UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

21|22

Trabajo fin de grado

Recomendación de restaurantes explotando aspectos de reseñas

Daniel Torres Candil

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

excelencia Campus Internacional UAM CSIC

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Recomendación de restaurantes explotando aspectos de reseñas

Autor: Daniel Torres Candil
Tutor: Alejandro Bellogín Kouki

enero 2022

Daniel Torres Candil
Recomendación de restaurantes explotando aspectos de reseñas

Daniel Torres Candil
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mi familia

The biggest risk is not taking any risk...
In a world that is changing really quickly,
the only strategy that is guaranteed
to fail is not taking risks.

Mark Zuckerberg

Agradecimientos

Tras unos meses de duro trabajo y constantes cambios en el devenir de este proyecto de fin de grado, en primer lugar me gustaría agradecer a mi tutor Alejandro Bellogín la implicación y paciencia que me ha trasladado desde el primer día.

También, me gustaría nombrar a ciertas personas que me han ayudado a no desistir y a terminar entregando este trabajo en una fecha límite en mi calendario personal. Gracias por ello a mi novia, María Cristina Vegas, y a mi familia: a mi madre, María Isabel Candil, a mi padre, José Luis Torres, y a mi hermana, María Torres. Sin ellos esto habría sido mucho más complicado.

Resumen

Cada día más y más gente trata de encontrar un sitio único donde conectar con la gastronomía. Esto ha llevado consigo la creación de múltiples tipos de restaurantes diferentes: cantoneses, mexicanos, de cocina fusión, etc. Sin embargo, muchas personas no han perdido el gusto por los restaurantes tradicionales. Es por esta globalización, y la variedad de locales, por lo que se necesita desarrollar un sistema que nos ayude a recomendar, en función de nuestros gustos y opiniones, los establecimientos gastronómicos más acordes a nuestras preferencias.

En este trabajo de fin de grado se lleva a cabo un proyecto de investigación en el que, partiendo prácticamente desde cero, se pone en marcha un estudio de diferentes sistemas de recomendación. En el presente trabajo se expondrá al lector los aspectos más importantes a tener en cuenta a la hora de recomendar ciertos ítems a usuarios. Para ello, se partirá de un conjunto de datos inicial, compuesto de multitud de reseñas realizadas por los usuarios de la plataforma de Yelp a ciertos restaurantes, el cual habrá que analizar y procesar hasta amoldarlo a nuestras preferencias, para después poder generar las recomendaciones. Entre estas preferencias se encuentran la extracción de cierta información propia de las reseñas, como son los aspectos (es decir, sustantivos relacionados con el ámbito de los restaurantes que definen una característica o un atributo sobre los mismos y productos relacionados, como la comida o el ambiente). Además, también se hace un análisis del sentimiento de las mismas, personalizando así cada recomendación devuelta al usuario, sin olvidarnos de la valoración numérica asociada a cada reseña. Con todo ello, se compararán diferentes tipos de sistemas de recomendación y se sacarán conclusiones a partir de las evaluaciones de los mismos.

Para poder desarrollar este proyecto se han estudiado y utilizado varias tecnologías; en concreto, algoritmos de recomendación basados en vecinos próximos y técnicas de procesamiento del lenguaje natural, que nos permiten entender las reseñas y extraer los ya mencionados aspectos, así como estimar su sentimiento.

Palabras clave

Sistema de recomendación, procesamiento del lenguaje natural, conjunto de datos, aspectos, función de similitud, evaluación, Python

Abstract

Every day more and more people try to find a unique place to connect with gastronomy. This has led to the creation of multiple types of restaurants, such as cantonese, mexicans, fusion cuisine, etc. However, many people have not lost their taste for traditional restaurants. It is due to this globalization, and the variety of establishments, that it is necessary to develop a system that helps us recommend the restaurants that best suit our preferences based on our tastes and opinions.

In this final degree thesis, a research project is carried out, starting almost from scratch, in which we will study a variety of different recommendation systems. This project will expose the reader to the most important aspects to take into account when recommending certain items to users. In order to do so, we will start from an initial dataset obtained from Yelp, composed of multiple reviews made by users to certain restaurants. This dataset will need to be analyzed and processed to mold it to our preferences, to then be able to generate the recommendations. Among these preferences is the extraction of certain information from the reviews, such as aspects (i.e., nouns related to the field of restaurants that define a characteristic or attribute about related products, such as food or ambience). In addition, the reviews are analyzed to detect their sentiment. This will help personalize each returned recommendation, without forgetting the rating that these users generate in each review. With all this, we will compare different types of recommendation systems and the conclusions will be drawn from their evaluations.

In order to develop this project, several technologies have been studied and used; specifically, recommendation algorithms based on nearest neighbors, and natural language processing techniques, which have allowed us to understand the reviews and extract the aforementioned aspects, as well as to estimate their sentiment.

Keywords

Recommender system, natural language processing, dataset, aspects, similarity function, evaluation, Python

Índice

Listas

Lista de ecuaciones

Lista de figuras

Lista de tablas

1

Introducción

En este primer capítulo se explicarán al lector las razones que han motivado a hacer este proyecto, así como los objetivos que se pretenden cumplir una vez terminado. A su vez, también habrá disponible una última sección en la que se explicará la estructura de esta memoria.

1.1. Motivación

El mundo está lleno de gente con gustos muy diversos. Hay veces incluso que pueden llegar a ser completamente opuestos. Diferentes gustos musicales, distintas formas de vestir, de peinarse, personas a las que les gusta disfrutar de un tipo determinado de actividades, etc. En esta diferenciación, los gustos gastronómicos no iban a ser menos. ¿Qué aspectos valoran más los usuarios a la hora de acudir a un restaurante? ¿La limpieza? ¿El trato del personal? ¿La calidad de la comida? No todos valoran los mismos.

Muchas han sido las veces en que hemos ido a un restaurante y el personal no ha sido lo suficientemente amable, o la comida no estaba como el comensal se esperaba. Incluso podemos creer que hemos pagado de más por un servicio mediocre o no acorde al precio. Para evitar que estas situaciones nos surjan de improviso, generalmente se suele pedir consejo a gente de nuestro alrededor, ya sean amigos, familiares o, incluso, en las redes sociales. Sin embargo, esto suele ser a veces bastante pesado y aburrido.

Pero ¿por qué basarse únicamente en las opiniones de personas de nuestro alrededor? ¿Por qué no pensar más allá y tratar de buscar una solución que ayude a recomendar restaurantes gracias a las opiniones de todos los comensales en vez de reducirlo a un grupo más pequeño?

Tras haber pensado en cómo aligerar el proceso de toma de decisiones, se decidió optar por desarrollar una serie de sistemas de recomendación que tuviesen en cuenta no solo las reseñas de las personas más parecidas sino de aquellas cuya opinión sobre determinados *aspectos* de los restaurantes coincidamos más. Así, cuantas más reseñas haya disponibles, y más capaces de entender y capturar esos aspectos y las opiniones correspondientes, más preciso deberá ser el sistema de recomendación y con más seguridad estará recomendando un local a un usuario concreto.

1.2. Objetivos

El objetivo principal de este trabajo de fin de grado es implementar y analizar varios algoritmos de recomendación que ofrezcan a los usuarios ciertos restaurantes en función de sus gustos, basándose tanto en los aspectos extraídos de las reseñas como en las puntuaciones asociadas a ellas.

Otro de los objetivos que se quiere cumplir es el de tratar de añadir simplicidad y rapidez a la hora de decidir sobre si un restaurante se adhiere a las preferencias de un usuario o no, dejando de lado todo el proceso de hablar con la gente para llegar a una decisión final.

También cabe destacar todo el proceso seguido. Desde la extracción y procesamiento de los datos hasta la posterior evaluación de los sistemas en función de varias métricas.

Por último, destacar otros objetivos más simples, pero también importantes en el desarrollo del proyecto, como son el aprendizaje continuo o la capacidad de saber diferenciar funcionalidades y separarlas por módulos.

1.3. Estructura

A continuación, se expone un resumen de cada uno de los capítulos que componen esta memoria.

Capítulo 1. Introducción. Motivación y objetivos que llevan al desarrollo del proyecto.

Capítulo 2. Estado del arte. Explicación detallada de cada una de las tecnologías que se utilizarán y/o implementarán en capítulos posteriores.

Capítulo 3. Diseño e implementación. Se exponen los diferentes módulos con los que cuenta el proyecto, tanto una explicación de qué hace cada uno como una descripción del detalle de su implementación.

Capítulo 4. Pruebas y resultados. Presentación de los equipos donde se han hecho las pruebas y explicación de los resultados de los algoritmos de recomendación implementados.

Capítulo 5. Conclusiones y trabajo futuro. Se detallan las conclusiones del proyecto, además de añadir una sección con los posibles progresos y mejoras que se pueden realizar en el futuro.

2

Estado del arte

En este capítulo se hará una explicación de las tecnologías que se han utilizado en el desarrollo de este trabajo final de grado, como son el procesamiento del lenguaje natural, los sistemas de recomendación o las métricas de evaluación.

2.1. Natural Language Processing (NLP)

El procesamiento del lenguaje natural (Natural Language Processing, en inglés) es una rama de la inteligencia artificial que ayuda a los ordenadores a procesar, analizar y entender la lengua humana, así como también a elaborar nuevos textos basados en las reglas del idioma.

Saber si el sentido de una frase es positivo o negativo es un problema que para el ser humano es casi innato determinar. Sin embargo, esta percepción no es más que un conjunto de reglas gramaticales que el ser humano ha ido comprendiendo y perfeccionando a lo largo de los años, y por ello se aplica de manera inherente.

En cambio, un ordenador no puede determinar el sentido ni el sentimiento de una frase sin conocer esas reglas. Sin embargo, intentar programar todo este conjunto de instrucciones no es algo trivial, además de que se deben tener en cuenta todos los posibles matices y ambigüedades existentes.

Para desarrollar esta funcionalidad, se puede hacer uso del aprendizaje automático (ML, por sus siglas en inglés: *Machine Learning*), el cual ayudará con el procesamiento y aprendizaje de estas reglas [1].

2.1.1. Tokenización

El primer paso para el procesamiento del lenguaje es hacer que el texto sea entendido por el ordenador. Un texto es una secuencia de caracteres que, sin un conjunto de reglas, no aportan ningún valor. Para su correcto procesamiento, éste debe dividirse de manera atómica. Aquí es donde entra el concepto de **tokenización**.

La tokenización es una forma de separar un texto en unidades más pequeñas llamadas **tokens**. Estos tokens pueden ser tanto palabras como caracteres o, incluso, subpalabras.

Consideremos la siguiente frase:

"*He is the founder of the biggest AI enterprise.*"

A partir de ella, veamos un ejemplo de cada tipo de tokenización:

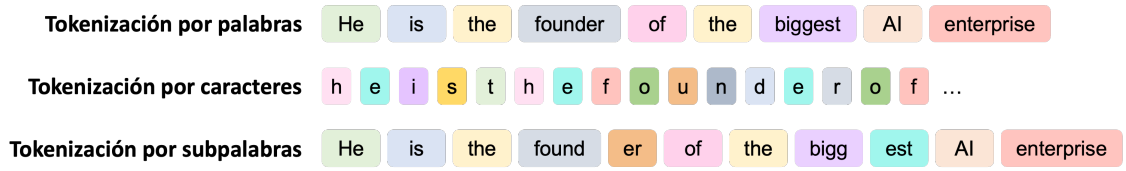


Figura 2.1: División por tokens

Una vez estos tokens se han identificado, se les puede asignar un número entero a cada uno para que un ordenador pueda diferenciarlos. Sin embargo, esto puede generar un problema. Si el ordenador trata estos valores identificativos como números enteros, en el ejemplo anterior se podría decir, por ejemplo, que “is” ($id = 55$) es más grande que “he” ($id = 23$). Técnicamente esto no es así, pues realmente son palabras y, a priori, unas no tienen valores mayores o menores que otras.

