clasificador

La aplicación del algoritmo clasificador se realizó con la ayuda de la herramienta Weka. Como se explicó en la sección 3.3, esta herramienta trabaja con archivos en formato ARFF. Por lo tanto, en primer lugar se procedió a realizar la transformación de los archivos en formato TSV a ARFF. En la Figura 8 se presenta el método creado para la transformación de los archivos.

```
private static void TSVtoArff(String arff, String txt) {
    File archivo = new File(arff);
    CSVLoader loader = new CSVLoader();
    loader.setFieldSeparator("\t");
    try {
        loader.setSource(new File(txt));
        Instances data = loader.getDataSet();
        ArffSaver saver = new ArffSaver();
        saver.setInstances(data);
        saver.setFile(archivo);
        saver.writeBatch();
    } catch (IOException ex) {
        System.out.println(ex);
    }
}
```

Figura 8: Método para transformar archivos TSV a ARFF.

Este método recibe como parámetro dos string de texto. El primer parámetro representa el nombre del archivo TSV a convertir y el segundo parámetro representa el nombre del archivo ARFF a crear. El método fue desarrollado utilizando las clases de CSVLoader y ArffSaver proporcionadas por Weka. CSVLoader es una clase que permite la lectura de archivos en formato de valores separados por comas (comma separated values o CSV en inglés), pero el separador de columnas puede ser cambiado para trabajar con otros formatos. ArffSaver es una clase que escribe datos en un archivo en formato ARFF. Una vez transformados ambos archivos en formato ARFF, se procede a aplicar un algoritmo clasificador.

Para aplicar el algoritmo clasificador se desarrolló el método presentado en la Figura 9 con la API de Weka. En primer lugar este método lee un archivo que es recibido como parámetro. Luego se crea la clase Instances la cual permite la manipulación de los datos contenidos en el archivo.

```
private static void entrenarModelo(String arff, String nombre) {
    try {
        BufferedReader dataset = new BufferedReader(new FileReader(arff));
        Instances trainInstance = new Instances(dataset);
        trainInstance.setClassIndex(trainInstance.numAttributes() - 1);
        FilteredClassifier fc = new FilteredClassifier();
        StringToWordVector filter = new StringToWordVector();
        filter.setInputFormat(trainInstance);
        filter.setLowerCaseTokens(true);
        fc.setFilter(filter);
        fc.setClassifier(new SMO());
        fc.buildClassifier(trainInstance);
        Evaluation eval = new Evaluation(trainInstance);
        eval.crossValidateModel(fc, trainInstance, 10, new Random(1));
        System.out.println(eval.toSummaryString("\nEvaluación\n=======\n", false));
        SerializationHelper.write(nombre + ".model", fc);
        dataset.close();
    } catch (Exception ex) {
        System.out.println(ex);
    }
}
```

Figura 9: Método para entrenar modelo.

Seguido de esto, se crea la clase FilteredClassifier que permite la configuración de diferentes filtros antes de aplicar el algoritmo clasificador y se crea el filtro StringToWordVector para la representación de las oraciones como bolsa de palabras. El método finaliza realizando una validación cruzada para la evaluación del modelo y guarda el modelo creado con el nombre que recibe como segundo parámetro. El algoritmo clasificador seleccionado fue *support vector machines* dado que obtuvo el mejor rendimiento en el trabajo de Elgueta (Elgueta, 2017) y está disponible en Weka. Cabe destacar que se generaron dos modelos a partir de los conjuntos de entrenamiento creados anteriormente.

5.4 Evaluación de Algoritmo Clasificador

Respecto a la evaluación de los modelos creados, se obtuvieron los resultados presentados a continuación. En la Tabla 10 se presenta la matriz de confusión del modelo de polaridad.

	Positivo	**Negativo**	**Neutral**
Positivo	691	6	41
Negativo	46	29	13
Neutral	115	6	31

Tabla 10: Matriz de confusión de polaridad.

A partir de la matriz de confusión presentada, se obtuvo el *accuracy* del modelo:

- **Accuracy:**

$$Accuracy = \frac{691 + 29 + 31}{978} = 0{,}76789366\ x\ 100 = \ 76{,}789\ \%$$

Formula 14: Valor de accuracy de modelo de polaridad.

Respecto al modelo de aspecto, en la Tabla 11 se presenta su matriz de confusión.

	Precio	**Comida**	**Servicio**	**Ambiente**
Precio	70	14	2	0
Comida	15	325	25	20
Servicio	4	34	198	10
Ambiente	1	34	15	137

Tabla 11: Matriz de confusión de aspecto.

Al igual que con el modelo anterior, se calculó su accuracy.

- **Accuracy:**

$$Accuracy = \frac{70 + 325 + 198 + 137}{904} = 0{,}807522\ x\ 100 = 80{,}752\ \%$$

Formula 15: Valor de accuracy de modelo de aspecto.

Como se puede observar, ambos modelos creados poseen una exactitud mayor al 75%, por lo que se procede a utilizar los modelos para identificar el aspecto y polaridad de las opiniones extraídas en el punto 4.

5.5 Extracción de Aspecto y Análisis de Sentimiento

Para realizar el análisis de sentimiento basado en aspecto se desarrolló el método presentado en las Figura 10 y Figura 11. Antes de comenzar a clasificar nuevos documentos, es necesario que el modelo utilizado conozca los atributos y las posibles clases de los documentos. Es por esto que en primer lugar se abren los archivos ARFF creados en el punto 5.3 y se copia su información. Para copiar la información se utiliza la clase *Instance* de Weka, la cual representa una instancia del conjunto de datos y además mantiene internamente los atributos y las posibles clases del mismo. Por lo tanto, se crearon

los objetos *clsAspecto* y *clsPolaridad*, los cuales mantienen la información de sus respectivos conjuntos de datos.

Luego se realiza la apertura y lectura del dataset que contiene las opiniones de restaurantes y se configura el divisor de frase de Standford CoreNLP para comenzar a clasificar nuevas oraciones.

```
private static void usarModelo(String archivo, String dataset) {
    List<Object[]> datos = new ArrayList<>();
    int criterio = 0;
    double preferencia = 0;
    try {
        Classifier clsA = (Classifier) SerializationHelper.read("aspecto.model");
        Classifier clsP = (Classifier) SerializationHelper.read("polaridad.model");
        BufferedReader readerP = new BufferedReader(new FileReader("Polaridad/Polaridad_preprocess.arff"));
        BufferedReader readerA = new BufferedReader(new FileReader("Aspecto/Aspecto_preprocess.arff"));
        Instances aspecto = new Instances(readerA);
        Instances polaridad = new Instances(readerP);
        aspecto.setClassIndex(aspecto.numAttributes() - 1);
        polaridad.setClassIndex(polaridad.numAttributes() - 1);
        Instance clsAspecto = (Instance) aspecto.instance(0).copy();
        Instance clsPolaridad = (Instance) polaridad.instance(0).copy();
        File archivoLeer = new File(dataset);
        File multi = new File(archivo);
        List<String[]> s = TSV.leerArchivo(archivoLeer);
        Properties props = new Properties();
        props.put("annotators", "tokenize, ssplit");
        props.setProperty("tokenize.language", "es");
        StanfordCoreNLP pipeline = new StanfordCoreNLP(props);
```

Figura 10: Método para usar modelo parte 1.

Para clasificar las oraciones es necesario recorrer todo el dataset y luego, para cada reseña presente en el archivo, aplicar el divisor de frase. Cada frase se guarda en los objetos *cslAspecto* y *clsPolaridad* debido a que poseen la información que los modelos necesitan para clasificarlos. Finalmente, se utilizan los modelos creados para clasificar cada oración de acuerdo a la polaridad y aspecto. En la Figura 11 se presenta lo explicado anteriormente.

```
for (String[] strings : reseñas) {
    String[] data = {strings[0], strings[1], strings[2], null, null, null, null};
    Annotation document = new Annotation(strings[4]);
    pipeline.annotate(document);
    List<CoreMap> sentences = document.get(SentencesAnnotation.class);
    for (CoreMap sentence : sentences) {
        String oracion = sentence.toString();
        clsAspecto.setValue(0, oracion);
        clsPolaridad.setValue(0, oracion);
        double predicted = clsA.classifyInstance(clsAspecto);
        double predicted1 = clsP.classifyInstance(clsPolaridad);
```

Figura 11: Método para usar modelo parte 2.

Cabe destacar que para la incorporación de información de preferencias de características de un ítem en un sistema de recomendación, se consideró el sentimiento extraído por el modelo creado anteriormente como las calificaciones o ratings expresadas por el usuario. Por lo tanto, el sentimiento expresado en la oración se interpreta de la siguiente forma:

- **Positivo:** En caso de que la oración exprese un sentimiento positivo, el rating del usuario respecto al aspecto considerado es de 5.0.