Para resolver este problema, se debe pensar en una forma de identificación diferente. Una de esas posibles soluciones podría ser generar un vector de tamaño N (donde N es el número total de palabras del vocabulario) para cada palabra en el que sus posiciones hacen referencia a cada una de las palabras del vocabulario. Así, cada una se podrá representar poniendo a 1 su posición en el vector. De esta forma, todas tendrán el mismo valor y “ninguna será mayor que otra”, pues geométricamente todas están a la misma distancia. A esta representación se le denomina **One-Hot encoding**.

Sin embargo, esta representación también lleva consigo un problema, el espacio. Imaginemos que tenemos una frase de M palabras y un vocabulario de N palabras en total. Si para cada palabra tenemos que generar un vector de tamaño N, estaríamos generando una matriz de $M \times N$. Si tenemos un tamaño de vocabulario amplio, esto acabaría generando un consumo de espacio muy grande.

Además, en esta representación, como se ha comentado antes, la distancia entre palabras es exactamente la misma. Es decir, por ejemplo, las palabras “ordenador” y “botella” tienen el mismo valor. Pero ¿no es cierto que, por ejemplo, “teclado” y “ordenador” son más afines conceptualmente que “teclado” y “botella”? ¿Podrían estar más cerca unas palabras de otras en el espacio del vocabulario? Aquí es donde entra el concepto de **Word embeddings**.

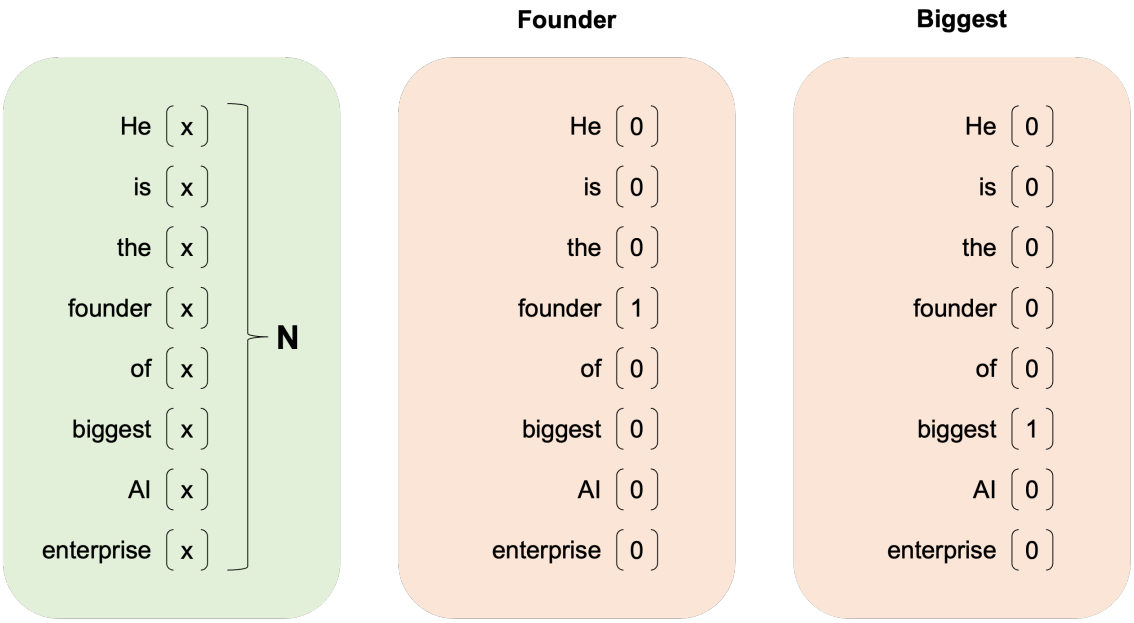


Figura 2.2: Representación One-Hot encoding

2.1.2. Word embeddings

Word embedding es un tipo de representación que permite a las palabras con un sentido similar tener una representación vectorial parecida, y **siempre con el mismo tamaño de vector**. A diferencia de la representación One-Hot encoding, estos vectores tienen un tamaño mucho menor, pues no es necesario tener un vector tan grande como el tamaño del vocabulario para representar cada una de las palabras.

Además, los valores de estos vectores generados con word embeddings tienen en cuenta la semántica de las frases [1]. Esto permite juntar o separar vectores geométricamente según los distintos textos procesados y su sentido gramatical. Por tanto, las palabras que se usan en contextos similares acabarán teniendo una representación también similar.

Cuanto más texto procese el algoritmo de NLP, mayor cantidad de datos se podrán recabar y, por tanto, el análisis puede ser más preciso, pudiendo incluso llegar a detectar ironía o sarcasmo [2].

2.1.3. Librería NLTK

Para evitar tener que hacer todo este procesamiento del lenguaje en el desarrollo del proyecto, se ha decidido apoyarse en el uso de la librería NLTK de Python, la cual nos ayudará con la tokenización, el análisis del sentimiento y el etiquetado de los tokens.

NLTK [3] (Natural Language Toolkit o Kit de Herramientas de Lenguaje Natural en español) es una librería de Python que se utiliza para ayudar en el desarrollo de programas que trabajan con

datos escritos en lenguaje natural. Ofrece múltiples bibliotecas de procesamiento de texto para la clasificación, tokenización, análisis de sentimiento y etiquetado, entre otros. A continuación, se van a explicar algunos de estos módulos.

Análisis del sentimiento de una frase

NLTK dispone de un analizador previamente entrenado llamado VADER (*Valence Aware Dictionary and Sentiment Reasoner*), el cual se puede utilizar mediante el módulo ***SentimentIntensityAnalyzer***. Al estar entrenado previamente, se pueden conseguir resultados más rápidamente que con otros analizadores [4].

Para poder analizar un texto escrito en lenguaje natural, se debe crear una instancia de la clase *nltk.sentiment.SentimentIntensityAnalyzer* y ejecutar el método *polarity_scores()* del objeto instanciado. Esto generará como salida un diccionario de Python con los siguientes valores:

- **'neg'**: porcentaje de negatividad en la frase. Sus valores se comprenden entre 0 y 1.
- **'neu'**: porcentaje de neutralidad en la frase. Sus valores se comprenden entre 0 y 1.
- **'pos'**: porcentaje de positividad en la frase. Sus valores se comprenden entre 0 y 1.
- **'compound'**: cálculo ponderado del sentimiento general de la frase. Sus valores se comprenden entre -1 y 1, siendo 1 el sentimiento más positivo y -1 el negativo.

En este estudio se ha decidido usar 'compound' como valor clave para determinar si el sentimiento general de la frase es positivo o negativo.

Tokenización

Además de analizar el sentimiento, NLTK también permite tokenizar una frase. Esto se consigue gracias a la función *nltk.word_tokenize()*, la cual devuelve una lista con los tokens generados a partir de un texto.

Etiquetado

A su vez, este módulo también tiene la capacidad de etiquetar los tokens gracias a haber sido previamente entrenado. Esto se consigue con la función *nltk.pos_tag()*, la cual genera una lista de pares (token,tag) en la que cada token tiene asociada una etiqueta. Estas etiquetas determinan el tipo de dato referente a cada token: adverbio, sustantivo, verbo,..., según el conjunto de etiquetas Penn Treebank [5].

2.2. Sistemas de recomendación

Con el auge de las nuevas tecnologías y el uso de las redes sociales, entre otros factores, en los últimos años se ha ido generando un aumento masivo de datos e información. Esta información, si

es bien tratada, puede llegar a ser de gran utilidad, tanto para las empresas como para los usuarios finales.

Entre muchas de estas aplicaciones se encuentran los sistemas de recomendación. Los **sistemas de recomendación** son algoritmos utilizados para sugerir a usuarios ciertos elementos relevantes, ya sean estos elementos películas, productos que comprar o libros que leer [6]. Tratan volúmenes de información muy grandes y filtran lo que, según el algoritmo concreto, consideran más relevante para el usuario.

Estos pueden ser útiles tanto para los usuarios finales, para ayudarles a encontrar más rápidamente elementos acordes a sus preferencias, como para empresas, en los que son útiles para, por ejemplo, determinar qué producto puede necesitar en ese momento.

Existen múltiples tipos de sistemas de recomendación. Desde los sistemas de recomendación basados en popularidad o los modelos de clasificación, hasta los basados en contenido y los de filtrado colaborativo, pasando por algunos sistemas híbridos que nos ofrecen funcionalidad de varios tipos. Este estudio se centrará en los tres últimos: sistemas de recomendación de filtrado colaborativo, sistemas de recomendación basados en contenido y sistemas de recomendación híbridos, además de añadir también los **sistemas de recomendación basados en aspectos**, siendo estos un tipo de sistema de recomendación híbrido que combina características tanto de los sistemas de recomendación basados en contenido como de los de filtrado colaborativo.

2.2.1. Sistemas de recomendación basados en contenido

Los **sistemas de recomendación basados en contenido** dan importancia al análisis de los atributos de los usuarios e ítems de cara a generar las recomendaciones. Si se considera el estudio actual como ejemplo, esta información extra podría ser el tipo de restaurante o el tipo de comida que ofrece para los ítems y el sexo, edad o cualquier otra información personal para los usuarios.

La idea principal de los métodos basados en contenido es tratar de crear un modelo que explique las interacciones usuario-ítem basándose en sus características principales. Así, considerando el mismo ejemplo de antes, podríamos crear un modelo basándonos en el hecho de que los jóvenes menores de 30 años tienden a calificar mejor un tipo de restaurantes, las mujeres otro tipo, y así. De esta manera, el sistema de recomendación, a partir de las características de ítems y usuarios, podrá recomendar unos elementos u otros.

Si tratamos de crear este modelo, producir predicciones para un nuevo usuario puede ser muy fácil: simplemente se deberá tener en cuenta su perfil (edad, sexo, ...) para así determinar los ítems que sean más relevantes para él.

Este tipo de métodos basados en contenido sufren menos el llamado “cold start problem” [1] [7], pues los nuevos usuarios se pueden describir por sus características y, a partir de ellas, se pueden crear nuevas recomendaciones.

2.2.2. Sistemas de recomendación de filtrado colaborativo

A diferencia de los basados en contenido, los **sistemas de recomendación de filtrado colaborativo** se basan únicamente en las interacciones pasadas registradas entre usuarios e ítems con el fin de producir nuevas recomendaciones.

En este estudio esas interacciones serán los ratings que los usuarios ponen a los restaurantes. Estas interacciones se almacenan en la denominada “matriz de interacciones usuario-ítem”.

	Ítem 1	**Ítem 2**	**Ítem 3**
Usuario 1	2		5
Usuario 2	2	2	
Usuario 3	4		3
Usuario 4		1	2

Tabla 2.1: Ejemplo de matriz de interacciones usuario-ítem

La idea principal de los métodos de filtrado colaborativo es que haya suficientes interacciones como para detectar usuarios o ítems similares y hacer predicciones basadas en estas proximidades.

La principal ventaja de estos algoritmos es que no requiere de ninguna información identificativa sobre los usuarios o ítems, por lo que pueden utilizarse en muchos ámbitos en los que esa información no está disponible. En este caso, cuantos más usuarios interactúen con los diferentes ítems, más precisas serán las recomendaciones.

Sin embargo, considerar únicamente las interacciones previamente registradas para hacer recomendaciones hace que estos modelos sufran el denominado “cold start problem” [7]. Es imposible recomendar algo a usuarios nuevos o recomendar un nuevo ítem a los usuarios, ya que estos no tienen interacciones previamente registradas. A su vez, también es muy difícil manejar correctamente recomendaciones de usuarios o ítems que tengan pocas interacciones.

2.2.3. Sistemas de recomendación híbridos

Los **sistemas de recomendación híbridos** combinan diferentes técnicas de recomendación de otros sistemas, tratan de paliar las posibles limitaciones e intentan enfocarse en las ventajas que ofrece

[1] El “cold start problem” (o “arranque en frío” en español) ocurre cuando un sistema se ve incapaz de generar una relación entre usuario e ítem por falta de información y, por tanto, no puede recomendar algo relevante al usuario.

cada uno. La idea principal de estos sistemas es combinar los diferentes algoritmos de modo que puedan generar recomendaciones más precisas.

2.3. Sistemas de recomendación basados en aspectos

Los **sistemas de recomendación basados en aspectos** son un tipo de sistemas de recomendación híbridos que buscan explotar las opiniones que los usuarios realizan sobre los ítems. Esto se hace extrayendo los atributos de esas opiniones, de manera que estos atributos, junto con el rating establecido por el usuario, serán los que posteriormente se utilizarán en el algoritmo de recomendación.

En el caso de este estudio, se quieren explotar las reseñas que los usuarios han escrito en los restaurantes. Pongamos un ejemplo: un usuario ha realizado una reseña en un restaurante (al cual le ha puesto una calificación de 3 sobre 5) en la que pone lo siguiente:

"El trato del personal ha sido adecuado y el sonido ambiente era muy bueno. Sin embargo, la calidad de la comida, aunque es buena, no va acorde con el precio. A pesar de ello, es posible que repitamos porque hemos estado muy a gusto."

De esta reseña se puede determinar que el usuario parece que da importancia tanto al personal y el ambiente como a la calidad de la comida.

De este modo, incluso cuando los usuarios han evaluado un restaurante con el mismo rating, estos sistemas de recomendación son capaces de captar puntos fuertes y débiles exclusivos de usuarios e ítems a partir de la reseña y tratar de recomendar de una manera óptima.

Por ejemplo, estos sistemas pueden determinar qué tiene más importancia para cada usuario y darle más énfasis a la hora de recomendar. Para ello, se puede dividir el proceso en: (1) extracción de los aspectos relevantes a partir de la reseña, (2) identificación de modificadores del aspecto y (3) recomendación basada en aspectos [8].

Se explicarán mas detalladamente los algoritmos en el capítulo 3 de este estudio.

2.4. Métricas de evaluación

¿Cómo sabemos si un sistema de recomendación funciona bien? ¿Cómo podemos cuantificar y comparar qué sistema funciona mejor? Para responder a estas preguntas se nos genera la necesidad de evaluar.

Una forma sencilla de ver la satisfacción de un usuario es ver si las recomendaciones proporcionadas son útiles o no. Es decir, si se han recomendado ítems relevantes a sus preferencias.

La **relevancia** es un concepto central en el ámbito de la recuperación de información. Una recomendación será relevante si satisface la necesidad del usuario. Generalmente suele ser una calificación binaria; es decir, o es relevante o no lo es. Aunque hoy en día se estudia la posibilidad de introducir distintos grados de relevancia y ser menos estricto. En este estudio se seguirá el enfoque binario.

Para poder determinar cuán de efectivo es cada uno de los sistemas de recomendación que se van a implementar, se van a desarrollar dos métricas: Precisión y Recall.

En este estudio, una recomendación para un usuario determinado será relevante si, tras haber hecho una división del conjunto de datos en entrenamiento-test, ese ítem, recomendado a partir del conjunto de entrenamiento, se encuentra en el conjunto de test de ese usuario.

2.4.1. Precisión

Representa una tarea donde el usuario quiere encontrar un número razonable de recomendaciones relevantes y valora la ratio de recomendaciones relevantes por unidad de esfuerzo.

Se puede calcular en forma de True Positive (recomendaciones que son relevantes y se han devuelto), False Positive (recomendaciones que no son relevantes, pero se han devuelto).

$$P = \frac{|Relevantes \cap Devueltos|}{|Devueltos|} = \frac{TP}{TP + FP} \tag{2.1}$$

También es muy normal limitar la métrica para no tener en cuenta todas las recomendaciones y sobrecargar la evaluación, de manera que únicamente se tengan en cuenta las k primeras, haciendo así una estimación de cómo está comportándose el algoritmo sin necesidad de considerar todas las recomendaciones. A esto se le denomina "precisión at k"($P@k$).

2.4.2. Recall

Representa una tarea donde el usuario quiere encontrar todas las recomendaciones relevantes y no repara en el esfuerzo de examinar todas las irrelevantes.

También se puede calcular de la misma forma que anteriormente, añadiendo aquí los False Negatives (recomendaciones que no se han realizado porque se creía que no eran relevantes pero que realmente sí lo son).

$$P = \frac{|Relevantes \cap Devueltos|}{|Relevantes|} = \frac{TP}{TP + FN} \tag{2.2}$$

3

Diseño e implementación

En este capítulo se dará paso a una explicación del diseño de este proyecto, así como a los requisitos funcionales y no funcionales que se deben cumplir. Además, también se dará en detalle una explicación de cada uno de los módulos que componen el proyecto.

3.1. Diseño

En esta sección se verá el diseño del proyecto, los módulos que lo componen y la relación que tienen unos sobre otros.

3.1.1. Estructura general

Como se puede ver en la figura 3.1, el proyecto está dividido en cuatro módulos. Éstos son:

- **Módulo de preprocesamiento de datos**: es el encargado de filtrar del conjunto de datos original de Yelp las reseñas referidas únicamente a establecimientos que son restaurantes. Además, también permite generalizar el proyecto, de manera que, si se quiere usar otro conjunto de datos, este será el encargado de transformarlos a un mismo formato para que los demás módulos no se vean afectados. A su vez, este módulo, además de crear el conjunto de datos a procesar, también genera otros conjuntos de datos de prueba de menos registros, útiles para trabajar poco a poco en el desarrollo.
- **Módulo de procesamiento de datos**: módulo necesario para procesar cada una de las reseñas extraídas del conjunto generado en el preprocesamiento. En él se hace uso de la librería NLTK, detallada en la sección 2.1.3, para generar el conjunto de datos con el que se harán las recomendaciones. Además, aquí se ha tratado de optimizar el tiempo de ejecución con el uso de varios procesos en paralelo.
- **Módulo de sistemas de recomendación**: es el módulo encargado de generar las recomendaciones a usuarios a partir de los registros previamente generados. En él se implementan los algoritmos de recomendación explicados en la sección 2.2.

- **Módulo de evaluación**: módulo necesario para evaluar las recomendaciones obtenidas para cada uno de los usuarios. Aquí se implementarán las métricas comentadas en la sección 2.4.

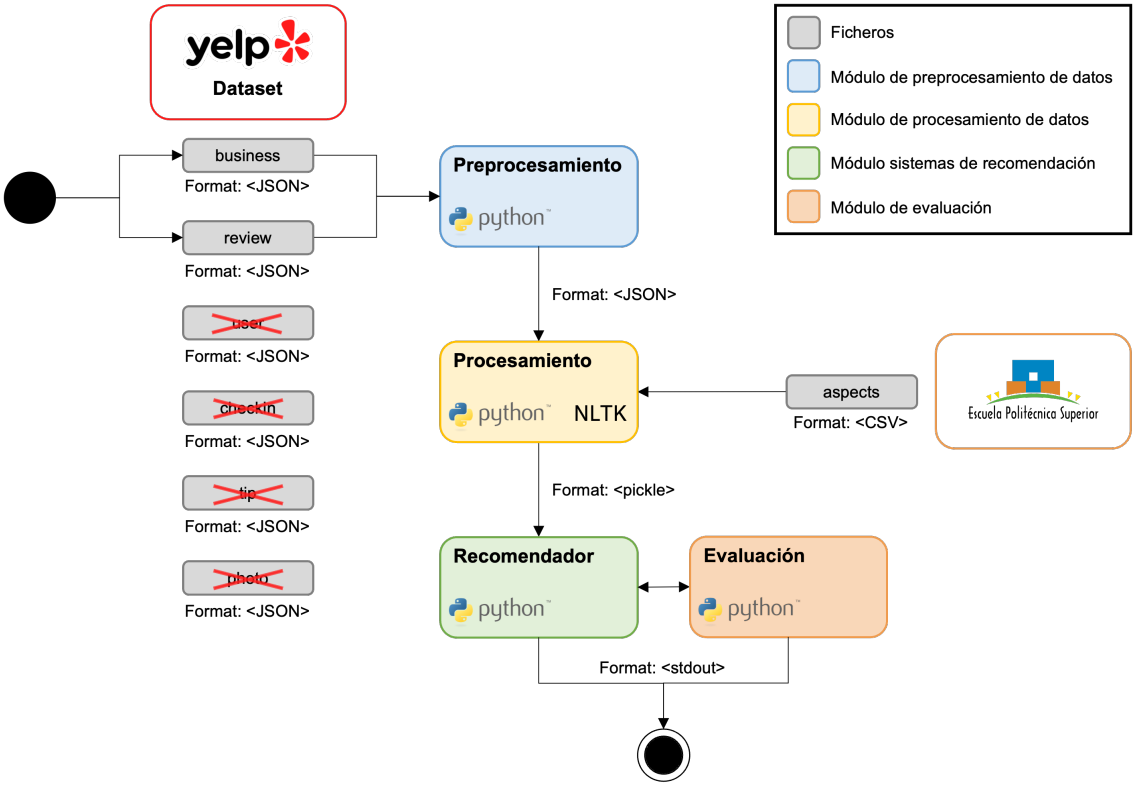


Figura 3.1: Diseño de arquitectura

3.2. Requisitos

Veamos a continuación en detalle los requisitos funcionales y no funcionales impuestos para el funcionamiento de cada uno de los módulos.

3.2.1. Requisitos funcionales

Preprocesamiento

RF-1.– Los datos de entrada siempre serán los del conjunto de datos de Yelp.

RF-2.– El conjunto de datos de entrada debe estar en formato JSON.

RF-3.– El conjunto de datos de entrada se precargará en RAM antes de comenzar la ejecución.

RF-4.– Los identificadores de usuarios y restaurantes deben ser únicos en el conjunto global.

RF-5.– El conjunto de datos de salida debe generarse en formato JSON.

RF-6.– Se deben generar tanto el conjunto de datos final como otros dos conjuntos de datos de pruebas con diferente número de registros cada uno, como mínimo.

RF-7.– El conjunto de datos de salida debe contener únicamente información relevante a los restaurantes.

Procesamiento

RF-8.– El conjunto de datos de entrada debe estar en el idioma inglés para asegurar el correcto funcionamiento en el procesamiento del lenguaje natural.

RF-9.– Es necesario un conjunto de datos de entrada relacionado con los aspectos de los restaurantes en formato (clave, valor).

RF-10.– Los datos de entrada siempre serán los generados por el módulo de preprocesamiento más el conjunto de datos de los aspectos.

RF-11.– El conjunto de datos de salida se persistirá en formato pickle y, opcionalmente, en formato JSON para que se pueda tener acceso a la información en texto plano.

RF-12.– Para mejorar el tiempo de ejecución, se deberá paralelizar el procesamiento en tantos procesos como núcleos tenga el sistema que lo ejecute.

Recomendación

RF-13.– El conjunto de datos de entrada será el fichero en formato pickle generado por el módulo de procesamiento. En ningún caso puede ser otro.

RF-14.– El conjunto de datos se precargará en RAM antes de comenzar la ejecución.