- **Negativo:** En caso de que la oración exprese un sentimiento negativo, el rating del usuario respecto al aspecto considerado es de 1.0.

- **Neutral:** En caso de que la oración no exprese ninguna opinión, el rating del usuario respecto al aspecto considerado es de 3.0.

Una vez clasificadas todas las oraciones del dataset, se obtiene un nuevo archivo de texto en formato TSV donde en la primera columna se encuentra el identificador de usuario, en la segunda columna el identificador de restaurant y en las siguientes 4 columnas se encuentran el rating general del ítem y los ratings de los aspectos considerados del mismo.

5.6 Generación de Recomendaciones Tradicional

La generación de recomendaciones de un criterio y multi criterio se realizó con la librería de Apache Mahout. Explicado en el punto 2.4.2, para generar recomendaciones en filtrado colaborativo es necesario poseer la información de interacción de los usuarios con los ítems. Mahout mantiene esta información como (usuario, ítem, valoración) en una clase llamada *Preference*. *PreferenceArray* mantiene un conjunto de preferencia de un usuario o ítem dependiendo de su uso. Por razones de eficiencia de memoria, Mahout solo permite que los identificadores de usuario e ítem sean numéricos.

El conjunto de datos que contiene toda la información de la interacción de los usuarios con los ítems es representado por la clase *DataModel*. Mahout proporciona diferentes implementaciones para cargar datos de diferentes fuentes tales como bases de datos o archivos. La clase para cargar archivos es *FileDataModel,* la cual espera que el archivo contenga en cada línea un identificador de usuario, un identificador de ítem, seguido por un valor de preferencia del usuario sobre el ítem. El archivo puede estar en formato TSV o CSV.

El archivo creado en la sección 5.5 posee esta estructura pero además contiene la preferencia en cada uno de los aspectos considerados, por lo que fue necesario separar la información en 5 archivos diferentes. Los archivos mantienen la estructura de (usuario, ítem, valoración) pero cada uno contiene las valoraciones de los diferentes aspectos del ítem.

Una vez que se tienen los archivos, se procede a realizar la generación de recomendaciones. Mencionado anteriormente, para cargar un conjunto de datos se utiliza la clase *FileDataModel,* como se muestra en la Figura 12.

```
DataModel model = new FileDataModel(new File("ProyectoTitulo/general.txt"));
```

Figura 12: Cargando datos desde archivo.

Seguido de esto, se procede a crear un motor de recomendación. La técnica de recomendación que se utilizará es filtrado colaborativo basado en usuario. Para facilitar la comprensión del funcionamiento de esta técnica en Mahout, se presenta la Figura 13. *UserSimilarity* define la medida de similitud utilizada entre usuarios. *UserNeighborhood* define el grupo de usuarios más similares a un usuario determinado. Finalmente, *Recommender* junta todos estos componentes para realizar recomendaciones. Mahout proporciona diferentes motores de recomendaciones dependiendo de la técnica de que se necesite usar, por lo que no todos los motores de recomendación tendrán los mismos componentes e interacciones.

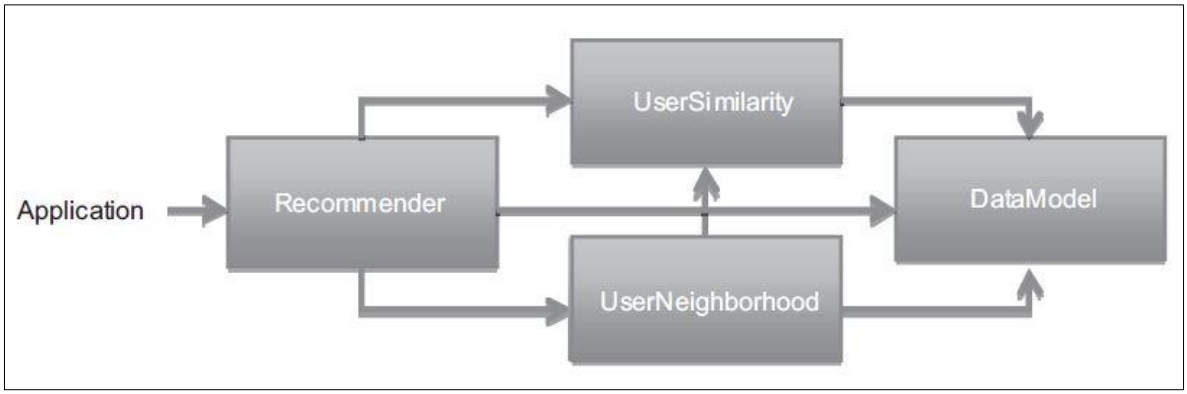

Figura 13: Interacción de los componentes de un motor de recomendación basada en usuario en Mahout (Owen et al., 2012).

Para realizar filtrado colaborativo basado en usuario y posteriormente evaluarlo, se crea un objeto *RecommenderBuilder*. Este objeto crea un recomendador en base a los diferentes parámetros que se le entreguen. En la Figura 14 se presenta la creación del objeto *RecommenderBuilder*. Como se puede observar, la medida de similitud utilizada es el Coeficiente de Correlación de Pearson y para la creación del vecindario se utilizan los K vecinos más cercanos, con un valor de K de 10.

```
RecommenderBuilder tradicional = new RecommenderBuilder() {
    @Override
    public Recommender buildRecommender(DataModel model) throws TasteException {
        UserSimilarity sim = new PearsonCorrelationSimilarity(model);
        UserNeighborhood neighborhood = new NearestNUserNeighborhood(10, sim, model);
        return new GenericUserBasedRecommender(model, neighborhood, sim);
    }
};
```

Figura 14: Creación de un Recomendador.

5.6.1 Evaluación Recomendador Tradicional

Una de las formas para calcular la precisión de las recomendaciones es separar el conjunto de datos en dos, un conjunto de datos de prueba y otro conjunto de datos de entrenamiento. Las preferencias (ratings) de los usuarios que han sido seleccionadas como prueba no están presentes al momento de crear el recomendador. Por lo tanto, se le pide al recomendador predecir los ratings de los datos de prueba, los cuales son comparados con los valores reales.

A pesar de que Mahout proporciona herramientas para realizar esta evaluación y calcular el error cuadrático medio (MAE) del recomendador, estas debieron ser modificadas debido a que el dataset obtenido presenta un problema de escasez de datos (conocido como data sparsity problem en inglés). Este problema se refiere a que los usuarios normalmente otorgan ratings a una pequeña cantidad de ítems presentes en el sistema. Debido a esto, no se pudo aplicar los evaluadores por defecto ya que sus resultados eran no numéricos. Esto quiere decir que el recomendador no podía generar recomendaciones para los usuarios seleccionados de prueba, los cuales son seleccionados de manera aleatoria por el evaluador.

Para solucionar esto, se modificó la interfaz que implementa el evaluador de error cuadrático medio con el objetivo de seleccionar como datos de prueba a los usuarios que han otorgado mayor número de ratings e ítems con mayor número de usuarios.

En primer lugar se implementó un método para encontrar los ítems con mayor número de usuarios. Este método se muestra en la Figura 15. En primer lugar se carga el archivo con la información de preferencias de los usuarios. Luego, utilizando un HashMap se contabiliza

la cantidad de ratings de cada ítem. Finalmente, los ítems con mayor cantidad de ratings se agregan a otro HashMap para ser entregado como parámetro de otro método.

```
public HashMap<String,Integer> Items() throws FileNotFoundException {
    File archivo = new File("ProyectoTitulo/general.txt");
    TsvParserSettings settings = new TsvParserSettings();
    TsvParser parser = new TsvParser(settings);
    List<String[]> allRows = parser.parseAll(new FileReader(archivo));
    HashMap<String, Integer> Items = new HashMap();
    HashMap<String, Integer> RestUsuario = new HashMap();
    for (String[] allRow : allRows) {
        if (RestUsuario.containsKey(allRow[1])) {
            RestUsuario.put(allRow[1], RestUsuario.get(allRow[1]) + 1);
        } else {
            RestUsuario.put(allRow[1], 1);
        }
    }
    for (Map.Entry<String, Integer> entry : RestUsuario.entrySet()) {
        if (entry.getValue() >= 100) {
            Items.put(entry.getKey(), 1);
        }
    }
    return Items;
}
```

Figura 15: Método para encontrar ítems con mayor número de usuarios.

Seguido de esto, se modificó el método *splitOneUserPrefs()* donde se realiza la selección de usuarios de prueba. Este método tiene dos listas de objetos *Preference*, donde en una se guardan las preferencias de prueba y en la otra lista las preferencias de entrenamiento.