RF-15.– El usuario que ejecute el programa no tendrá interacción con el mismo.

RF-16.– El usuario podrá configurar, aunque de forma limitada, algunos parámetros específicos de los sistemas de recomendación.

RF-17.– Se limitará el número de recomendaciones a generar para no saturar el sistema.

RF-18.– Se implementarán, como mínimo, cuatro sistemas de recomendación diferentes.

Evaluación

RF-19.– Se hará una evaluación de los sistemas de recomendación con las métricas de evaluación Precisión y Recall, explicadas anteriormente.

RF-20.– Se evaluarán los sistemas del módulo de recomendación de acuerdo con una partición del conjunto de datos de entrada.

RF-21.– Opcionalmente, se implementará la posibilidad de permitir hacer una evaluación con una muestra de los usuarios para hacer pruebas más rápidamente.

RF-22.– Los resultados de la evaluación se mostrarán por la salida estándar del terminal.

3.2.2. Requisitos no funcionales

RNF-1.– El proyecto estará completamente desarrollado en el lenguaje de programación Python.

RNF-2.– La interfaz de usuario será una terminal (o Shell) donde poder ejecutar el programa Python.

RNF-3.– De acuerdo con la Ley Orgánica 3/2018, de Protección de Datos Personales y garantía de los derechos digitales [9], todos los usuarios y restaurantes del conjunto de datos tendrán un identificador, pero no se podrá acceder a su información detallada, como el nombre u otra información sensible.

RNF-4.– Las funcionalidades estarán separadas en diferentes módulos de Python, de manera que haya modularidad y se puedan ejecutar por separado.

RNF-5.– La salida de la ejecución será sencillamente entendible por el usuario, sin necesidad de tener que seguir una explicación para comprenderla.

RNF-6.– La ejecución del programa será independiente de la plataforma que se esté utilizando, siempre y cuando se tenga instalado el intérprete de Python 3 o superior y las librerías necesarias en el entorno de ejecución.

3.3. Implementación

En esta sección se hablará con detalle de cada uno de los módulos desarrollados en el proyecto. Primero se hablará de a qué se dedica cada uno, para después explicar cómo ha sido su desarrollo, así como la explicación de algunas decisiones de implementación.

3.3.1. Pre-procesamiento

Antes de comenzar con el desarrollo, primero se tiene que ver de dónde se van a sacar los datos, para después poder hacer un análisis y procesamiento de estos.

Desde un principio se ha tenido claro utilizar el conjunto de datos proporcionado por la empresa Yelp, red social que permite compartir las experiencias de los usuarios en todo tipo de establecimientos, permitiendo escribir reseñas y puntuándolos de 1 a 5.

Yelp guarda estas evaluaciones de los usuarios públicamente en un conjunto de datos. Para poder tener acceso a él, es necesario rellenar un formulario con el nombre, email e iniciales para que la empresa se asegure de que se va a hacer un correcto uso de ellos. Una vez se rellena esto, tras unos días se recibe el conjunto de datos por correo, el cual ya se podrá empezar a utilizar.

Como primer análisis, se hace un estudio de qué contiene realmente el conjunto de datos. Como se puede observar en [10], éste está compuesto de varios ficheros en formato JSON. Estos son:

- Conjunto de datos con la información asociada a los **establecimientos**.
- Conjunto de datos con la información relevante a las **reseñas**.
- Conjunto de datos con la información relevante a los **usuarios**.
- Otros conjuntos de datos relacionados con **checkins**, **tips** y **fotos** de cada establecimiento.

Para este estudio se van a utilizar únicamente los conjuntos de datos de los establecimientos y las reseñas. Los demás no serán necesarios, pues la información que aportan no es relevante en este trabajo de investigación.

En relación con el conjunto de datos de los **establecimientos**, se debe tener en cuenta el atributo "categories". Como Yelp es un conjunto de datos que contiene información de todo tipo de establecimientos – hoteles, salones de belleza, clínicas veterinarias, ... – éste se debe filtrar para únicamente guardar los necesarios para este estudio, los restaurantes. Es en este atributo donde nos fijaremos para saber si un establecimiento es un restaurante o no.

Así, una vez se tiene una lista de los establecimientos que son restaurantes, podremos filtrar el dataset de **reseñas** y quedarnos únicamente con aquellas que tengan asociado un identificador de establecimiento asociado a un restaurante.

Por otra parte, una vez se ha hecho este filtrado en las reseñas, se tiene que seguir filtrando el conjunto de datos, pues aun tenemos usuarios y restaurantes con pocas reseñas asociadas a ellos. Para paliar este problema, lo que hacemos es filtrar todos aquellos usuarios y restaurantes que, entre ellos, tengan más de un número determinado de reseñas. Es decir, un usuario debe haber hecho, como mínimo, un número N de reseñas a restaurantes que también tengan ese mínimo N con otros usuarios.

Como el equipo de pruebas (tabla 4.1) donde ejecutamos los módulos está limitado por la RAM, se ha decidido establecer tanto un mínimo como un máximo de reseñas para cada usuario e ítem. Así, nos quedaremos únicamente con los usuarios y los restaurantes que tengan asociadas 30 o más reseñas, y menos de 100, evitando así tener problemas de memoria en las posteriores ejecuciones de los demás módulos.

Para implementar este filtrado, se ha desarrollado un módulo en Python, el cual (1) carga en memoria los conjuntos de datos de Yelp, (2) filtra el dataset de establecimientos quedándose únicamente con los que tienen como categoría "Restaurants", (3) filtra las reseñas, quedándose únicamente con las asociadas a restaurantes, (4) filtra de nuevo el conjunto de datos de las reseñas limitandolo a un mínimo de 30 y un máximo de 100 por usuario e ítem y (5) guarda estas reseñas en un nuevo fichero JSON. De esta forma este script solo se tendrá que ejecutar una vez, guardándose estos filtros en un fichero en formato JSON de manera persistente.

Además, también se generan otros ficheros de menos registros, con los cuales se pretenden hacer pruebas a medida que se van desarrollando los siguientes módulos.

A continuación, se muestran los detalles de la implementación:

1.– Primero, se cargan en memoria los conjuntos de datos de establecimientos, y reseñas, y se eliminan las columnas innecesarias para que ocupen menos memoria.

2.– Se filtran los establecimientos que contengan la categoría “Restaurants”.

3.– Se filtran únicamente las reseñas que contengan los identificadores de los restaurantes.

4.– Limita el conjunto de datos de las reseñas a usuarios e ítems que tengan más de 30 y menos de 100 reseñas asociadas. Este paso se realiza por potenciales problemas de memoria en los futuros módulos.

5.– Por último, se genera tanto el conjunto de datos final como los conjuntos de datos de pruebas.

3.3.2. Procesamiento

En este apartado se va a explicar cómo ha sido el análisis y procesamiento de los datos con los que luego se trabajará en el módulo de recomendación.

El conjunto de datos de entrada con el que nos encontramos es un fichero en formato JSON con la información correspondiente en la tabla 3.1 por cada uno de los registros.

Review Dataset

review_id	Identificador de la reseña. P. ej. "*jCPO8haIMpAYyRbOkE1O7w*"
user_id	Identificador del usuario que ha realizado la reseña. P. ej. "*YK0Ie74a2fsGqpCV-XzRjw*"
business_id	Identificador del establecimiento al que hace referencia la reseña. P. ej. "*3LWsVfsSmb_Nzbi2YQ-NIA*"
stars	Entero entre 1 y 5 que indica el grado de satisfacción del cliente. P. ej. *3*
text	Reseña del usuario sobre el establecimiento. P. ej. "*I’ve eaten here a few times and I love the food and the service!*"

Tabla 3.1: Campos generados en el preprocesamiento

Además, puesto que lo que queremos es explotar los aspectos de las reseñas, también contaremos con un fichero proporcionado por el equipo de investigación del tutor, el cual asocia palabras de la jerga de los restaurantes con ciertos aspectos. Así, por ejemplo, “tarta” y “tiramisú” pueden ir asociados con el aspecto “postres”. Destacar que este fichero fue generado en un trabajo de investigación previo, con un conjunto de datos de Yelp más antiguo [8]. A pesar de haberse hecho hace tiempo, el fichero de aspectos [11] sigue siendo provechoso, pues el vocabulario relacionado con este sector no ha

cambiado a lo largo del tiempo. Sin embargo, al haber reseñas nuevas, la asignación de aspectos se debe hacer desde cero, y es lo que se explica a continuación.

Este fichero proporcionado simplificará el número total de aspectos que tendremos, pues así se consigue una mejor y más simplificada estructuración. Se encuentra en formato CSV y está compuesto de pares de atributos clave-valor en los que la clave es el aspecto y el valor es el término asociado con ese aspecto, como vemos en la tabla 3.2.

Aspects Dataset

aspect	Aspecto relacionado con los restaurantes. P. ej. "*booking*"
term	Término asociado al aspecto. P. ej. "*reservation*"

Tabla 3.2: Campos del conjunto de datos de aspectos

El objetivo de este procesamiento, por tanto, es generar un conjunto de datos final a partir de todas las reseñas de los usuarios, cuyos registros sigan la estructura de la tabla 3.3.

Annotations Dataset

review_id	Identificador de la reseña. P. ej. "*jCPO8haIMpAYyRbOkE1O7w*"
user_id	Identificador del usuario que ha realizado la reseña. P. ej. "*YK0Ie74a2fsGqpCV-XzRjw*"
restaurant_id	Identificador del restaurante al que hace referencia la reseña. P. ej. "*3LWsVfsSmb_Nzbi2YQ-NIA*"
rate	Entero entre 1 y 5 que indica el grado de satisfacción del cliente. P. ej. *3*
term	Término extraído de la reseña. P. ej. "*entrees*"
aspect	Aspecto asociado al término analizado. P. ej. "*appetizers*"
feeling	Booleano que denota si el sentido de una frase es positivo (1) o negativo (-1). P. ej. *-1*

Tabla 3.3: Campos del conjunto de datos final

Para generar todas las entradas de este conjunto de datos primero se han cargado en memoria los datos generados en el preprocesamiento, así como el conjunto de datos de los aspectos. Una vez tenemos esto cargado en memoria, se utiliza el módulo NLTK de Python para procesar todas las reseñas y sacar cada uno de los registros del conjunto final.

Destacar que todo este procesamiento tardaría bastante tiempo en ejecutarse debido a la gran cantidad de datos que hay. Es por ello por lo que, atendiendo a los requisitos, se ha hecho este procesamiento en paralelo, con tantos hilos como número total de núcleos tenga el sistema en que se ejecute. Así, el tiempo total de ejecución se reduce considerablemente.

Veamos a continuación cómo es el proceso a seguir en cada uno de los hilos de ejecución. Por cada registro del conjunto de datos de entrada, se extrae la reseña para procesarla. Los pasos que se siguen en su procesamiento son los siguientes:

1.– Se divide la reseña en **frases** con la función *nltk.sent_tokenize()*.

2.– Para cada una de esas frases, se determina el **sentimiento general** – positivo o negativo – con la clase *SentimentIntensityAnalyzer*. Si el valor de 'compound' es mayor que 0, el atributo 'feeling' será 1. Por el contrario, si es menor o igual que 0, el valor de 'feeling' será -1.

3.– Además, como también queremos extraer los aspectos de la frase, generaremos una entrada por cada término asociado a un aspecto de los restaurantes. Por tanto, tendremos que extraer también todos los sustantivos de la frase que tengan que ver con el vocabulario de los restaurantes. Para ello nos ayudamos de la función *nltk.pos_tag()* de NLTK. En este estudio nos queremos quedar únicamente con los sustantivos. Por ello, nos quedamos con los tokens cuya etiqueta comience por 'N' [5] y se encuentren dentro del conjunto de datos de los aspectos.

4.– Una vez tenemos todos esos términos, añadimos un registro al conjunto de datos final por cada uno de ellos, generando así tantos registros como términos se hayan encontrado en cada una de las frases de las reseñas.

Cuando los hilos han ejecutado sus reseñas correspondientes, se guardan todos los registros generados en una tabla (concretamente en un dataframe de Pandas), que persistiremos en formato pickle y, opcionalmente JSON, para la posterior carga en el módulo de recomendación.

3.3.3. Sistemas de recomendación y evaluación

Cuando ya tenemos el conjunto de datos final podemos entonces comenzar a implementar los sistemas de recomendación. En esta sección se explicarán los módulos de **recomendación** y **evaluación**, así como la implementación seguida para cada uno de ellos. Pero antes de entrar en ello, veamos cómo se van a representar los objetos. Como se hizo en su momento en [8], las representaciones son las siguientes:

Usuarios e ítems

De aquí en adelante un usuario va a estar representado como un vector

$$u_m = \{w_{m,1}, w_{m,2}, w_{m,3}, \ldots, w_{m,K}\}$$

donde $w_{m,k}$ puede ser:

1.– El peso que le da el usuario m al aspecto a_k, siendo K el número total de aspectos.

2.– El rating que le ha puesto el usuario m al restaurante i_k, siendo K el número total de restaurantes.

De la misma forma, los restaurantes estarán representados como un vector

$$i_n = \{w_{n,1}, w_{n,2}, w_{n,3}, \ldots, w_{n,K}\}$$

donde $w_{n,k}$ puede ser:

1.– El peso que le da el restaurante n al aspecto a_k, siendo K el número total de aspectos.

2.– El rating del restaurante n asociado al usuario u_k, siendo K el número total de usuarios.

Funciones de similitud

La función de similitud que se va a estudiar en este trabajo será la **función coseno**. De esta manera, podremos tener cuatro funciones de similitud diferentes, una por cada representación de cada objeto. Así, estas serían las representaciones de cada una de ellas:

- **Similitud de usuario** con los dos posibles valores de $w_{m,k}$.

$$sim(u,v) \equiv cos(u,v) = \frac{\sum_{w_{u,k}\neq\emptyset \wedge w_{v,k}\neq\emptyset} w_{u,k} w_{v,k}}{\sqrt{\sum_{w_{u,k}\neq\emptyset} w_{u,k}^2 \sum_{w_{v,k}\neq\emptyset} w_{v,k}^2}} \tag{3.1}$$

- **Similitud de ítem** con los dos posibles valores de $w_{n,k}$.

$$sim(i,j) \equiv cos(i,j) = \frac{\sum_{w_{i,k}\neq\emptyset \wedge w_{j,k}\neq\emptyset} w_{i,k} w_{j,k}}{\sqrt{\sum_{w_{i,k}\neq\emptyset} w_{i,k}^2 \sum_{w_{j,k}\neq\emptyset} w_{j,k}^2}} \tag{3.2}$$

A partir de estas similitudes se podrá después determinar cuál es la que funciona mejor recomendando para el conjunto de datos con el que trabajamos.

Recomendadores

Para recomendar restaurantes a usuarios se han desarrollado tres recomendadores diferentes: un recomendador básico basado en contenido, y otros dos recomendadores híbridos que recomiendan de acuerdo a la función de similitud que se les pase como argumento.

$$\hat{r}(u,i) \equiv cos(u,i) = \frac{\sum_{k\in common\_aspects} w_{u,k} w_{i,k}}{\sqrt{\sum_{w_{u,k}\neq\emptyset} w_{u,k}^2 \sum_{w_{i,k}\neq\emptyset} w_{i,k}^2}} \tag{3.3}$$

$$\hat{r}(u,i) = \sum_{v \in N_k(u)} sim(u,v) r(v,i) \tag{3.4}$$

$$\hat{r}(u,i) = \sum_{j \in N_k(i)} sim(i,j) r(u,j) \tag{3.5}$$

Así, tendríamos finalmente los siguientes sistemas de recomendación.

- Sistema de recomendación basado en contenido (**cb**).
- Sistema de recomendación híbrido con similitud de usuario basada en ratings (**ub**).
- Sistema de recomendación híbrido con similitud de usuario basada en aspectos (**cbub**).
- Sistema de recomendación híbrido con similitud de ítem basada en ratings (**ib**).
- Sistema de recomendación híbrido con similitud de ítem basada en aspectos (**cbib**).

Cada uno de estos recomendadores creará un ranking de top-n recomendaciones y serán estas las que se validarán en las posteriores métricas de evaluación.

Una vez hemos explicado las representaciones de cada uno de los objetos necesarios para llevar a cabo el proyecto, veamos cómo han sido implementados los sistemas de recomendación. Para desarrollarlos, se han implementado varias clases que serán de ayuda para modular las distintas funcionalidades. Así, se contará con las siguientes clases: ratings, similitudes, ranking y recomendadores. Veamos cada una de ellas por separado:

Ratings

Tiene como principal objetivo calcular y guardar ponderadamente los scores por cada uno de los aspectos en función del rating y del sentimiento de cada registro, tanto de los usuarios como de los restaurantes. Este cálculo tiene la ventaja de que **se puede ajustar** si los resultados de la recomendación no son los esperados por el usuario. De esta manera, se tendrían infinidad de posibilidades de generar estos valores hasta dar con un resultado adecuado.

Además, esta clase también guardará todos los ratings de cada uno de los usuarios y restaurantes. Es decir, generará lo que llamamos anteriormente los vectores de usuarios e ítems, siendo estos ítems los restaurantes.

La forma de guardar estos valores es en diccionarios de Python, los cuales funcionan como un hash, de manera que el acceso a ellos es eficiente. Por último, esta clase también será la encargada de guardar todos los identificadores únicos de usuarios y restaurantes del conjunto de datos.

Los cálculos de los que hablamos se realizan en el constructor de la clase. Así, una vez instanciada, se puede acceder a cualquiera de los valores calculados gracias a los siguientes métodos:

- $ratings(object)$: devuelve, en función del parámetro, todos los ratings asociados a un usua-

rio o a un restaurante concreto.

- $user\_rating(user, restaurant)$ y $restaurant\_rating(restaurant, user)$: devuelven el rating que ha puesto un usuario a un restaurante o el puesto a un restaurante por un usuario. Son dos formas distintas de devolver el mismo resultado, pero con los parámetros de entrada cambiados de orden.
- $aspects(obj)$: devuelve, en función del parámetro, los scores ponderados de todos los aspectos asociados con un usuario o con un restaurante concretos.
- $aspect\_user\_weight(user, aspect)$ y $aspect\_restaurant\_weight(restaurant, aspect)$: devuelve el peso que le da un usuario o un restaurante a un aspecto, respectivamente.
- $users()$: devuelve una lista con todos los identificadores de los usuarios.
- $restaurants()$: devuelve una lista con todos los identificadores de los restaurantes.

Estos métodos serán utilizados por algunas de las clases que explicaremos a continuación.

Similitudes

Esta parte del módulo de recomendación se encarga de implementar las funciones de similitud que luego podremos utilizar en los recomendadores. Son de especial utilidad ya que en función de estas similitudes el algoritmo final de recomendación puede comportarse de una manera u otra, generando recomendaciones diferentes.

Para implementar las similitudes se ha creado primero una clase abstracta, la cual recibe como argumento los ratings generados en un paso previo, y cuenta con un método, también abstracto, que será el encargado de calcular las similitudes entre dos objetos.

Las clases implementadas son:

- *CosineUserSimilarityRatings*: calcula la similitud coseno entre dos usuarios a partir de los ratings.
- *CosineUserSimilarityAspects*: calcula la similitud coseno entre dos usuarios a partir de los pesos de los aspectos.
- *CosineRestaurantSimilarityRatings*: calcula la similitud coseno entre dos ítems a partir de los ratings.
- *CosineRestaurantSimilarityAspects*: calcula la similitud coseno entre dos ítems a partir de los pesos de los aspectos.

Teniendo estas cuatro similitudes diferentes se podrá después determinar cuál es la más óptima recomendando para el conjunto de datos con el que trabajamos.

Recomendadores

La implementación de estas clases se ha realizado de manera parecida a las similitudes. Primero se ha implementado una clase abstracta, a la cual se le pasan los ratings calculados anteriormente. También cuenta con un método ya implementado, el cual se encarga de recomendar ítems a usuarios. Este método crea un ranking para cada uno de los usuarios, el cual estará ordenado de mejor a peor recomendación. También tiene un método abstracto, el cual habrá que implementar en cada una de las clases de recomendadores, y que será el encargado de calcular el score final para cada ítem.

De esta manera, los recomendadores que tendremos serán los siguientes:

- *CosineRecommender*: sistema de recomendación basado en contenido el cual calcula el score mediante la ecuación 3.3.
- *UserKNNRecommender*: sistema de recomendación híbrido que calcula el score de acuerdo con la ecuación 3.4.
- *RestaurantKNNRecommender*: sistema de recomendación híbrido que calcula el score de acuerdo con la ecuación 3.5.

Destacar que los recomendadores *UserKNNRecommender* y *RestaurantKNNRecommender* tienen en cuenta en la recomendación a los k vecinos más próximos de los usuarios e ítems, respectivamente.

Evaluación

Por último, para poder hacer una evaluación de los recomendadores implementados, se ha decidido hacer otro módulo, en el cual se implementarán las métricas que evaluarán los resultados.

Las métricas elegidas, como hemos comentado en la sección 2.4, son Precisión y Recall. Precisión nos dará una estimación de las recomendaciones correctas que ha hecho el sistema, mientras que Recall nos dirá, de todos los ítems relevantes que considere el usuario, cuáles han sido bien recomendados.

En este estudio se ha decidido que, para la parte de pruebas, un usuario considera relevante un ítem cuando este ha establecido un rating superior a un umbral determinado. Por tanto, implementar estas métricas es bastante sencillo. Para Precisión bastaría con, para cada usuario, guardar el número de recomendaciones relevantes devueltas por el recomendador y dividirlo por el límite establecido para la evaluación. Esto generará un valor diferente para cada usuario. Por ello, se hace una media aritmética de todos los valores para determinar el valor medio de la métrica.

Por otra parte, para calcular Recall se divide el número de recomendaciones relevantes devueltas entre el número total de ítems relevantes para el usuario. Una vez tenemos estos cálculos para cada usuario, calculamos, al igual que en precisión, la media aritmética como valor final.

Así, a mayor media, mejor valor de la métrica.

Procesamiento final

Para poner todo en conjunto, se ha creado un programa principal, el cual será el que se ejecute tanto para recomendar como para evaluar. Este programa tendrá como parámetros de entrada los siguientes:

- $dataset$ (*obligatorio*): conjunto de datos a utilizar. Sus posibles valores son ['5k', '10k', 'complete'].
- $recommender$ (*obligatorio*): cadena que determina el recomendador a ejecutar. Los valores pueden ser uno de entre ['cb', 'ub', 'cbub', 'ib', 'cbib'].
- $topn$ (*obligatorio*): tamaño del ranking de recomendación.
- k (*obligatorio*): número de vecinos próximos a tener en cuenta para los recomendadores (exceptuando *cb*).
- $threshold$ (*opcional*): entero que determina el umbral para el que un usuario toma como relevante un restaurante. Sus valores van de 1 a 5. Por defecto es 1.
- $test\_size$ (*opcional*): determina el tamaño del conjunto de pruebas. Por defecto es 0.2.
- $test\_items$ (*opcional*): entero que se indica si queremos señalar un número concreto de usuarios a recomendar; útil para agilizar las pruebas o hacer ejecuciones más rápidas.