En primer lugar se realiza un ciclo para recorrer el *PreferenceArray* del usuario y obtener cada *Preference* (usuario, ítem, valoración) del mismo. Dado que *Preference* representa la valoración otorgada de un usuario a un ítem, el número de *Preference* es equivalente al número de ratings que el usuario ha entregado. Seguido de esto, se pregunta si la cantidad de ratings es mayor o igual a 6. Si no cumple con esta condición, todos los ratings del usuario son dejados para entrenamiento. En caso de tener 6 o más ratings, se pregunta además si uno de los ítems que el usuario ha valorado, se encuentra dentro del HashMap

retornado del método *Ítems()*. En caso de cumplir también con esta condición, el *Preference* es agregado para el conjunto de prueba.

```
List<Preference> oneUserTrainingPrefs = null;
List<Preference> oneUserTestPrefs = null;
PreferenceArray prefs = dataModel.getPreferencesFromUser(userID);
int size = prefs.length();
for (int i = 0; i < size; i++) {
    Preference newPref = new GenericPreference(userID, prefs.getItemID(i), prefs.getValue(i));
    if (size >= 6 && items.containsKey(String.valueOf(prefs.getItemID(i)))
                                    && oneUserTestPrefs == null) {
        if (oneUserTestPrefs == null) {
            oneUserTestPrefs = new ArrayList<>(3);
        }
        oneUserTestPrefs.add(newPref);
    } else {
        if (oneUserTrainingPrefs == null) {
            oneUserTrainingPrefs = new ArrayList<>(3);
        }
        oneUserTrainingPrefs.add(newPref);
    }
}
```

Figura 16: Selección de usuarios de prueba.

La interfaz modificada se le llamó *CustomRecommenderEvaluator*. Para realizar la evaluación del recomendador creado en el punto 5.6, se utiliza el método *evaluate* el cual recibe como parámetro el recomendador y el modelo de datos. Los últimos parámetros no son utilizados en la interfaz modificada dado que estos representan la cantidad de datos que se separan para entrenamiento y prueba.

```
CustomRecommenderEvaluator evaluatorMAE = new MAE();
double valorMAE = evaluatorMAE.evaluate(tradicional, null, model, 0.9, 1);
System.out.println("\n Valor MAE : " + valorMAE);
```

Figura 17: Evaluación de recomendador.

Finalmente el MAE obtenido es de 0.9741008. Esto quiere decir que, en promedio las predicciones de rating que realiza el recomendador tienen una diferencia de 0.9741008 en comparación los rating reales.

5.7 Generación de Recomendaciones Multi-criterio

Para la generación de recomendaciones multi-criterio, se implementó el método de Similitud Promedio propuesto por Adomavicius (Adomavicius & Kwon, 2007), explicado en el punto 2.5.1.

Similitud Promedio calcula la similitud de cada uno de los aspectos por separado, para luego sumarlos y dividirlos por la cantidad de aspectos total. Para realizar esto, se desarrolló la clase *AVGMultiCriteriaSimilarity* que implementa la interfaz de *UserSimilarity*. Esta clase encapsula 5 objetos (para los 4 aspectos más el rating general) que implementen esta interfaz.

Para calcular la similitud entre dos usuarios, se implementó el método *userSimilarity* presentado en la Figura 18. Este método en primer lugar calcula la similitud entre los usuarios para cada aspecto por separado. Luego, para cada aspecto se determina si se logró calcular la similitud. En caso de que se logra calcular la similitud, es sumada a la similitud total de los usuarios y se suma 1 a la variable n (cantidad de aspectos). Esto es debido a que las preferencias de los aspectos de los usuarios fueron extraídas de sus reseñas, por lo tanto, para muchos usuarios no se tiene rating para todos los aspectos del ítem.

```
public double userSimilarity(long UserID1, long UserID2) throws TasteException {
    int n = 0;
    double simTotal = 0;
    double sim = a.userSimilarity(UserID1, UserID2);
    double sim1 = b.userSimilarity(UserID1, UserID2);
    double sim2 = c.userSimilarity(UserID1, UserID2);
    double sim3 = d.userSimilarity(UserID1, UserID2);
    double sim4 = e.userSimilarity(UserID1, UserID2);
    if (!Double.isNaN(sim)) {
        simTotal += sim;
        n += 1;
    }
    if (!Double.isNaN(sim1)) {
        simTotal += sim1;
        n += 1;
    }
```

Figura 18: Implementación de Similitud Promedio.

El método retorna la similitud total si el valor de n es mayor o igual a 3. Esto quiere decir que 3 es el número mínimo de aspectos a considerar en el cálculo de similitud entre usuarios. En caso de que n sea menor que 3, se retorna la similitud de un criterio.

Una vez implementada la técnica de Similitud Promedio, se procede a crear el recomendador multi-criterio. En primer lugar se cargan los datos de los aspectos, como se muestra en la Figura 19.

```
DataModel model = new FileDataModel(new File("ProyectoTitulo/general.txt"));
DataModel modelComida = new FileDataModel(new File("ProyectoTitulo/comida.txt"));
DataModel modelAmbiente = new FileDataModel(new File("ProyectoTitulo/ambiente.txt"));
DataModel modelServicio = new FileDataModel(new File("ProyectoTitulo/servicio.txt"));
DataModel modelPrecio = new FileDataModel(new File("ProyectoTitulo/precio.txt"));
```

Figura 19: Cargando datos de aspectos.

Seguido de esto, se crea el recomendador multi-criterio, como se muestra en la Figura 20. La medida de similitud seleccionada es el Coeficiente de Correlación de Pearson y el tamaño del vecindario es de 10. A diferencia del recomendador creado en el punto 5.6, en este caso se utiliza la clase *AVGMultiCriteriaSimilarity* que contiene las similitudes de los diferentes aspectos. Cabe destacar que los parámetros utilizados son los mismos que en el recomendador tradicional.

```
RecommenderBuilder multiCriterio = new RecommenderBuilder() {
    @Override
    public Recommender buildRecommender(DataModel model) throws TasteException {
        UserSimilarity sim = new PearsonCorrelationSimilarity(model);
        UserSimilarity sim2 = new PearsonCorrelationSimilarity(modelPrecio);
        UserSimilarity sim3 = new PearsonCorrelationSimilarity(modelServicio);
        UserSimilarity sim4 = new PearsonCorrelationSimilarity(modelComida);
        UserSimilarity sim5 = new PearsonCorrelationSimilarity(modelAmbiente);
        AVGMultiCriteriaSimilarity sum = new AVGMultiCriteriaSimilarity(sim,sim2,sim3,sim4,sim5);
        UserNeighborhood neighborhood = new NearestNUserNeighborhood(10, sum, model);
        return new GenericUserBasedRecommender(model, neighborhood, sum);
    }
};
```

Figura 20: Recomendador multi-criterio.

5.7.1 Evaluación Recomendador Multi-criterio

Para la evaluación del recomendador multi-criterio, se utiliza el evaluador creado en el punto 5.6.1, como se muestra en la Figura 21.

```
CustomRecommenderEvaluator evaluatorMAE = new MAE();
double MAEMulti = evaluatorMAE.evaluate(multiCriterio, null, model, 0.9, 1);
System.out.println("\n Valor MAE Multi-criterio : " + MAEMulti);
```

Figura 21: Evaluación recomendador multi-criterio.

Finalmente el MAE del recomendador multi-criterio es de 0.9025005. Esto quiere decir que, en promedio las predicciones de rating que realiza el recomendador tienen una diferencia de 0.9025005 en comparación los rating reales.

6 Desarrollo de la aplicación

En este capítulo se presenta los requerimientos del sistema y su diagrama de casos de uso.

6.1 Especificación de Requerimientos

Identificación del requerimiento	RF01
Nombre del Requerimiento	Seleccionar y abrir archivo.
Descripción del requerimiento	El sistema debe permitir la selección y apertura de un archivo presente en el disco duro. .

Identificación del requerimiento	RF02
Nombre del Requerimiento	Pre procesar datos.
Descripción del requerimiento	El sistema debe permitir la aplicación de diferentes técnicas de limpieza de datos para mejorar la efectividad del modelo de clasificación a generar.

Identificación del requerimiento	RF03
Nombre del Requerimiento	Crear modelo de clasificación.
Descripción del requerimiento	Se debe permitir la selección de un algoritmo de clasificación para generar un modelo y guardarlo. .

Identificación del requerimiento	RF04
Nombre del Requerimiento	Cambiar formato de archivo.
Descripción del requerimiento	El sistema debe permitir el cambio de formato de archivo TSV a ARFF.

Identificación del requerimiento	RF05
Nombre del Requerimiento	Presentar métricas de evaluación de modelo de clasificación..
Descripción del requerimiento	El sistema debe permitir la visualización de métricas de evaluación del modelo de clasificación generado..

Identificación del requerimiento	RF06
Nombre del Requerimiento	Técnicas de recomendación.
Descripción del requerimiento	El sistema debe permitir el ajuste de diferentes técnicas de recomendación tales como medida de similitud, creación de vecindario y tipo de recomendación (tradicional o multi-criterio).