4

Pruebas y resultados

En este capítulo se hará una explicación de los entornos donde se han ejecutado todas las pruebas, así como un análisis de los resultados que han generado los diferentes algoritmos implementados. Aquí se comentarán también todos los problemas y cambios que nos hemos ido encontrando a medida que se han hecho las pruebas.

4.1. Entorno de pruebas

Las pruebas realizadas a lo largo de todo este trabajo final de grado se han llevado a cabo en un ordenador local, cuyas características las podemos ver en la tabla 4.1. Apuntar también que las ejecuciones de los módulos y las pruebas se hicieron en otro ordenador local previamente, pero se decidió volver a realizarlas debido a que este contaba con mejores prestaciones.

Recurso	Características
OS	macOS Monterey 12.1 (21C52)
CPU	10-core Built-In Apple M1 Pro
RAM	16 GB LPDDR5
Versión Python	Python 3.10.0

Tabla 4.1: Entorno de pruebas personal

También indicar que, para correr algunas de las pruebas en paralelo sin sobrepasar la memoria del ordenador local, con la ayuda del tutor hemos podido también ejecutar pruebas en otro entorno, el cual tiene las características indicadas en la tabla 4.2.

Puesto que estos equipos, como vemos, cuentan con características suficientes como para realizar un conjunto de pruebas real y extenso, no se pensó en otras opciones alternativas en las que se pudiesen ejecutar estas pruebas. A pesar de ello, sí que animamos al lector a tratar de buscar una solución alternativa a los equipos locales. Esto podrá ayudarle a ejecutar el conjunto completo de reseñas, sin tener que filtrar en el módulo de preprocesamiento.

Recurso	Características
OS	Ubuntu 18.04
CPU	16-core
RAM	128 GB
Versión Python	Python 3.8.5

Tabla 4.2: Entorno de pruebas tutor

4.2. Análisis de pruebas

Es en esta sección donde se explicarán al lector los resultados extraídos de las pruebas realizadas en los recomendadores, así como una explicación de por qué dan mejores resultados unos que otros. También se explicarán los datos que finalmente se han utilizado para hacer estas pruebas, pues como ya se ha comentado en capítulos anteriores, se ha tenido que reducir el conjunto final para que cupiese en memoria.

4.2.1. Explicación de los datos empleados

Todas las pruebas realizadas en este trabajo final de grado han sido ejecutadas en los entornos de las tablas 4.1 y 4.2. Así pues, esta ejecución se ve limitada por los recursos existentes, de manera que los datos con los que se puede trabajar son menos de los que realmente se podrían utilizar. En la tabla 4.3 se exponen los conjuntos de datos que se pueden generar en el preprocesamiento, siendo el conjunto 'adaptado' el utilizado en este estudio.

	#Reseñas	#Usuarios	#Restaurantes
Muestra 5k	5.000	3.752	3.395
Muestra 10k	10.000	5.946	4.875
Adaptado	294.938	10.309	6.439
Completo	5.055.992	139.916	63.944

Tabla 4.3: Desglose de los diferentes conjuntos de datos

En todos ellos vemos el número de reseñas, usuarios y restaurantes con los que cuentan. Las muestras de 5k y 10k son dos subconjuntos del conjunto adaptado, y se ha hecho uso de ellos para ir probando el desarrollo de los sistemas de recomendación más rápidamente, pues cuentan con muy pocas reseñas, lo que hace que el proceso sea más rápido. Por otra parte, el conjunto completo cuenta con todos los datos extraídos del conjunto de datos Yelp. Sin embargo, no se han podido realizar las pruebas con este conjunto debido a su tamaño, ya que los equipos donde se ejecutan se ven limitados por recursos. Así pues, se generó el conjunto de datos 'adaptado', el cual cuenta con un número suficiente de reseñas, usuarios y restaurantes como para generar recomendaciones entre ellos y poder

analizarlas.

4.2.2. Recomendaciones

En esta sección se hará un análisis de todas las pruebas realizadas con los sistemas de recomendación explicados anteriormente. Recordemos cuáles eran:

- **cb**: sistema de recomendación basado en contenido.
- **ub**: sistema de recomendación híbrido con similitud de usuario basada en ratings.
- **cbub**: sistema de recomendación híbrido con similitud de usuario basada en aspectos.
- **ib**: sistema de recomendación híbrido con similitud de ítem basada en ratings.
- **cbib**: sistema de recomendación híbrido con similitud de ítem basada en aspectos.

A su vez, recordemos también que para la evaluación de estos sistemas de recomendación se ha hecho uso de las métricas Precisión y Recall, explicadas en la sección 2.4. No tiene mucho sentido tratar de evaluar las métricas más allá de las primeras 50 recomendaciones, pues generalmente un usuario no suele ver más de ese número cuando se le recomiendan ciertos ítems. Es por ello por lo que se ha limitado la k de las métricas en ese número. Por tanto, veamos en la tabla 4.4 las métricas que se tendrán en cuenta a la hora de evaluar.

Precision@5	Precision@10	Precision@20	Precision@50
Recall@5	Recall@10	Recall@20	Recall@50

Tabla 4.4: Métricas a evaluar

Dado que se han desarrollado los sistemas de recomendación basados en vecinos próximos (salvo *cb*), se ha decidido ejecutar varias veces los sistemas de recomendación **variando este número**, hasta un máximo de 100. Se ha determinado este máximo porque con un número mayor de vecinos próximos el sistema de recomendación sobrepasaba el tiempo de ejecución de 12 horas y, debido a algunos problemas de tiempo con la fecha de entrega, no se podía dejar ejecutando mucho más.

Así, por una parte tendremos las evaluaciones del sistema de recomendación basado en contenido, el cual no depende de vecinos próximos, y por otra tendremos las ejecuciones en función de los valores de k (número de vecinos próximos).

Hagamos una estimación previa a la evaluación que nos permita entender después los resultados obtenidos. Primero, hemos de preguntarnos: **¿cuál o cuáles deberían ser los sistemas de recomendación que mejores resultados nos proporcionen?** Si nos basamos en las explicaciones del capítulo 2, a priori los sistemas de recomendación que mejores resultados deberían generar son los sistemas de recomendación basados en aspectos (éston son, *cbub* y *cbib*), pues, como previamente se ha comentado, estos sistemas combinan las ventajas de los demás tipos de sistemas, como los ba-

sados en contenido y los de filtrado colaborativo. Sin embargo, hemos de tener en cuenta que, sin una óptima configuración de los hiperparámetros que en nuestro caso calculan los pesos de los aspectos, estos sistemas pueden recomendar de manera menos óptima.

Por otra parte, los sistemas de recomendación *ub* e *ib*, que se basan únicamente en los ratings, generarán en la medida de lo posible unas recomendaciones aceptables, pues en el conjunto de datos con el que trabajamos hay suficientes interacciones usuario-ítem que pueden ayudar a proporcionar información a estos sistemas.

Por último, el sistema de recomendación basado en contenido *cb*. Este sistema de recomendación se basa únicamente en los aspectos extraídos de usuarios e ítems para recomendar, sin tener en cuenta otros parámetros. A priori, este debería ser de los que peores recomendaciones generasen.

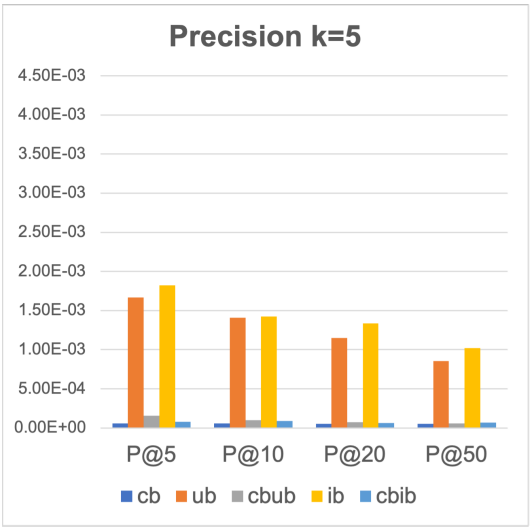
Una vez hemos hecho un pequeño estudio de cómo creemos deberían comportarse los sistemas de recomendación, veamos entonces cómo han sido los resultados empíricos de las evaluaciones, que se encuentran en la tabla 4.5.

	P@5	**P@10**	**P@20**	**P@50**	**R@5**	**R@10**	**R@20**	**R@50**
cb	5.82E-05	5.82E-05	5.33E-05	5.04E-05	2.73E-05	4.89E-05	7.33E-05	1.90E-04
$k = 5$								
ub	1.67E-03	1.41E-03	1.15E-03	8.54E-04	4.73E-04	8.40E-04	1.39E-03	2.76E-03
cbub	1.55E-04	9.70E-05	7.28E-05	5.63E-05	5.72E-05	6.56E-05	1.07E-04	2.08E-04
ib	1.82E-03	1.43E-03	1.33E-03	1.02E-03	5.88E-04	9.43E-04	1.76E-03	3.28E-03
cbib	7.76E-05	8.73E-05	6.31E-05	6.79E-05	2.63E-05	5.10E-05	8.09E-05	2.34E-04
$k = 10$								
ub	1.69E-03	1.56E-03	1.34E-03	9.41E-04	5.64E-04	1.02E-03	1.65E-03	2.87E-03
cbub	7.76E-05	8.73E-05	7.76E-05	6.01E-05	2.83E-05	5.56E-05	1.20E-04	2.39E-04
ib	1.84E-03	1.80E-03	1.47E-03	1.13E-03	5.65E-04	1.12E-03	1.88E-03	3.71E-03
cbib	7.76E-05	5.82E-05	7.28E-05	5.82E-05	2.40E-05	3.31E-05	8.64E-05	2.11E-04
$k = 50$								
ub	**2.39E-03**	1.87E-03	1.60E-03	1.25E-03	7.59E-04	1.19E-03	2.15E-03	4.04E-03
cbub	5.82E-05	5.82E-05	5.82E-05	6.21E-05	1.58E-05	2.46E-05	7.00E-05	2.11E-04
ib	2.25E-03	2.07E-03	1.68E-03	1.27E-03	7.30E-04	1.38E-03	2.27E-03	4.21E-03
cbib	1.16E-04	1.16E-04	9.70E-05	7.57E-05	2.69E-05	5.49E-05	1.22E-04	2.33E-04
$k = 100$								
ub	1.94E-03	1.96E-03	**1.77E-03**	1.27E-03	6.06E-04	1.30E-03	**2.36E-03**	4.19E-03
cbub	7.76E-05	7.76E-05	8.25E-05	6.98E-05	1.45E-05	3.10E-05	1.03E-04	2.58E-04
ib	2.37E-03	**2.24E-03**	1.76E-03	**1.29E-03**	**7.76E-04**	**1.50E-03**	2.34E-03	**4.31E-03**
cbib	5.82E-05	6.79E-05	7.28E-05	9.12E-05	1.58E-05	2.92E-05	9.81E-05	3.12E-04

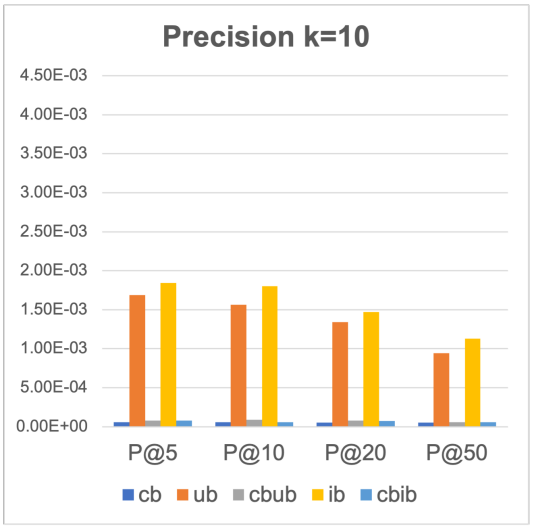
Tabla 4.5: Resultados de las pruebas

Como podemos observar en esta tabla, se han hecho varias evaluaciones de los sistemas de recomendación para diferentes valores de k (número de vecinos próximos). De esta manera, podemos entender cómo funcionan estos sistemas a medida que se va variando este parámetro.

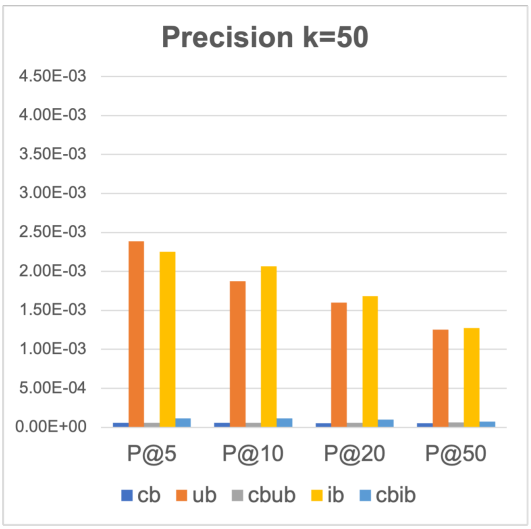
Debido a que hay muchos datos y a simple vista es difícil sacar una conclusión de qué recomendadores funcionan mejor, se ha decidido generar una serie de gráficos que nos ayudarán a entender mejor los resultados. Éstos se hallan en las figuras 4.1, 4.2, 4.3 y 4.4.



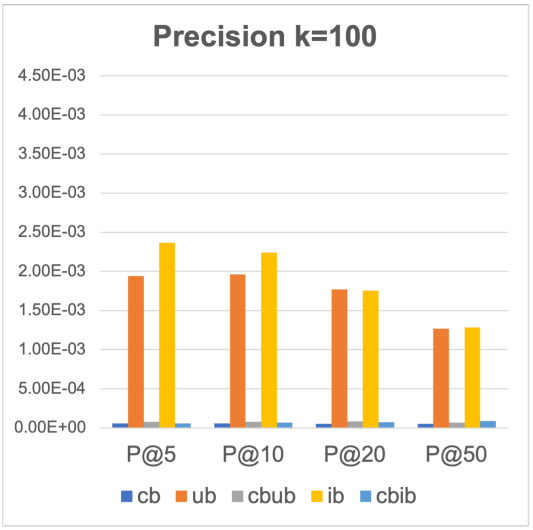
(a) Precisión con $k = 5$ vecinos



(b) Precisión con $k = 10$ vecinos



(c) Precisión con $k = 50$ vecinos



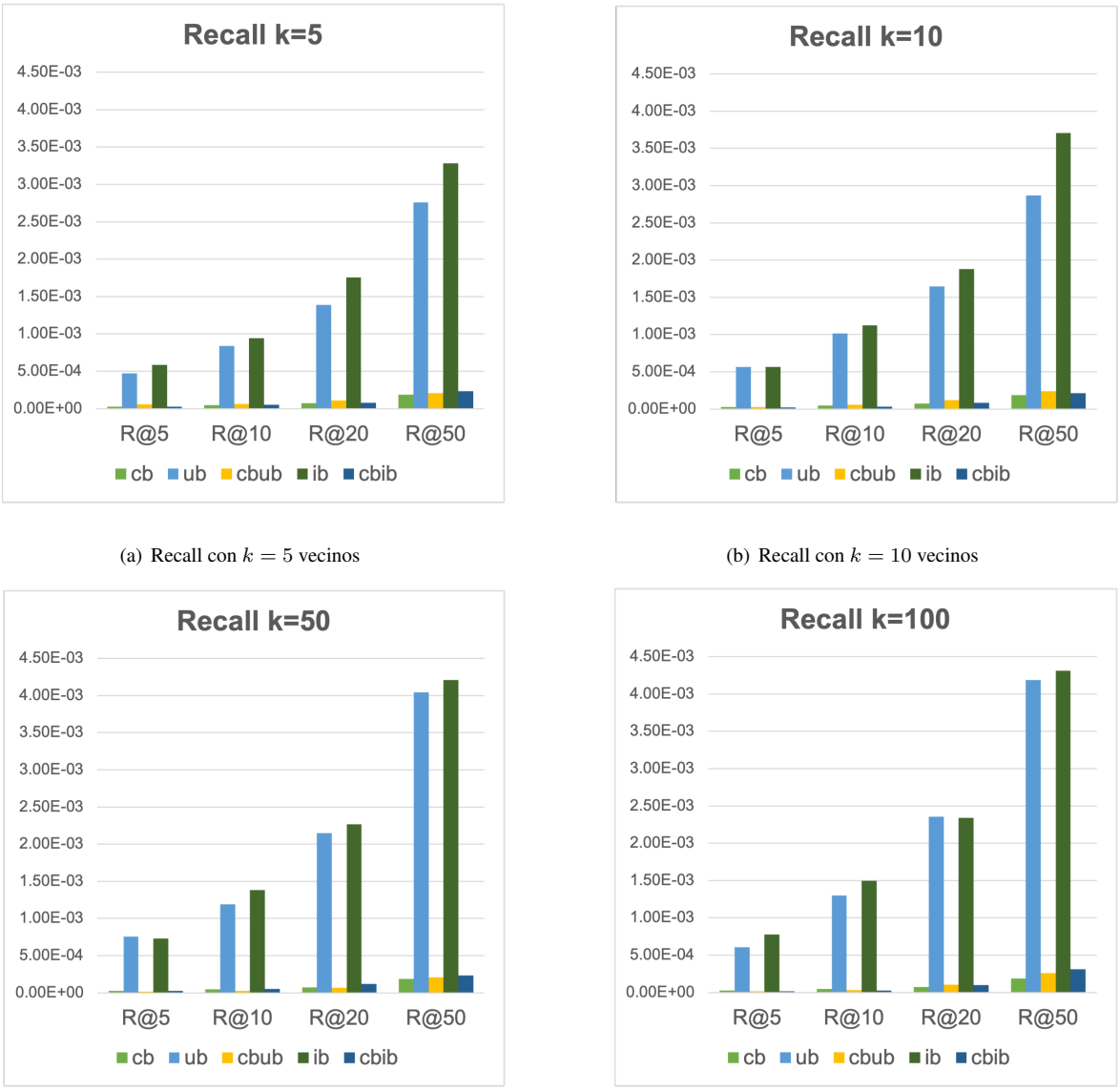
(d) Precisión con $k = 100$ vecinos

Figura 4.1: Resultados de Precisión para diferentes valores de vecinos próximos

Como podemos observar en la figura 4.1, vemos que, al contrario que habíamos pensado previamente, los sistemas de recomendación que mejor están recomendando según la métrica de Precisión son *ub* e *ib*. Además, observamos que, cuanto mayor es el *cutoff* de la métrica, sea cual sea el número de vecinos, éste va disminuyendo. Esto es debido a que cuanto más grande es el tamaño de la muestra

que se quiere evaluar, más probabilidades hay de que haya recomendaciones no relevantes para un usuario. Por ello, el valor de la métrica va cayendo cuanto más grande es el cutoff.

También se puede observar que, si bien los sistemas basados en aspectos no mejoran a *ub* e *ib*, sí que se puede observar un ligero incremento en los resultados con respecto al sistema de recomendación basado en contenido. En este caso, como bien adelantamos previamente, el sistema de recomendación basado en contenido es el que peores recomendaciones genera de acuerdo a la métrica de Precisión.



(a) Recall con $k = 5$ vecinos

(b) Recall con $k = 10$ vecinos

(c) Recall con $k = 50$ vecinos

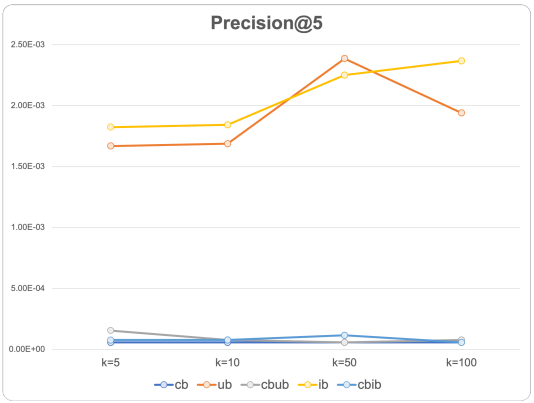
(d) Recall con $k = 100$ vecinos

Figura 4.2: Resultados de Recall para diferentes valores de vecinos próximos

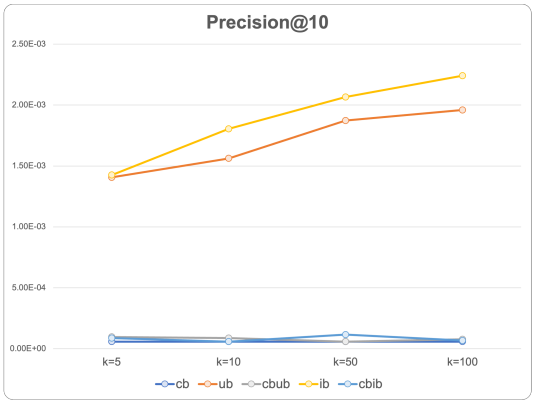
Con respecto a la figura 4.2, observamos a primera vista una clara tendencia a mejorar a medida que el límite de la métrica va aumentando. Esto es lógico, pues esta métrica tiene siempre como divisor el mismo número de recomendaciones relevantes. En este caso, a medida que aumenta el tamaño de

la evaluación, es más probable que se encuentren recomendaciones relevantes para el usuario, por lo que la métrica, en términos generales, tiende a aumentar.

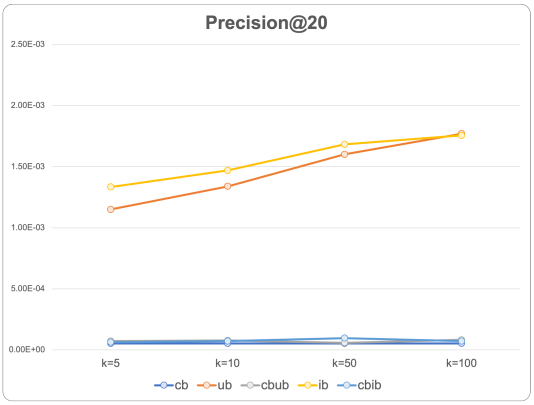
Por otro lado, tenemos la representación de los mismos datos de una forma diferente, que nos van a permitir analizar la tendencia con respecto al número de vecinos. En este caso, en las figuras 4.3 y 4.4 vemos los valores de las métricas por separado en función de esta variable. Como podemos observar en la figura 4.3, a medida que el número de vecinos va aumentando, se observa un claro incremento en el número de aciertos por parte de los algoritmos *ub* e *ib*. Por el contrario, para los demás sistemas de recomendación (excluyendo *cb* al que no le afecta) este parámetro parece ser irrelevante. Nuevamente nos encontramos con que los sistemas de recomendación *cbub* y *cbib* tienen un porcentaje ínfimo de acierto. Podríamos achacar estos resultados a lo que se comentaba anteriormente: los pesos de los hiperparámetros que se han escogido ($60\,\%$ rating - $40\,\%$ feeling) para determinar el peso final de los aspectos no parece ser el más adecuado.