Identificación del requerimiento	RF07
Nombre del Requerimiento	Evaluación de recomendación.
Descripción del requerimiento	El sistema debe permitir la visualización de métricas de evaluación de las recomendaciones generadas.

6.2 Diagrama de Casos de Uso

En esta sección se presenta el diagrama de casos de uso del software realizado.

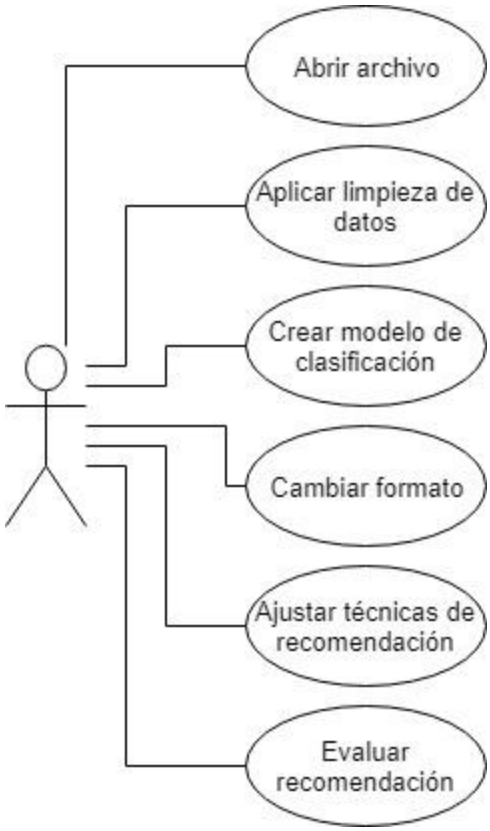

Figura 22: Diagrama de Casos de Uso

A continuación se explica cada caso de uso.

- **Abrir archivo:** El usuario puede seleccionar y abrir un archivo.

- **Aplicar limpieza de datos:** El usuario puede elegir entre 3 tipos de limpieza de datos a aplicar (filtro de tildes, filtro de stopwords y eliminación de puntuaciones y otros caracteres). Se debe tener un archivo en formato TSV abierto.

- **Crear modelo de clasificación:** Mediante una lista desplegable el usuario puede elegir el algoritmo de clasificación a utilizar. Se debe tener abierto un archivo en formato ARFF para aplicar el algoritmo.

- **Cambiar formato:** Una vez aplicada la limpieza de datos sobre un archivo TSV, el usuario puede cambiar su formato a ARFF.

- **Ajustar técnicas de recomendación:** El usuario puede ajustar las técnicas de recomendación de medida de similitud, creación de vecindario y tipo de recomendación

- **Evaluar recomendación:** Seleccionadas las técnicas de recomendación, el usuario procede a generar recomendaciones y se presentan métricas de evaluación.

7 Uso de la Aplicación

En este capítulo se presenta el funcionamiento de la aplicación desarrollada en este proyecto de título.

7.1 Limpieza de Datos

Al iniciar la aplicación, el usuario se encontrará con las funcionalidades de Limpieza de Datos, Clasificación y Recomendación, como se muestra en la Figura 23. Para comenzar con la limpieza de datos, el usuario debe hacer click en "Abrir archivo" y seleccionar el archivo en formato TSV donde se encuentren las oraciones. La ruta del archivo se mostrará en el campo de texto.

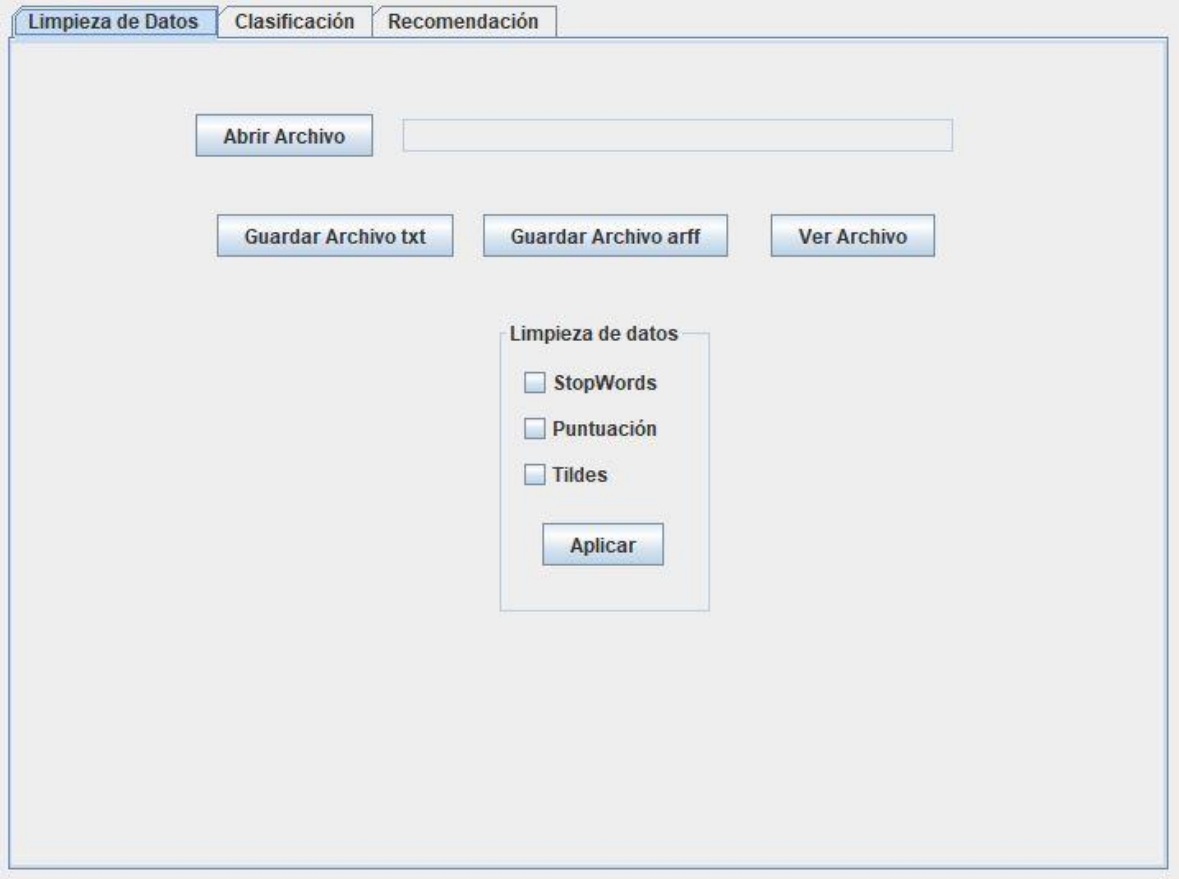

Figura 23: Limpieza de Datos.

Seguido de esto el usuario puede seleccionar alguna o todas las técnicas de limpieza de datos presentes en el sistema haciendo click en las cajas correspondientes. Luego el usuario debe apretar el botón "Aplicar" para aplicar las técnicas seleccionadas. En caso de que la

limpieza se aplique de forma exitosa, se mostrará un mensaje. Luego de aplicar la limpieza de datos, el usuario puede guardar el contenido creado en un nuevo archivo de texto en formato TSV o puede guardarlo directamente en formato ARFF para comenzar con el proceso de clasificación. El usuario puede ver el contenido de los archivos con el botón "Ver Archivo", el cual abre una ventana para seleccionar un archivo y mostrar su contenido.

7.2 Clasificación

Para comenzar el proceso de crear un modelo de clasificación, el usuario debe cambiar a la pestaña de "Clasificación", hacer click en "Abrir Archivo" y seleccionar un archivo en formato ARFF. Si el archivo se ha abierto correctamente, se mostrará su ruta en el campo de texto a un lado del botón.

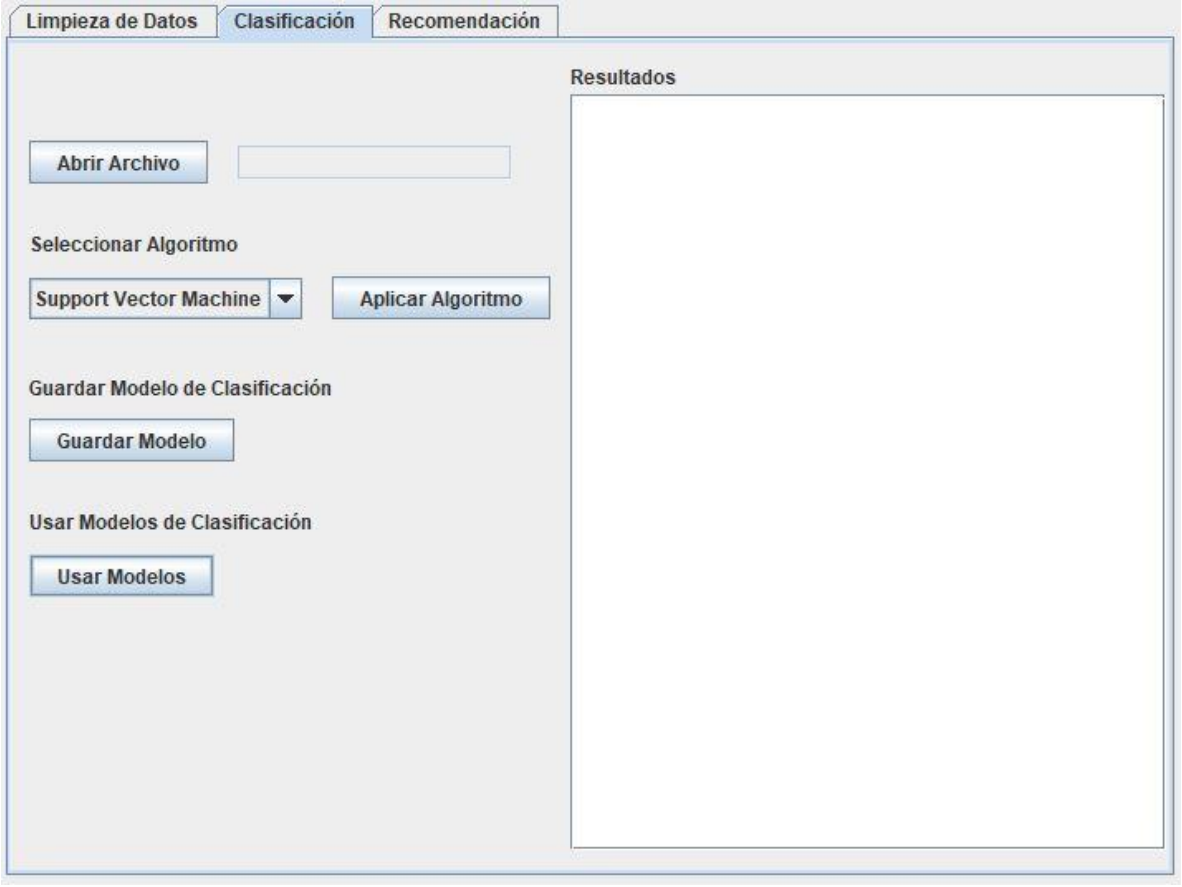

Figura 24: Pantalla de Clasificación.

Seguido de esto el usuario puede seleccionar el algoritmo clasificador de una lista y al hacer click en “Aplicar Algoritmo” se crea un modelo de clasificación en base al algoritmo seleccionado y se presentan métricas de evaluación y la matriz de confusión en el lado derecho de la pantalla, como se muestra en la Figura 25.

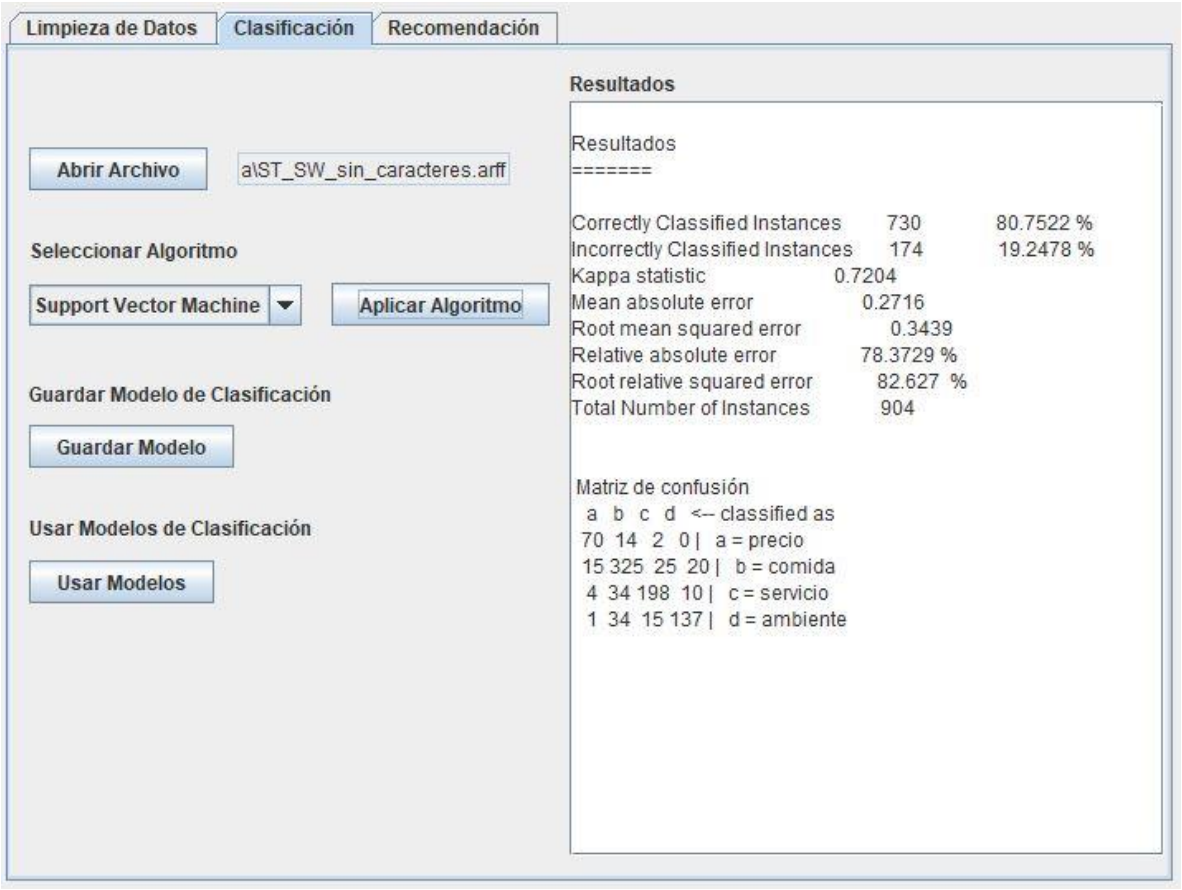

Figura 25: Resultados de clasificación.

El usuario tiene la opción de guardar el modelo de clasificación generado, haciendo click en “Guardar Modelo”. Finalmente, el usuario puede utilizar ambos modelos de clasificación para clasificar las oraciones de las reseñas presentes en el dataset creado en el capítulo 4. Para esto, debe hacer click en el botón “Usar Modelos” y seleccionar el archivo donde se encuentran las reseñas. Seguido de esto se crean 5 archivos, uno por cada aspecto considerado y un archivo con el rating general del ítem.

7.3 Recomendación

Al cambiar a la pestaña de "Recomendación", se le presenta al usuario una serie de opciones para ajustar las técnicas de recomendación. Esto se muestra en la Figura 26. La primera opción que se muestra es el tipo de recomendación, donde el usuario puede elegir entre recomendación tradicional y recomendación multi-criterio.

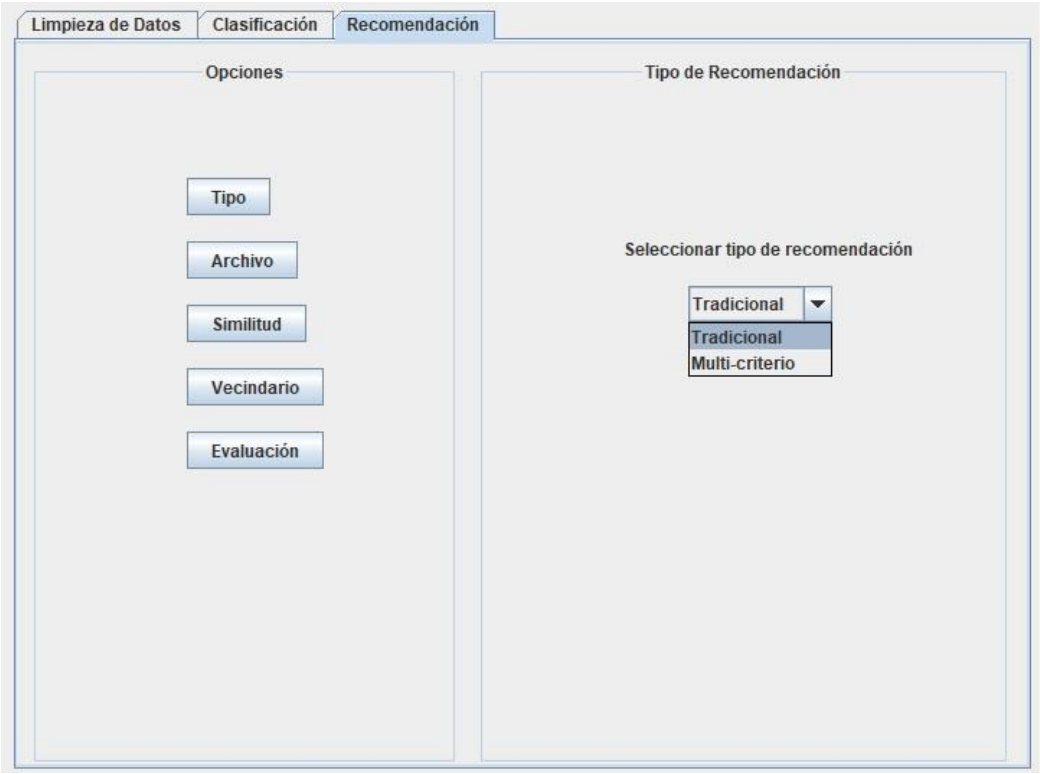

Figura 26: Selección de tipo de Recomendación.

Al seleccionar recomendación tradicional, la opción de “Archivo” presentará un botón para abrir un archivo en formato TSV o CSV con la información de preferencias de los usuarios, como se muestra en la Figura 27.

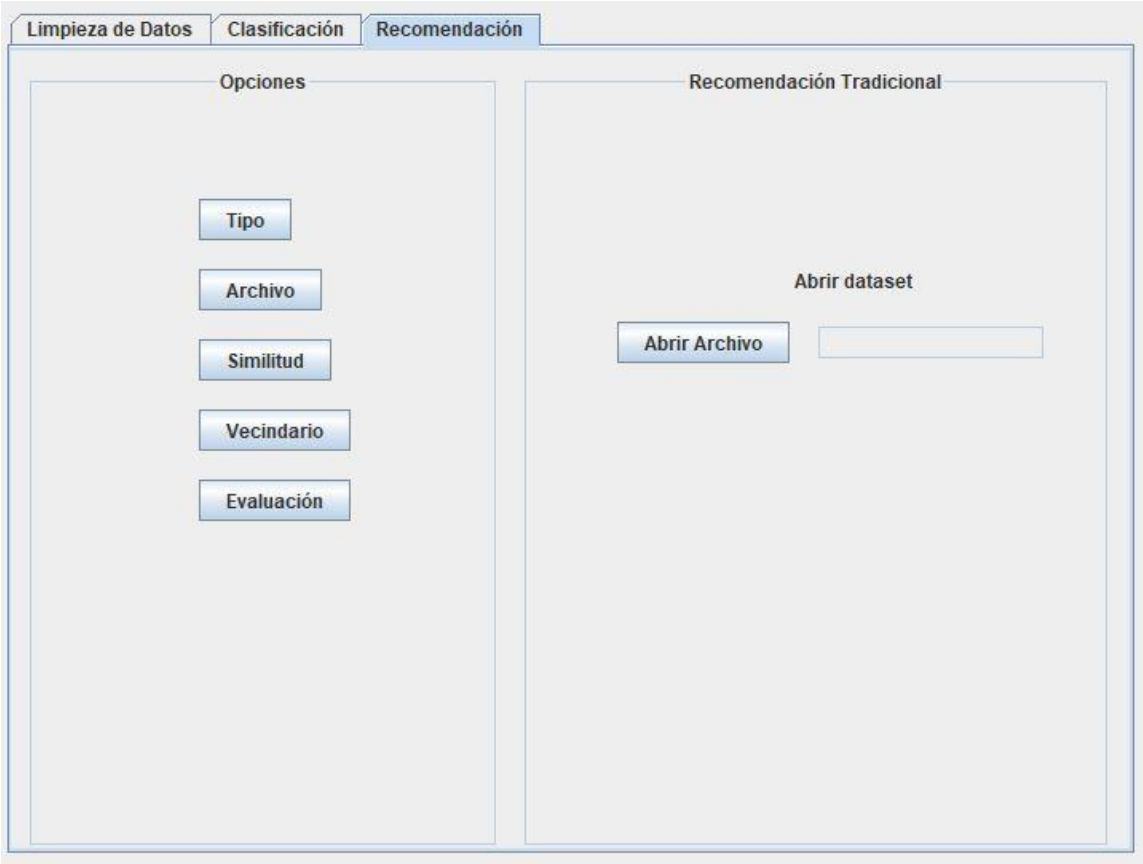

Figura 27: Recomendación tradicional.

Si el usuario selecciona recomendación multi-criterio, la opción de “Archivo” presentará la opción de cargar 5 archivos (valoración general más 4 aspectos) en formato TSV o CSV con las preferencias del usuario respecto a los aspectos del ítem. En la Figura 28 se presenta esta pantalla.

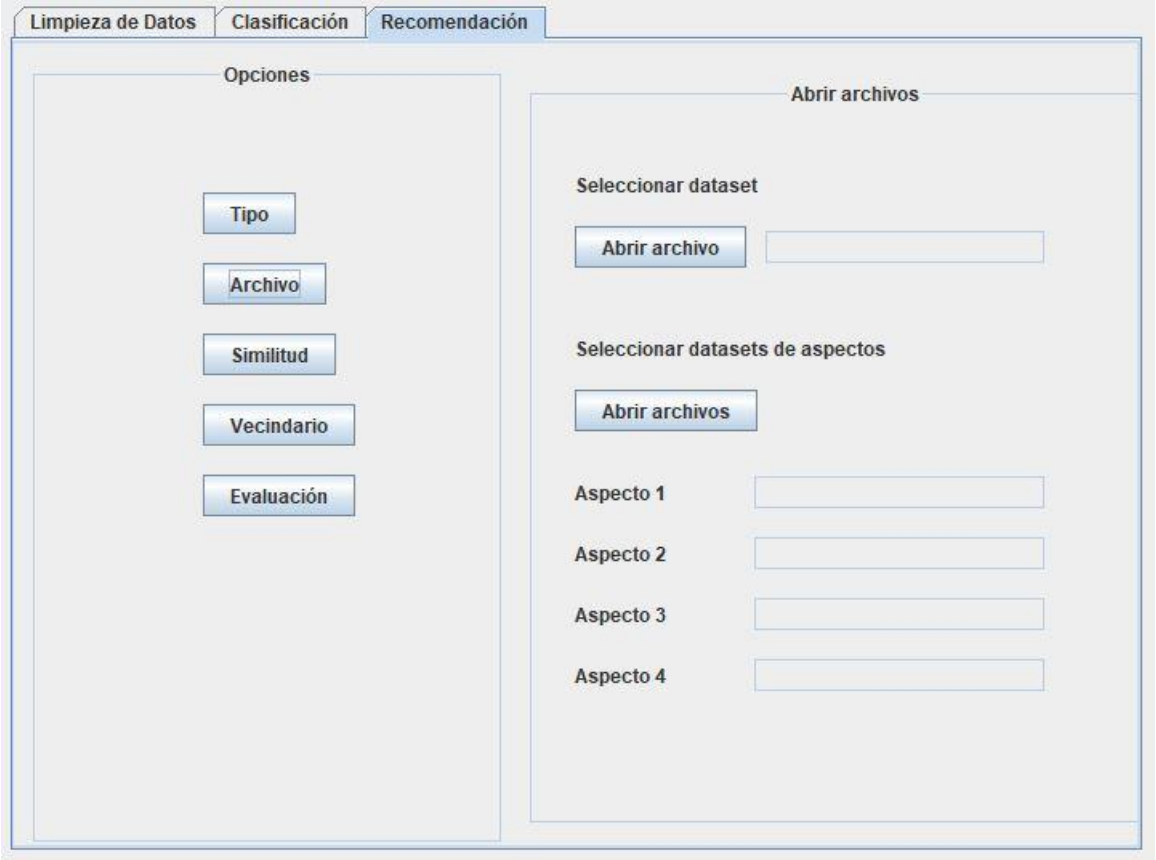

Figura 28: Recomendación multi-criterio.

Seleccionado el tipo de recomendación y cargado los respectivos archivos, el usuario puede seleccionar la medida de similitud utilizada y el tipo de vecindario a utilizar. Esto se muestra en la Figura 29 y Figura 30.

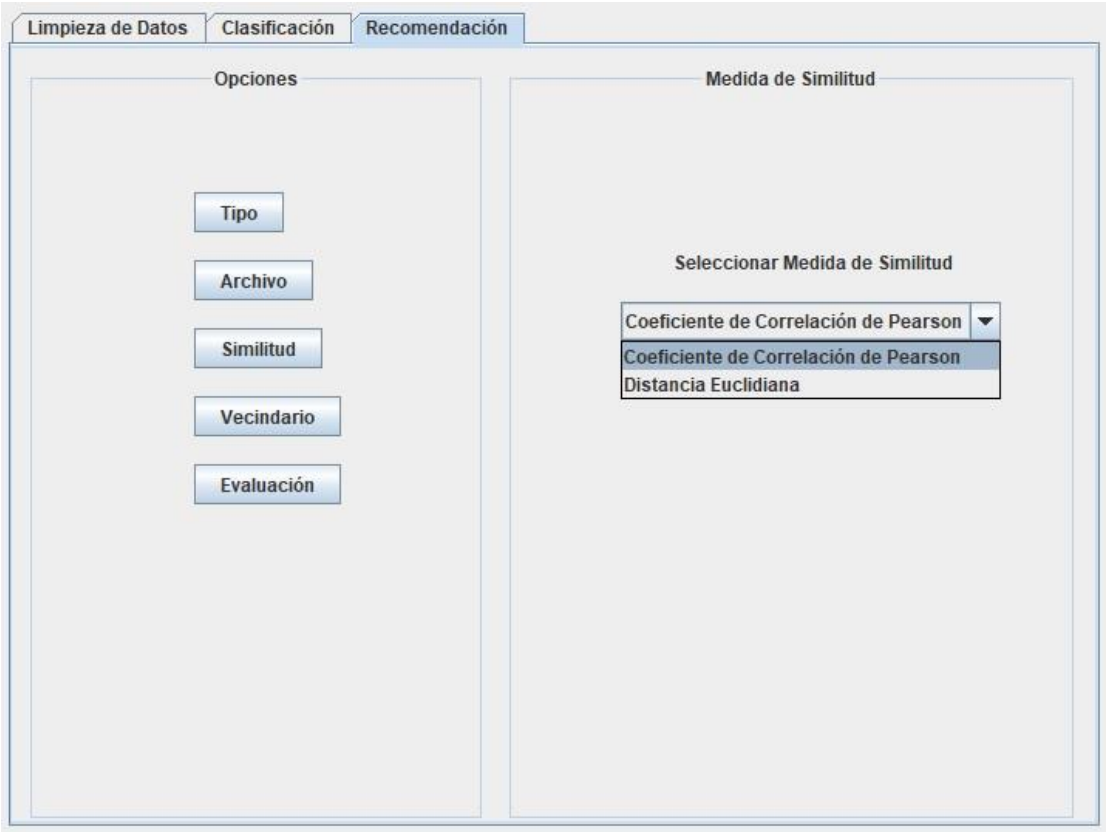

Figura 29: Selección medida de similitud.

Respecto a la selección de vecindario, en caso de elegir Umbral de similitud se pide un valor de umbral en lugar de un tamaño de vecindario.

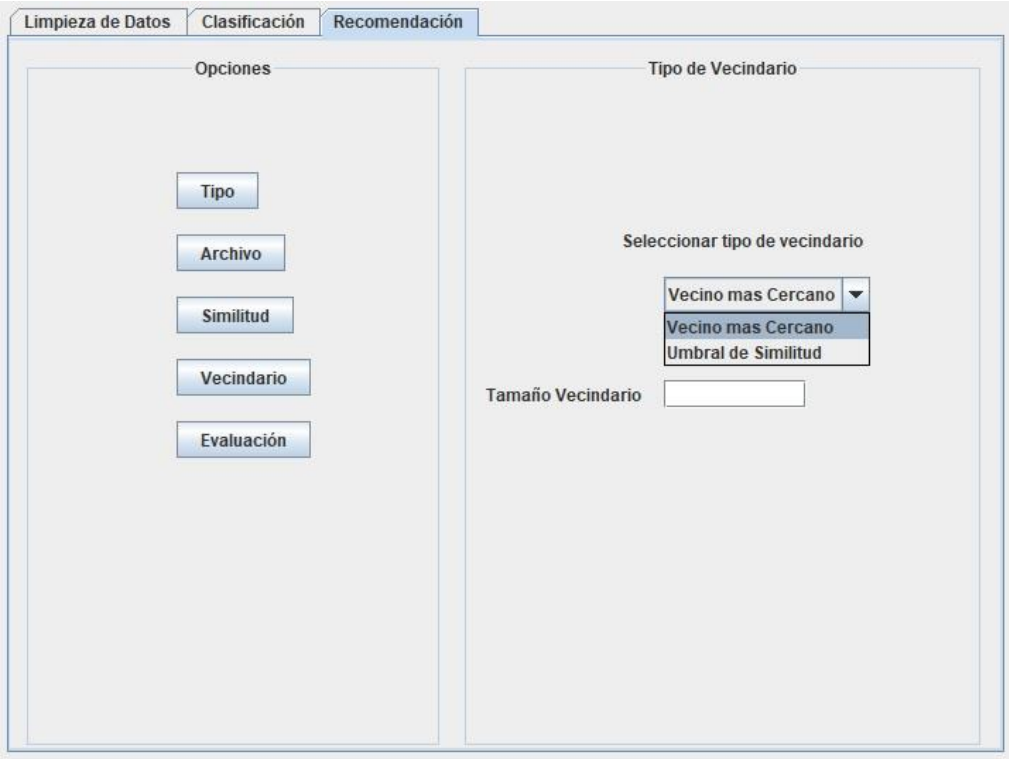

Figura 30: Selección de vecindario.

Finalmente, una vez que el usuario ha abierto los archivos correspondientes y seleccionadas las técnicas de recomendación correspondientes, se puede realizar la evaluación de la recomendación mostrando su error absoluto medio. Esto se presenta en la Figura 31.

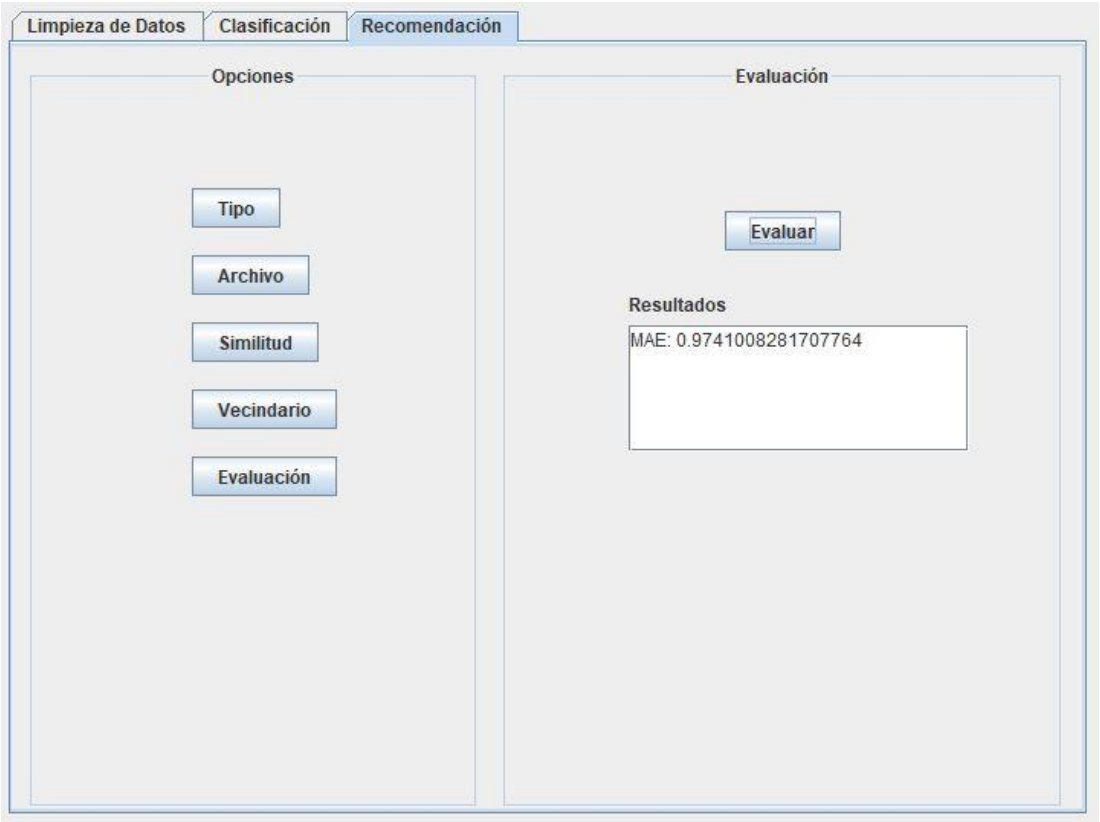

Figura 31: Evaluación de recomendación.

8 Experimento

El siguiente experimento tiene como objetivo responder a la siguiente pregunta:

1. ¿Cómo es el rendimiento del sistema de recomendación creado en comparación a un sistema de recomendación tradicional?

El experimento se realizó utilizando el dataset de ratings multi-criterio obtenido del proceso de análisis de sentimiento basado en aspecto realizado anteriormente. En la Tabla 12 se presenta la estructura de este dataset. Para lograr este dataset, en primer lugar se procesaron las reseñas quitando stopwords, tildes y signos de puntuación. Para extraer aspectos y sentimientos de las reseñas se utilizó un enfoque de aprendizaje automático supervisado. Como algoritmo de recomendación tradicional y multi-criterio se utilizó filtrado colaborativo basado en usuario. Para crear el vecindario se utilizó Coeficiente de Correlación de Pearson. Se utilizó un tamaño de vecindario de 10. El tamaño de vecindario es pequeño debido al problema de escasez de datos mencionado anteriormente.

La efectividad de las recomendaciones fue evaluada con error absoluto medio.

ID Usuario	ID Restaurante	General	Comida	Ambiente	Servicio	Precio
0	5355	4.0			5.0	
1	5355	5.0			5.0	
2	5355	3.5	5.0	5.0		
3	5355	4.0		5.0		

Tabla 12: Dataset multi-criterio.

8.1 Análisis de Resultados

La Tabla 13 muestra la precisión de predicción de rating obtenida de los recomendadores.

	MAE
Recomendador Tradicional	0.9741008
Recomendador Multi-criterio	0.9025005

Tabla 13: Precisión de recomendadores.

Se puede observar de la tabla que la precisión del recomendador multi-criterio es mejor en comparación a la del recomendador tradicional. Por lo tanto, respondiendo a la pregunta que se realizó al inicio de este experimento, se puede decir que el rendimiento del sistema de recomendación multi-criterio creado a partir de las reseñas de los usuarios es mejor en comparación al sistema de recomendación tradicional.

A pesar de los resultados, no es posible asegurar que esto sea así en todos los casos, dado que debido al problema de escasez de datos presentado en el dataset, los datos de prueba se vieron reducidos a 12 usuarios. Este problema fue aún mayor cuando se realizó las recomendaciones multi-criterio debido a que en pocas ocasiones un usuario otorgará su opinión de todos los aspectos en una reseña. Este es un problema que se debe tener en cuenta en futuros trabajos de sistemas de recomendación multi-criterio basado en reseñas de usuarios.

9 Conclusiones

En este proyecto de título se implementó un sistema de recomendación multi-criterio creado a partir de reseñas de restaurantes en idioma español. En primer lugar se creó un dataset de comentarios de restaurantes en español extraídos del sitio web Atrápalo.com. A partir de este dataset se realizó un análisis de sentimiento basado en aspecto para determinar la polaridad y aspectos considerados en las reseñas extraídas. Para esto se utilizó un enfoque de aprendizaje automático. Por lo tanto, fue necesario separar las reseñas por oración y etiquetarlas de acuerdo al aspecto y sentimiento expresado para posteriormente aplicar un algoritmo clasificador. Para mejorar la efectividad del algoritmo se aplicaron diferentes técnicas de limpieza de datos.

La información extraída del proceso de análisis de sentimiento basado en aspecto realizado se utilizó para crear un sistema de recomendación multi-criterio el cual fue evaluado en comparación a un sistema de recomendación tradicional. Para la creación del sistema de recomendación multi-criterio se utilizó la técnica de Similitud Promedio propuesta por Adomavicius.

Finalmente se desarrolló un software con interfaz gráfica de usuario para el ajuste de las diferentes técnicas utilizadas en el proceso de análisis de sentimiento basado en aspecto y sistemas de recomendación,

10 Bibliografía

Adomavicius, G., & Kwon, Y. (2007). New recommendation techniques for multicriteria rating systems. *IEEE Intelligent Systems*, *22*(3), 48–55. https://doi.org/10.1109/MIS.2007.58

Adomavicius, G., & Kwon, Y. (2015). Multi-Criteria Recommender Systems. In *Recommender Systems Handbook* (pp. 847–880). Boston, MA: Springer US. https://doi.org/10.1007/978-1-4899-7637-6_25

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 734–749. https://doi.org/10.1109/TKDE.2005.99

Aggarwal, C. C. (2016). Knowledge-Based Recommender Systems. In *Recommender Systems* (pp. 167–197). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-29659-3_5

Buhmann, M. D., Melville, P., Sindhwani, V., Quadrianto, N., Buntine, W. L., Torgo, L., … Fürnkranz, J. (2011). Recommender Systems. In *Encyclopedia of Machine Learning* (pp. 829–838). Boston, MA: Springer US. https://doi.org/10.1007/978-0-387-30164-8_705

Burke, R. (2007). Hybrid Web Recommender Systems. In *The Adaptive Web* (pp. 377–408). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-72079-9_12

Chen, L., Chen, G., & Wang, F. (2015). Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction*, *25*(2), 99–154. https://doi.org/10.1007/s11257-015-9155-5

Elgueta, J. (2017). *Comparación de rendimiento de técnicas de aprendizaje automático para análisis de afecto sobre textos en español*. Universidad del Bío-Bío, Chile.

Feldman, R., & Sanger, J. (2006). *Text Mining Handbook*.

Gunawardana, A., & Shani, G. (2015). Evaluating Recommender Systems. In *Recommender Systems Handbook* (pp. 265–308). Boston, MA: Springer US. https://doi.org/10.1007/978-1-4899-7637-6_8

Liu, B. (2012). *Sentiment Analysis and Opinion Mining*. *Synthesis lectures on human language technologies*. https://doi.org/10.2200/S00416ED1V01Y201204HLT016

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., & Mcclosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55–60.

Musto, C., de Gemmis, M., Semeraro, G., & Lops, P. (2017). A Multi-criteria Recommender System Exploiting Aspect-based Sentiment Analysis of Users' Reviews. *Proceedings of the Eleventh ACM Conference on Recommender Systems - RecSys '17*, 321–325. https://doi.org/10.1145/3109859.3109905

Ning, X., Desrosiers, C., & Karypis, G. (2015). A Comprehensive Survey of Neighborhood-Based Recommendation Methods. In *Recommender Systems Handbook* (pp. 37–76). Boston, MA: Springer US. https://doi.org/10.1007/978-1-4899-7637-6_2

Owen, S., Anil, R., Dunning, T., & Friedman, E. (2012). *Mahout in Action*.

Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender Systems: Introduction and Challenges. In *Recommender Systems Handbook* (pp. 1–34). Boston, MA: Springer US. https://doi.org/10.1007/978-1-4899-7637-6_1

Sebastiani, F., & Fabrizio. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, *34*(1), 1–47. https://doi.org/10.1145/505282.505283

Witten, I. H., & Frank, E. (2005). *Introduction to Weka*. *Data Mining: Practical machine learning tools and techniques*. https://doi.org/http://dx.doi.org/10.1016/B978-0-12-374856-0.00005-5

Zhang, Y., Chen, M., & Liu, L. (2015). A review on text mining. In *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)* (pp.

681–685). IEEE. https://doi.org/10.1109/ICSESS.2015.7339149

11 Anexos

11.1 Especificación de Casos de Uso

A continuación se presenta la especificación de casos de uso del sistema realizado.

Caso de uso	Abrir archivo.
Referencia cruzada	RF01
Actores	Usuario.
Descripción	Se carga un archivo en el sistema.
Precondiciones	Ninguna.
Postcondiciones	El sistema mantiene un archivo abierto para ser utilizado.

Caso de uso	Aplicar limpieza de datos.
Referencia cruzada	RF02
Actores	Usuario.
Descripción	Se aplica una o más técnicas de limpieza de datos a un archivo de texto.
Precondiciones	El sistema debe tener cargado un archivo de texto en formato TSV.
Postcondiciones	El contenido del archivo es modificado y puede ser guardado en otro archivo.

Caso de uso	Crear modelo de clasificación.
Referencia cruzada	RF03, RF05
Actores	Usuario.
Descripción	Se selecciona y se aplica un algoritmo de clasificación.
Precondiciones	El sistema debe tener cargado un archivo ARFF.
Postcondiciones	Se crea un modelo de clasificación utilizando el algoritmo seleccionado y se presentan métricas de evaluación.

Caso de uso:	Cambiar formato.
Referencia cruzada	RF04
Actores	Usuario.
Descripción	El usuario puede cambiar el formato de un archivo de texto en formato TSV a ARFF.
Precondiciones	El sistema debe tener un archivo de texto TSV cargado.
Postcondiciones	Se crea un nuevo archivo en formato ARFF.

Caso de uso:	Ajustar técnicas de recomendación.
Referencia cruzada	RF06
Actores	Usuario.
Descripción	El usuario puede seleccionar diferentes técnicas de recomendación.
Precondiciones	Ninguna.
Postcondiciones	Ninguna.

Caso de uso:	Evaluar recomendación.
Referencia cruzada	RF07
Actores	Usuario.
Descripción	Se aplican las técnicas de recomendación seleccionadas y se presenta el error absoluto medio.
Precondiciones	El sistema debe tener el o los archivos cargados y las técnicas de recomendación deben estar seleccionadas.
Postcondiciones	Se presenta el error absoluto medio de las recomendaciones.