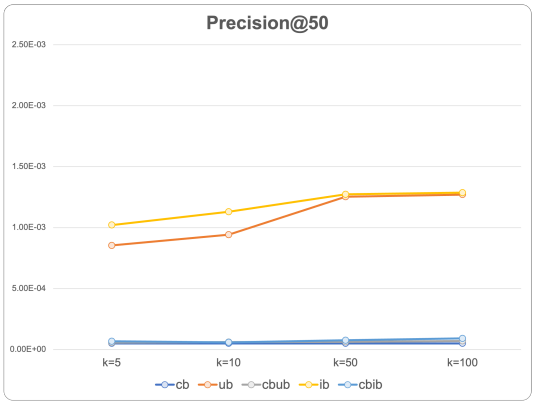
(a) Precisión con $cutoff = 5$



(b) Precisión con $cutoff = 10$



(c) Precisión con $cutoff = 20$

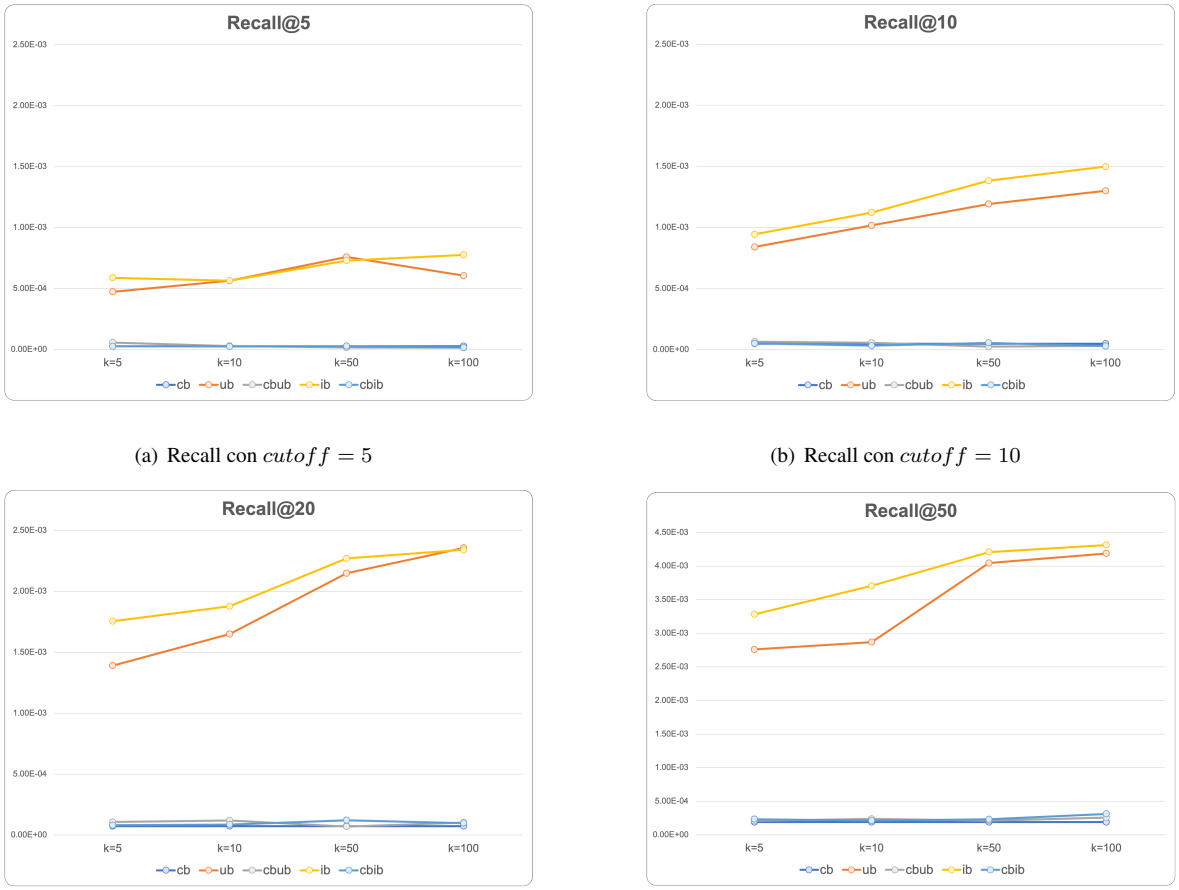


(d) Precisión con $cutoff = 50$

Figura 4.3: Resultados de Precisión para diferentes valores de cutoff

Lo mismo podemos observar en la figura 4.3. Los recomendadores *ub* e *ib* parecen ser los más acertados en la recomendación de restaurantes a los usuarios. Es por ello por lo que sería útil revisar y tratar de generar una mejor configuración de estos hiperparámetros para que los recomendadores

cbub y *cbib* puedan conseguir unos mejores resultados, pues al fin y al cabo el principal objetivo de este trabajo de fin de grado es tratar de comparar los sistemas de recomendación *tradicionales* con los sistemas de recomendación basados en aspectos. Sin embargo, se deja este análisis en profundidad sobre encontrar los parámetros óptimos para el futuro, debido a las altas restricciones de tiempo y memoria que requieren los experimentos.



(a) Recall con $cutoff = 5$

(b) Recall con $cutoff = 10$

(c) Recall con $cutoff = 20$

(d) Recall con $cutoff = 50$

Figura 4.4: Resultados de Recall para diferentes valores de cutoff

5

Conclusiones y trabajo futuro

En este capítulo se expondrán las conclusiones sacadas de este trabajo de fin de grado, así como una serie de pautas que pueden ayudar en la posible mejora y continuación del trabajo.

5.1. Conclusiones

La consecución de este trabajo de fin de grado ha pasado por diversas fases, entre las cuales podemos encontrar la motivación, el entusiasmo, la incertidumbre, la satisfacción y la desesperación, sin ir necesariamente en ese orden. Se ha pasado por algunas fases varias veces, tanto positivas como negativas, pero finalmente se ha llegado a la conclusión de que, sin este trabajo de fin de grado, hoy no me llamaría aun más la atención el mundo del tratamiento y procesamiento de datos.

Si bien ha sido un trabajo duro y arduo con el que he pasado la mayor parte del tiempo estos últimos meses, puedo sacar muchas cosas positivas del mismo. Desde saber cómo realizar un trabajo de investigación casi desde cero en cuanto a la parte técnica, hasta haber aprendido a mejorar las técnicas de coordinación y cooperación con el tutor para optimizar los tiempos de trabajo.

Con respecto a la parte experimental de este proyecto, saco varias conclusiones. La primera, y de las más importantes, es que en un trabajo donde se traten grandes cantidades de datos es imprescindible hacer un primer paso de entendimiento y preprocesamiento de los mismos. Si no se hace bien, se pueden llegar a tener resultados incorrectos o sesgados. Además, en cuanto a los resultados de los recomendadores, personalmente estoy satisfecho a medias con ellos. Si bien hemos conseguido comprender que sin una buena gestión y tratamiento de los hiperparámetros en un sistema híbrido puede hacer que los resultados sean, como hemos visto, malos, también hemos conseguido saber llegar a esa conclusión, lo cual es un aspecto positivo que podemos sacar de ello. Sin embargo, se insta al lector a seguir con el estudio de este trabajo y tratar de conseguir unos sistemas de recomendación más óptimos.

En cuanto a la parte organizativa, destacar en todo momento la ayuda del tutor y casi la inmediata disponibilidad constante durante estos meses. También en cuanto a organización, resaltar la mejoría personal en la detección de los distintos tipos de funcionalidades que puede haber dentro de un

proyecto de investigación, lo cual simplifica mucho las tareas.

5.2. Trabajo futuro

A continuación, pasaremos a enumerar algunos de los posibles cambios o mejoras que se pueden hacer sobre este trabajo, una vez ya se ha dado por terminado. Habría sido interesante poder implementar algunos de ellos en este proyecto, pero debido a diferentes complicaciones y ajustes con el tiempo no se han podido desarrollar.

Como primer punto a mejorar y que se considera muy importante, es la inclusión de otras funciones de similitud y otros sistemas de recomendación diferentes. De esa forma se podrían comparar más resultados y, por tanto, determinar más conclusiones de cuáles son mejores para este problema en concreto. Del mismo modo, como ya se ha ido comentando, también se considera importante encontrar el valor de los hiperparámetros que más se adecúe de cara a generar recomendaciones basadas en aspectos más óptimas.

Por otra parte, también sería interesante poder añadir datos de otras fuentes, ya sea mediante fuentes públicas o tratando de extraer información mediante web scraping o con el uso de APIs de terceros. De esta manera, el número de datos con el que se trabajaría sería aún más amplio, y también podría servir para tratar de reducir los posibles sesgos que tengan algunas plataformas. Sumado a esto, también resultaría útil ejecutar el conjunto de datos completo sobre un servidor dedicado, el cual no sea la consecuencia que limite la posibilidad de tener en cuenta la mayoría de registros posibles a la hora de recomendar.

Intentar guardar los datos del conjunto final en una base de datos en la nube podría ser útil para descentralizar los módulos, teniendo únicamente que preocuparse por ejecutar correctamente los programas, sin la necesidad de tener en el equipo los datos en un fichero. Así, esa base de datos sería persistente y se le podrían añadir nuevos registros en cualquier momento sin necesidad de sobrescribir ficheros, mejorando, una vez más, el proceso de descentralización.

Finalmente, una vez se tiene un sistema de recomendación correcto y bien evaluado, sería interesante también desarrollar alguna aplicación, ya sea web o SDK, que proporcione a un usuario las recomendaciones de una manera más visual. Por ejemplo, con un mapa que recomiende los restaurantes más cercanos al usuario en un momento concreto, o un listado de los restaurantes más afines a él en la ciudad donde se encuentra.

Todos estos son una serie de posibles cambios y mejoras que se podrían hacer sobre este trabajo, si bien no se cierra la puerta a cualquier otro progreso que el lector quiera realizar.

Bibliografía

[1] D. Johnson, "Natural language processing tutorial: What is nlp? examples." https://www.guru99.com/nlp-tutorial.html, 2021. Visitado última vez en Diciembre 2021.

[2] E. A. Group, "Natural language processing: ¿cómo es la técnica word embeddings?." https://blog.enzymeadvisinggroup.com/natural-language-processing, 8 2019. Visitado última vez en Diciembre 2021.

[3] S. Bird, E. Loper, and E. Klein, *Natural Language Processing with Python*. O'Reilly Media Inc., 2009.

[4] M. Mogyorosi, "Sentiment analysis: First steps with python's nltk library." https://realpython.com/python-nltk-sentiment-analysis/, 2021. Visitado última vez en Diciembre 2021.

[5] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of english: The penn treebank," *Computational Linguistics*, vol. 19, pp. 313–330, 1993.

[6] B. Rocca, "Introduction to recommender systems." https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada, 6 2019. Visitado última vez en Diciembre 2021.

[7] M. Milankovich, "The cold start problem for recommender systems." https://medium.com/yusp/the-cold-start-problem-for-recommender-systems-89a76505a7, 7 2015. Visitado última vez en Diciembre 2021.

[8] M. Hernández-Rubio, I. Cantador, and A. Bellogín, "A comparative analysis of recommender systems based on item aspect opinions extracted from user reviews," *User Modeling and User-Adapted Interaction*, vol. 29, pp. 381–441, 4 2019.

[9] A. E. B. O. del Estado and J. del Estado, *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales*. BOE, 2018.

[10] Y. Inc, "Yelp dataset." https://www.yelp.com/dataset/documentation/main, 2021. Visitado última vez en Diciembre 2021.

[11] I. R. G. . UAM, "Extracting opinions about item aspects from user reviews." http://ir.ii.uam.es/aspects/, 2017. Visitado última vez en Diciembre 2021.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID