pyRecLab: *framework* para desarrollo y enseñanza de sistemas de recomendación

pyRecLab: a framework for development and teaching of recommender systems

Gabriel Sepúlveda[1], alumno doctorado.
Denis Parra[1], profesor asistente.

[1]Departamento de Ciencia de la Computación, Escuela de Ingeniería, Pontificia Universidad Católica de Chile.
*Autor para correspondencia: dparra@ing.puc.cl.

Gabriel Sepúlveda[1], PhD student.
Denis Parra[1], assistant professor.

[1]Departament of Computer Science, School of Engineering, Pontificia Universidad Católica de Chile.
*Correspondence author: dparra@ing.puc.cl.

RESUMEN

Dentro de las tareas de recomendación, ya sea en el ámbito de predicción de ratings como en el de elaboración de rankings, existen dos componentes fundamentales que permiten llevarlas a cabo. Por un lado, se cuenta con un conjunto de datos que relacionan el nivel de afinidad expresada por ciertos usuarios hacia determinados elementos o ítems. Por otro lado, se cuenta con algoritmos capaces de convertir esos datos en información útil que permite predecir el grado de afinidad o rechazo que pueda experimentar cada usuario hacia los ítems pertenecientes al subconjunto de elementos desconocidos por ellos.

Para poder reunir estos dos componentes, existen múltiples herramientas de software que logran resolver el problema de recomendación sin tener que adentrarse en las complejidades inherentes de la programación de algoritmos, pero que sin embargo, no logran concentrar una batería completa de herramientas que permitan la realización de experimentos y análisis en un solo ambiente. Por este motivo, se ha desarrollado la biblioteca *pyRecLab* (Sepulveda, Dominguez & Parra, 2017), que intenta cubrir estas falencias proporcionando métodos de recomendación de fácil uso, y que se espera puedan ir ampliando su cobertura a través del tiempo.

Palabras clave: algoritmo de recomendación, predicción de ratings, ranking, desarrollo de software, framework python.

ABSTRACT

Within recommendation tasks, either in the field of ratings prediction or in rankings development, there are two fundamental components allowing them to be carried out. One is a set of data that relate the level of affinity expressed by certain users towards certain elements or items. The other is a group of algorithms capable of converting the data into information that predicts degrees of affinity or rejection that users may experience towards items belonging in the subset of unknown elements.

To bring these two components together, there are multiple software tools that manage to solve the recommendation problem without having to go into the intricacies inherent in programming algorithms, but do not have a complete battery of tools for conducting experiments and analyses in a single environment. The pyRecLab library was developed in order to overcome these shortcomings by providing easy-to-use recommendation methods (Sepulveda, Dominguez & Parra, 2017), which are expected to expand their coverage over time.

Key words: recommendation algorithm, ratings prediction, ranking, software development, python framework.

1. INTRODUCCIÓN

A grandes rasgos, todo problema de **SISTEMA DE RECOMENDACIÓN** tiene dos grandes aristas. La primera de ellas correspondiente a los datos, y cumple el rol de proporcionar un conjunto de valores desde los cuales se espera extraer algún tipo de información con respecto a las relaciones internas del sistema, con el fin de poder predecir relaciones futuras que aún no se han manifestado. En particular, dado un grupo de usuarios que puede consumir de un grupo ítems, se espera poder determinar la afinidad que cada sujeto pueda tener con ciertos productos no consumidos, para lograr predecir qué relaciones tienen mayor potencial de ser establecidas entre ambos conjuntos. Por otra parte, la segunda arista corresponde a las técnicas que pueden ser aplicadas sobre los datos, con el objetivo de convertirlos en información que nos lleve a aumentar nuestro conocimiento del sistema, y haga posible la tarea final, que consiste en la recomendación de ítems a usuarios. Este conjunto de técnicas se basa en la idea general de que la extracción de información útil para un usuario no solo será calculada a partir de los datos proporcionados por él, sino que también, por lo datos proporcionados por el universo

1. INTRODUCTION

Broadly speaking, any recommendation problem (**RECOMMENDER SYSTEM**) has two large aspects. The first is the data, which provides a set of values from which some type of information can be extracted, with respect to the internal relations of the system, in order to predict future, as-yet-unseen, relationships. In particular, given a group of users that can consume a group of items, it is expected to determine the affinity that each subject may have with certain unconsumed products, in order to predict the relationships with the greatest potential to be established between the two groups. The second aspect is the set of techniques that can be applied to the data, to convert them into information that improves the knowledge of the system, enabling the final task of recommending items to users. This set of techniques is based on the general idea that the extraction of useful information for a user will be calculated from the data provided by him and also by the data provided by the total user universe. This idea is commonly known as **COLLABORATIVE FILTERING** (Schafer, Frankowski, Herlocker, & Sen, 2007).

total de usuario. Esta idea es comúnmente conocida como **FILTRADO COLABORATIVO** (Schafer, Frankowski, Herlocker & Sen, 2007).

Hoy en día existe una gran variedad de técnicas que se aplican de forma satisfactoria a una gran diversidad de problemas. Pero al llegar a este punto, a menudo nos encontramos con la dificultad de no contar con herramientas de cálculo que satisfagan por completo nuestras necesidades, ya sea porque las que se encuentran disponibles de forma gratuita son limitadas, por su dificultad de uso, o bien porque son proyectos pequeños con poca difusión y proyección futura. Para avalar esta afirmación, se realizó un estudio comparativo de las herramientas de software para sistemas de recomendación que poseen mayor popularidad o que pueden ser encontradas con mayor facilidad en la web. Además, se consideró como parte de esta base, un subconjunto de las herramientas evaluadas previamente (Said & Bellogín, 2014). La lista generada quedó conformada por las siguientes aplicaciones: My Media Lite[1] (Gantner, Rendle, Freudenthaler & Schimdt, 2011), LensKit[2], LibRec[3], LightFM[4], Mrec[5], Rrecsys[6], Recommenderlab[7].

A partir del conjunto de herramientas de software encontradas, se estableció un sistema de evaluación que se enfocó en los siguientes aspectos generales: Usabilidad, Escalabilidad y Nivel de completitud en su batería de algoritmos. Para este último punto, se tomó como criterio un conjunto de algoritmos que son típicamente utilizados en los cursos introductorios a sistemas de recomendación, los cuales son: Most Popular, User Average, Item Average, User KNN, Item KNN, Slope One, Funk SVD (Koren, Bell, & Volinsky, 2009).

La **Tabla 1** resume los resultados más relevantes de nuestro análisis. Si bien en ella se aprecia que el framework My Media Lite posee todos los parámetros evaluados, a nuestro juicio presenta mayor dificultad de uso debido a que su uso está restringido a los lenguajes de programación C#, Clojure y F#.

There are currently many techniques that are successfully applied to many problems, but there often aren't calculation tools that completely satisfy our needs, either because free tools are limited, often by their difficulty-of-use or scale. To assess the functionality of these tools, we conducted a comparative study of the most popular software tools for recommendation systems that can be found most easily on the web, in addition to a subset of the tools evaluated by Said & Ballogin (2014). The generated list was made up of the following programs: My Media Lite[1] (Gantner & Rendle, 2011), LensKit[2], LibRec[3], LightFM[4], Mrec[5], Rrecsys[6], Recommenderlab[7].

We developed an evaluation system for this set of software tools that focused on the following aspects: Usability, Scalability, and Level of Completeness in its battery of algorithms. For this last point, we used a set of algorithms typically used in introductory courses to recommendation systems as a criterion, namely: Most Popular, User Average, Item Average, User KNN, Item KNN, Slope One, and Funk SVD (Koren, Bell, & Volinsky, 2009).

Table 1 summarizes the most relevant results of our analysis. Although it is appreciated that My Media Lite includes all the evaluated parameters, in our opinion it is more difficult to use, because it is restricted to the programming languages C#, Clojure and F#.

Table 1. Comparative summary of existing software tools for recommendations systems.

	My Media Lite	Lenskit	LibRec	lightfm	mrec	rrecsys	recommenderlab
Scalability	Yes	Yes	Yes	Yes	Yes	No	Yes
User Average	Yes	No	Yes	No	No	Yes	No
Item Average	Yes	No	Yes	No	No	Yes	No
Most Popular	Yes	No	Yes	No	Yes	Yes	Yes
UserKNN	Yes	Yes	Yes	No	No	No	Yes
ItemKNN	Yes	Yes	Yes	No	Yes	Yes	Yes
Slope One	Yes	Yes	Yes	No	No	No	No
Matrix Factorization	Yes	Yes	Yes	Yes	Yes	Yes	Yes

1. Available at: http://www.mymedialite.net/
2. Available at: http://lenskit.org/
3. Available at: http://www.librec.net/
4. Available at: https://github.com/lyst/lightfm
5. Available at: https://github.com/Mendeley/mrec
6. Available at: https://cran.r-project.org/web/packages/rrecsys/index.html
7. Available at: https://cran.r-project.org/web/packages/recommenderlab/index.html

En este trabajo se presenta el desarrollo de una herramienta de software para la generación de recomendaciones, el cual pretende alcanzar un alto nivel de eficiencia en cuanto a la utilización de recursos computacionales, y entregando al usuario una interfaz simple y amigable que permita la experimentación con ***DATASETS*** con información de ratings, y que provengan de diversas fuentes.

2. SOLUCIÓN

La motivación principal que nos llevó a tomar la decisión de implementar *pyRecLab*, fue la carencia de una biblioteca que reuniera tres características, que a nuestro juicio, son fundamentales en una herramienta de manejo de grandes cantidades de datos. Estas son: 1) Implementación de estructuras de datos eficientes en un lenguaje de bajo nivel, que no sobrecarguen los recursos de computación en tareas que son ajenas a los algoritmos de recomendación. 2) Interfaz amigable que acelere el proceso de desarrollo y pruebas, orientado al estudio de datos y análisis de sensibilidad de parámetros. 3) Base completa de algoritmos tradicionales, y con una arquitectura que permita aumentar la batería de métodos de forma rápida y sencilla.

Para responder a estos requerimientos, se plantea el desarrollo de un módulo basado en los siguientes puntos:

·Para lograr simplicidad de uso, se define una interfaz de usuario en lenguaje Python, ya que esto facilitará los aspectos de implementación y permitirá la realización de pruebas por línea de comando.
·Adicionalmente, se implementan los mecanismos necesarios para generar dos sabores de la biblioteca, correspondientes a las versiones 2 y 3 de Python.
·Implementación eficiente y escalable: Todo el módulo está implementado de forma íntegra en C++, lo cual otorga mayores garantías de rendimiento en comparación con otros lenguajes.
·Soporte para los algoritmos de recomendación tradicionales: se han implementado los algoritmos User KNN, Item KNN, Slope One, Item Average, Most Popular y Funk SVD, y se deja la posibilidad abierta, de seguir incorporando otros métodos.
·Facilidad para realización de análisis de sensibilidad, debido al acceso de los parámetros de cada algoritmo, a través de la modificación de los argumentos de los métodos ejecutados.
·Capacidad para el cálculo de predicciones de ratings y recomendaciones de ítems por ranking, para un usuario en particular, y para un grupo de usuarios contenidos en un set de test.
·Soporte para flexible de formatos de entrada y salida: csv, json, texto plano.

This paper presents the development of a software tool for recommendation generation, which aims to achieve a high efficiency level in terms of computational resource use and providing the user with a simple and friendly interface that allows experimentation with **DATASETS** with ratings information from different sources.

2. SOLUTION

The main motivation that led us to make the decision to implement *pyRecLab* was the lack of a library that would have three characteristics that are fundamental in a tool designed to handling large amounts of data: 1) Efficient data structure implementation at a low level language that does not overload computing resources in tasks alien to the recommendation algorithms. 2) A friendly interface that accelerates the development and testing process, oriented to the study of data and parameter sensitivity analysis. 3) A complete foundation of traditional algorithms, with an architecture that quickly and easily permits expanding the battery of methods.

To respond to these requirements, we propose a module based on the following points:

·To achieve simplicity of use, a user interface is defined in Python, thus facilitating the implementation aspects and allowing for command-line testing.
·The necessary mechanisms are implemented for generating versions of the library for Python versions 2 and 3.
·Efficient and scalable implementation: The entire module is fully implemented in C++, which provides greater performance guarantees compared to other languages.
·Support for traditional recommendation algorithms: the KNN, Item KNN, Slope One, Item Average, Most Popular, and Funk SVD algorithms; the possibility is left open to continue incorporating other methods.
·Ease to conduct sensitivity analysis, due to each algorithm's parameter access, by modifying the arguments of the executed methods.
·Ability to calculate ratings predictions and ranked item recommendations for a particular user or group of users contained in a test set.
·Flexible support for input and output formats: csv, json, plain text.

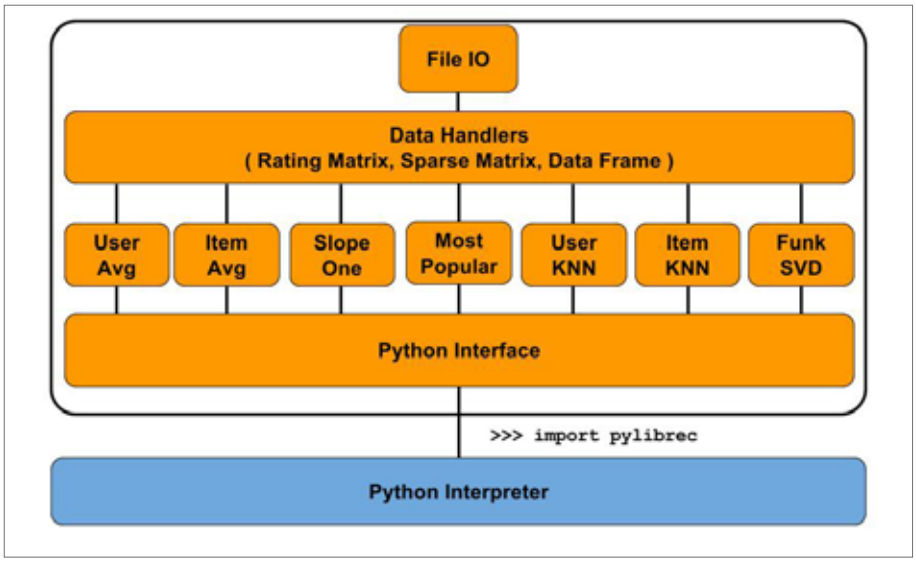

Figura 1. Arquitectura y componentes de biblioteca *pyRecLab*.

Figure 1. Architecture and components of the *pyRecLab* library.

2.1. DISEÑO GENERAL.

La **Figura 1**, muestra el diseño general de la arquitectura de la biblioteca *pyRecLab*, y los principales módulos que la componen.

Tal como puede verse de la parte inferior, el bloque azul representa el intérprete Python, el cual tendrá la misión de cargar los métodos y estructuras de datos, al importarse el módulo *pyRecLab*. De este modo, el intérprete tendrá a disponible todo el conjunto de algoritmos desarrollados para la realización de recomendaciones.

En la parte superior, en color naranja, aparecen todos los sub-módulos que componen la biblioteca, y cuyas tareas, son descritas a continuación.

2.1.1 FILE IO

Este componente representa el punto de entrada y salida de datos, ya que permite la lectura de archivos que contienen el conjunto de entrenamiento y/o prueba, además de la escritura de predicciones de ratings y/o rankings, si así se desea.

Este otorga un buen nivel de flexibilidad en cuanto a formatos de textos, debido a que sus campos pueden estar separados por cualquier tipo de carácter, el cual debe ser especificado por el usuario a través de la interfaz de alto nivel.

2.1.2 DATA HANDLERS

Este módulo contiene una serie de estructuras de datos, que permiten tener un acceso homogéneo a los valores de ratings, otorgando un mayor nivel de independencia del formato original en que fueron leídos, y con un mayor nivel de abstracción. Estas serán directamente utilizadas por los algoritmos de recomendación para el procesamiento, almacenamiento y generación de datos.

2.1. GENERAL DESIGN.

Figure 1 shows the general design of the *pyRecLab* library architecture, and the main modules composing it. The blue block represents the Python interpreter, which will load the methods and data structures when the pyRecLab module is imported. In this way, the interpreter will have the whole set of algorithms developed for making recommendations available. All the orange sub-modules that make up the library have the tasks described below.

2.1.1 FILE IO

This component represents the point of entry and exit for data, since it permits reading the files containing the training and/or test set, as well as writing ratings predictions and/or rankings, if desired. This provides a good level of text format flexibility, because the fields can be separated by any type of character, as specified by the user through the high-level interface.

2.1.2 DATA HANDLERS

This module contains a series of data structures that permit homogeneous access to the ratings values, granting a greater independence from the format in which they were originally read, and with a higher level of abstraction. These will be directly used by the recommendation algorithms for processing, storing, and generating data.

2.1.3 ALGORITMOS DE RECOMENDACIÓN

Bajo el bloque *data handlers*, se encuentra una serie de bloques contiguos que representan a cada uno de los algoritmos de recomendación implementados, y que están disponibles para la predicción de ratings y/o recomendación de ratings.

La lista de algoritmos disponibles para la predicción de ratings son: Item Average, Slope One, User KNN, Item KNN y Funk SVD. Por otra parte, para la generación de recomendaciones a través de ranking se cuenta por el momento, solo con el método Most Popular.

2.1.4 PYTHON INTERFACE

Este módulo representa la interfaz de comunicación entre los algoritmos de recomendación, y el intérprete Python. Al igual que el resto de la biblioteca, esta interfaz fue desarrollada completamente en C++, para lo cual fue necesario evaluar una serie de alternativas que permitieran realizar esta conexión entre lenguajes, pero manteniendo por sobre todo, un nivel aceptable de legibilidad del código. Por esta razón, se decidió utilizar la API Python/C, la cual viene incluida en el paquete estándar de Python. Esta permite definir estructuras de bajo nivel en lenguaje C, las que tienen un mapeo directo con los objetos manejados por el intérprete de Python. Con ella se podrán definir estructuras nativas de Python como listas, diccionarios y tuplas, así como también, clases de mayor complejidad para permitir el manejo y creación de todo tipo de datos abstractos.

Gracias a esto, se ha definido un tipo de dato por cada uno de los algoritmos de recomendación, los cuales podrán ser instanciados directamente desde el intérprete de *Python*, y permitirán controlar el comportamiento de cada algoritmo a través de los parámetros a los que se ha dado acceso.

3. EXPERIMENTACIÓN Y METODOLOGÍA

Para evaluar la implementación de *pyRecLab*, se realizan dos tipos de comparaciones con la biblioteca *LibRec* (Said & Bellogín, 2014). La primera de ellas corresponde a una evaluación de las predicciones de ratings generadas por nuestra biblioteca. Para ello, se toma una de los 5 particiones proporcionadas por el *dataset* de ***MOVIELENS***, y se ejecutan los mismos algoritmos en igualdad de condiciones, tanto de parámetros, como de datos de prueba. Luego de obtener los modelos entrenados, es posible calcular las métricas de evaluación MAE y **MRSE** sobre el conjunto de datos de prueba, para posteriormente, comparar dichos valores con los obtenidos por *LibRec*.

Adicionalmente, se realizaron pruebas comparativas en cuanto al tiempo que toman los procedimientos de entrenamiento y prueba en cada una de las herramienta, con el fin de tener una línea base que permita evaluar la calidad del desarrollo, y nos muestre si fue posible alcanzar

2.1.3 RECOMMENDATION ALGORITHMS

Under the *data handlers* block are contiguous blocks that represent each of the implemented recommendation algorithms, which are available for ratings prediction and/or ratings recommendation. The list of available algorithms ratings prediction are: Item Average, Slope One, User KNN, Item KNN and Funk SVD. For the recommendation generation through ranking, only the Most Popular method is presently available.

2.1.4 PYTHON INTERFACE

This module is the communication interface between the recommendation algorithms and the Python interpreter. This interface was also completely developed in C++, and it was necessary to evaluate a series of alternatives that would allow connection between languages, while maintaining, above all, an acceptable level of code readability. It was therefore decided to use the Python/C API, included in the standard Python package. This permits defining low-level structures in C, which directly maps with objects handled by the Python interpreter. Thus, one can define native Python structures such as lists, dictionaries, tuples, and more complex classes to permit the handling and creation of all types of abstract data. Given this, a data type has been defined for each recommendation algorithm, which can be instantiated directly from the Python interpreter, and can control each algorithm's behavior through the parameters it has been given access to.

3. EXPRIMENTATION AND METHODOLOGY

To evaluate the implementation of *pyRecLab*, two types of comparisons were made with the LibRec library (Said & Bellogín, 2014). The first was an evaluation of the ratings predictions generated by our library. For this, one of the five partitions provided by the **MOVIELENS** dataset was used, and the same algorithms were executed under equal conditions for both the parameters and the test data. After obtaining the trained models, it was possible to calculate the MAE and **MRSE** evaluation metrics on the test data set and compare these values with those from *LibRec*.

Comparative tests were also carried out on the time used by the training and testing procedures in each tool in order to have a baseline to assess the development quality and show whether it was possible to achieve one of the main objectives, namely developing an efficient library.

Table 2. Evaluation error metrics of *pyRecLab* vs *LibRec.*

	MAE		RMSE	
	pyRecLab	*LibRec*	*pyRecLab*	*LibRec*
User Avg	0.850191274	0.850191	1.062995128	1.062995
Item Avg	0.8275684033	0.827568	1.033411371	1.033411
Slope One	0.74712453	0.748299	0.9504995824	0.95246
User KNN	0.7547545252	0.755361	0.9618831072	0.966395
Item KNN	0.7512348708	0.748354	0.9557856323	0.953433
Funk SVD	0.7353138195	0.731808	0.9331742108	0.923738

uno de los objetivos principales a lograr, el cual consistía en obtener una biblioteca eficiente.

4. RESULTADOS Y DISCUSIÓN

Para comenzar, se presentan los resultados obtenidos al calcular las métricas de evaluación implementadas, sobre los datos de test. En la **Tabla 2** es posible ver que los valores obtenidos por cada biblioteca son similares para cada método, y por lo tanto, es posible validar la capacidad predictiva de la presente implementación.

A continuación se presentan los resultados obtenidos al medir los tiempos que demora cada herramienta de software en su proceso de entrenamiento y pruebas.

Tal como se observa en las **Tablas 3-5** y **Figuras 2-7**, los resultados de velocidad de cómputo obtenidos resultan ser favorables en algunos casos y desfavorables en otros. A pesar de la existencia de algunos resultados adversos, las diferencias medidas se encuentran dentro de rangos de valores razonables lo cual hace que este parámetro de rendimiento pueda ser considerado como aceptable para nuestra herramienta.

4. RESULTS AND DISCUSSION

First we present the results obtained when calculating the evaluation metrics implemented on the test data. In the **Table 2** it is possible to see that the values obtained by each library are similar for each method, and therefore, it is possible to validate the predictive capacity of the current implementation.

Next, the results obtained from when measuring the time each software tool took in its training and testing process are presented. As seen in **Tables 3-5** and **Figures 2-7**, the results of the computation speed are favorable in some cases but unfavorable in others. Despite the existence of some adverse results, the measured differences are within a reasonable ranges of values, which makes this performance parameter acceptable to our tool. However, the times obtained when executing the User Average, Item Average, and Slope One algorithms, are more often better when using LibRec, as seen in **Tables 3-5**. However, the execution times of FunkSVD in *pyRecLab* are similar LibRec in the training stage and much better than *LibRec* in the testing stage. This can be seen in **Figures 6-7**.

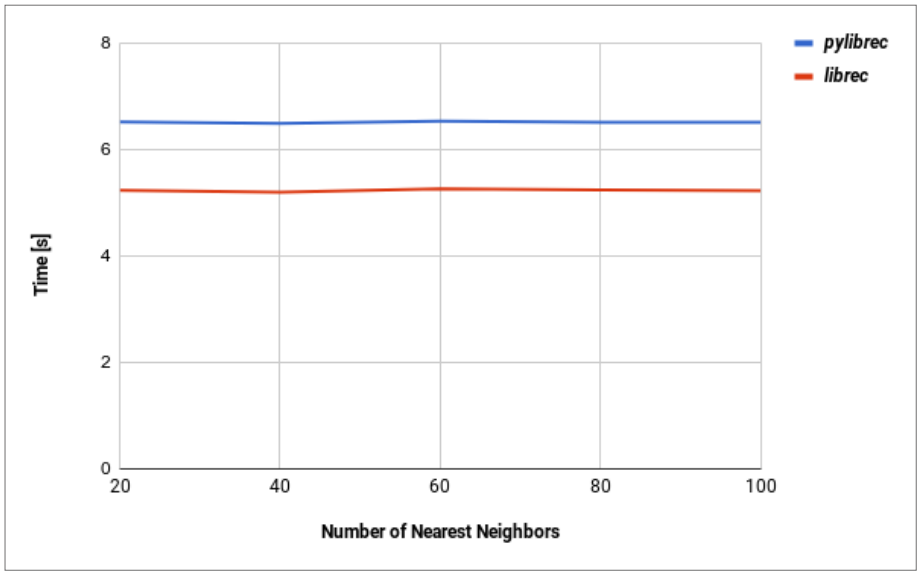

Figura 2. Tiempo de entrenamiento de pyRecLab y LibRec para algoritmo User KNN.

Figure 2. Training times in pyRecLab and LibRec for the User KNN algorithm.

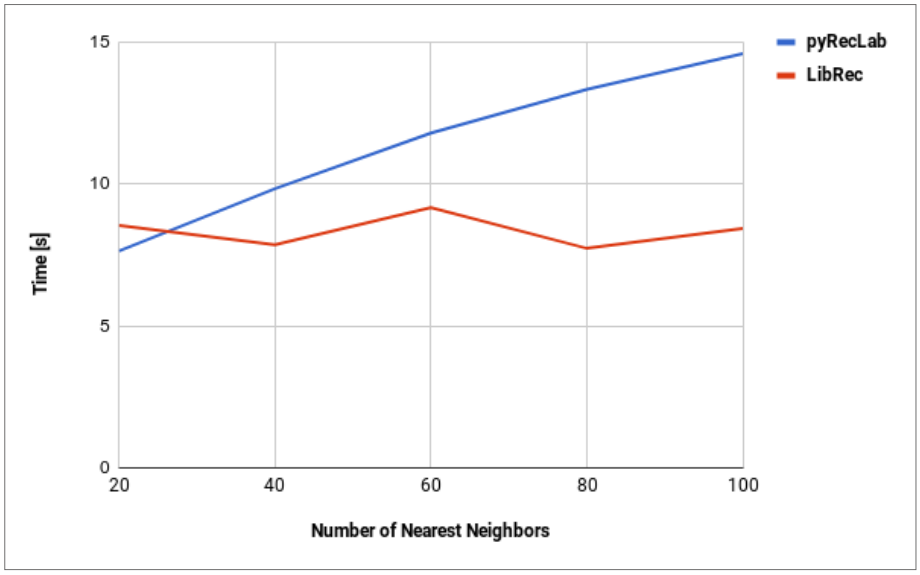

Figura 3. Tiempo de prueba de pyRecLab y LibRec para algoritmo User KNN.

Figure 3. Test times in pyRecLab and LibRec for the User KNN algorithm.

Sin embargo, hay que destacar que los tiempos obtenidos al ejecutar los algoritmos User Average, Item Average y Slope One, son en su gran mayoría mejores que los experimentados al utilizar LibRec, tal como se puede apreciar en las **Tablas 3-5**. Por otra parte, los tiempos resultantes de la ejecución del algoritmo FunkSVD en *pyRecLab* son similares a los de LibRec en etapa de entrenamiento, y mucho mejores que LibRec en etapa de prueba. Esto puede apreciarse en las **Figuras 6-7**.

A continuación, se presenta todo el detalle de los tiempos obtenidos para cada algoritmo evaluado, en donde también se realizaron variaciones de parámetros en los casos donde ellos eran de gran relevancia para el método en cuestión.

5. CONCLUSIONES

El logro de sistemas de procesamiento eficiente requiere de un fuerte análisis de cada una de las estructuras de datos

Next, all the details of the times obtained for each evaluated algorithm are presented, where parameter variations were also made in the cases where they were relevant for the method in question.

5. CONCLUSIONS

The achievement of efficient processing systems requires a strong analysis of each the data structure involved in the

Table 3. Training and test times for the *User Average* algorithm.

Training time [s]		Test time [s]	
pyRecLab	*LibRec*	*pyRecLab*	*LibRec*
0.007798	0.0125	0.125853	0.7178

Table 4. Training and test times for the *Item Average* algorithm.

Training time [s]		Test time [s]	
pyRecLab	*LibRec*	*pyRecLab*	*LibRec*
0.019149	0.0132	0.115199	0.5704

Table 5. Training and test times for the *Slope One* algorithm.

Training time [s]		Test time [s]	
pyRecLab	*LibRec*	*pyRecLab*	*LibRec*
0.562113	0.5667	0.224082	1.0392

involucradas en el proceso, debido a que tanto los tiempos de acceso como escritura de datos, comienzan a incidir de forma importante cuando su número su manejo es en grandes cantidades.

En el caso particular de los sistemas de recomendación basados en ratings, se tiene un desafío adicional que tiene directa relación con el manejo de matrices ralas. Aún cuando existen estructuras capaces de manejar de forma eficiente estos datos, y sin provocar un sobre consumo de memoria, las distintas técnicas elaboradas no son siempre las mejores en todas las situaciones. Un claro ejemplo de ello son las dos metodologías para almacenar datos de una matriz en memoria, una posicionando datos de forma contigua recorriendo las filas, y el otro recorriendo las columnas. Aquí claramente habrá que sopesar cuál de los dos métodos es más conveniente según que tipo de recorrido o acceso es más frecuente para nuestra aplicación.

process, since both access and data writing times start having an important impact when their numbers are handled in great quantities.

In the particular case of ratings based recommendation systems, there is an additional challenge directly related to the management of sparse matrices. Even when there are structures capable of handling this data efficiently, and without causing an over consumption of memory, the different developed techniques are not the best in all situations. A clear example is that of the two methodologies for storing data from a matrix in memory, one positioning data contiguously through the rows, and the other through the columns. Here, it will clearly be necessary to weigh which of the two methods is more convenient according to which type of route or access is more frequent for our application.

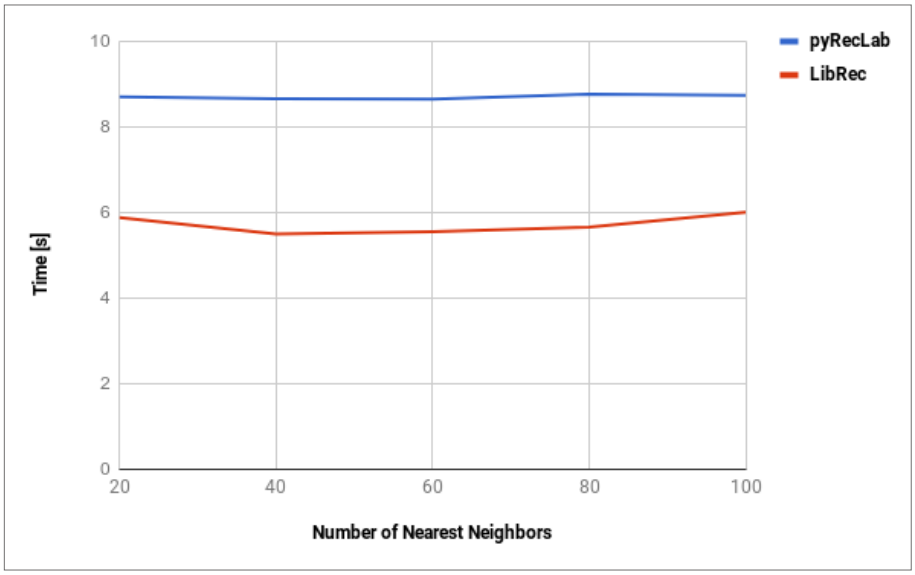

Figura 4. Tiempo de entrenamiento de pyRecLab y LibRec para algoritmo Item KNN.

Figure 4. Training times in pyRecLab and LibRec for the Item KNN algorithm.

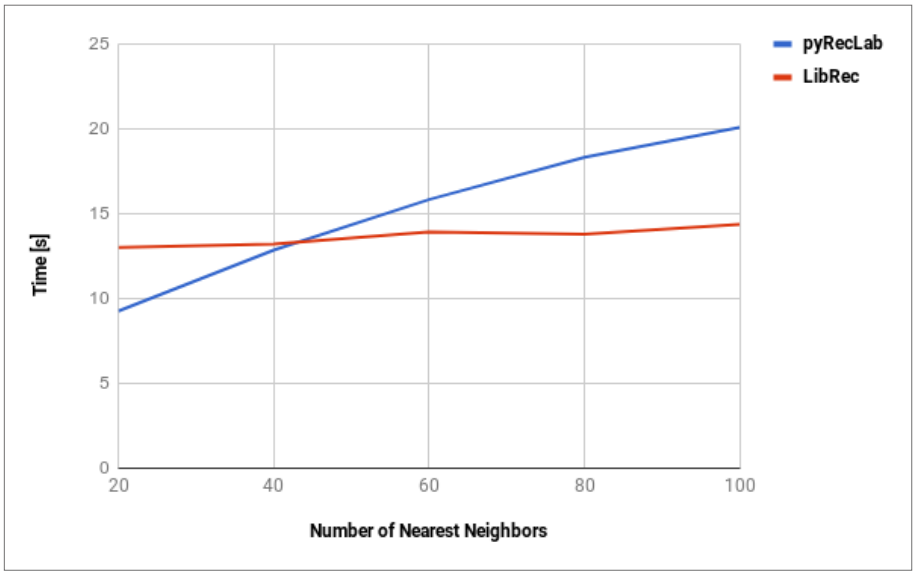

Figura 5. Tiempo de prueba de pyRecLab y LibRec para algoritmo Item KNN.

Figure 5. Test times in pyRecLab and LibRec for the Item KNN algorithm.

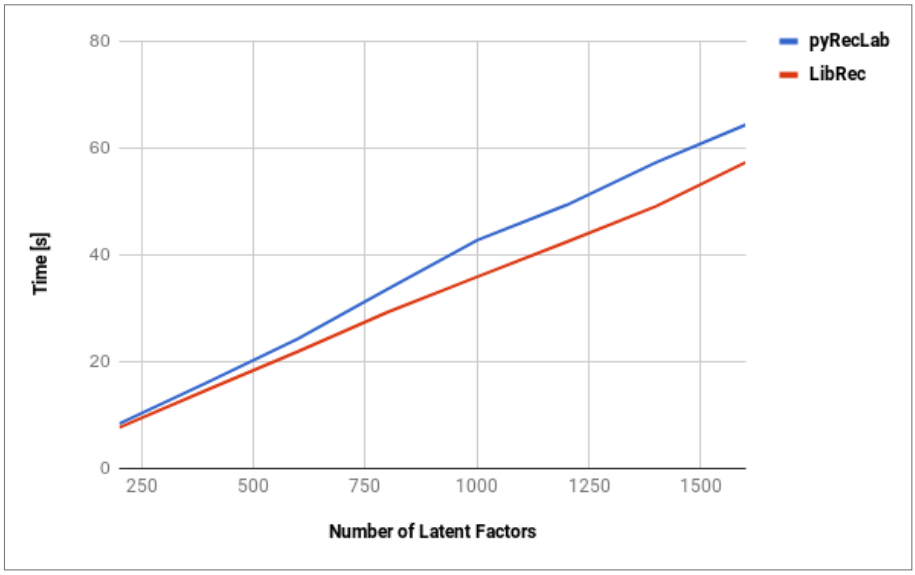

Figura 6. Tiempo de entrenamiento de pyRecLab y LibRec para algoritmo Funk SVD.

Figure 6. Training times in pyRecLab and LibRec for the Funk SVD algorithm.

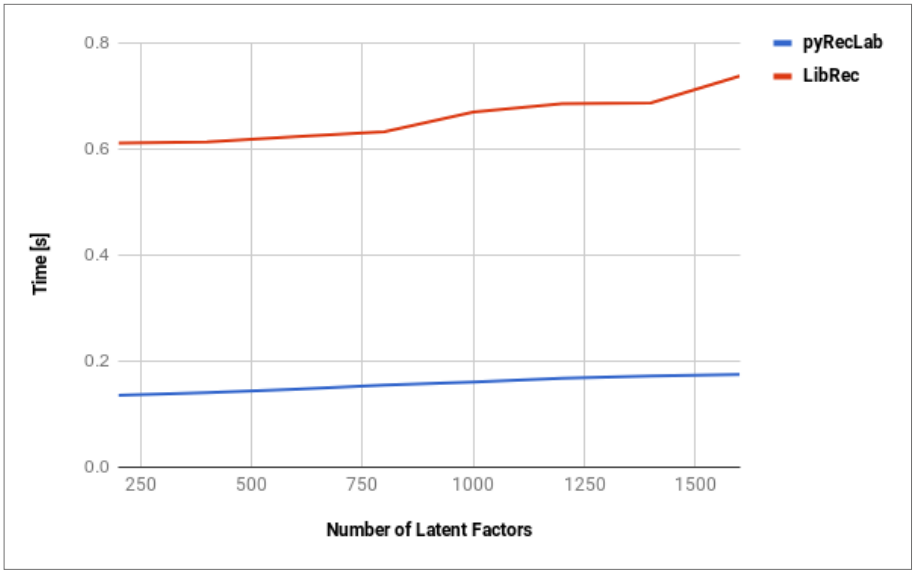

Figura 7. Tiempo de prueba de pyRecLab y LibRec para algoritmo Funk SVD.

Figure 7. Test times in pyRecLab and LibRec for the Funk SVD algorithm.

Uno de los resultados más importantes obtenidos, se presentó durante la fase iniciales del proyecto, en la cual se estudiaba el beneficio de utilizar C++ por sobre otros lenguajes. En uno de los experimentos, se programó un algoritmo para calcular una integral aproximada por rectángulos. Este fue codificado en dos versiones, una en Python puro, y la otra, en C++ pero cuya ejecución se realizó desde Python. Con esto se pudo demostrar, que para iguales operaciones de punto flotante, C++ demoraba 3 veces menos en terminar la operación en comparación con el tiempo que tomó el intérprete de Python. Esto era de esperarse al tener presentes las diferencias entre la ejecución de un código compilado, que ejecuta instrucciones directamente sobre la CPU, y entre otro cuyas operaciones pasan por otra entidad de software que debe interpretar cada instrucción.

Finalmente, durante el estudio y desarrollo quedaron varias ideas para seguir mejorando esta biblioteca, y marcar diferencias con las que comúnmente utilizadas.

One of the most important results obtained was presented during the initial project phase, in which the benefit of using C++ over other languages was studied. In one of the experiments, an algorithm was programmed to calculate an approximate integral by rectangles. This was coded in two versions, one in pure Python, and the other in C++ and executed in Python. With this, it was possible to demonstrate that for equal operations of floating point, C++ delayed three times less in completing the operation compared with the Python interpreter. This was to be expected when considering the differences between the execution of a compiled code, which executes instructions directly on the CPU, and another whose operations pass through another software entity.

Finally, several ideas to continue improving this library and mark differences with those commonly used were proposed during development. One idea is the application of approximate search techniques of nearest neighbor, based on tree structures. According to some preliminary tests, the gains in time, compared to the loss of accuracy of the neighbors

Una de ellas es la aplicación de técnicas de búsqueda aproximada de vecinos cercanos, basadas en estructuras de árboles. Según algunas pruebas preliminares, la ganancia en tiempo experimentada, en comparación con la pérdida de precisión de los vecinos encontrados, es sobresaliente. Es de extrañarse la no inclusión de dichas técnicas en las bibliotecas de recomendación analizadas.

found, is outstanding. It is surprising that these techniques are not included in the analyzed recommendation libraries.

El logro de sistemas de procesamiento eficiente requiere de un fuerte análisis de cada una de las estructuras de datos involucradas en el proceso, debido a que tanto los tiempos de acceso como escritura de datos, comienzan a incidir de forma importante cuando su número su manejo es en grandes cantidades.

GLOSARIO

SISTEMA RECOMENDADOR: es un sistema que ayuda a un usuario (o grupo de usuario) a elegir elementos relevantes desde un espacio repleto de información.
***DATASET*:** Conjunto de datos utilizados como parámetros de entrada a un sistema o algoritmo. En una primera etapa, a partir de su análisis se espera se espera la obtención de un modelo matemático que permita la predicción de sucesos futuros, no necesariamente incluidos en el *dataset* original.
***MOVIELENS*:** *dataset* de tuplas, (usuario, película, ratings) creado y compartido por el laboratorio grouplens de la Universidad de Minnesotta.
FILTRADO COLABORATIVO: grupo de técnicas que permiten la predicción de parámetros (por ejemplo, ratings) pertenecientes a una única entidad o usuario, basándose en los parámetros pertenecientes a un conjunto de otras entidades (múltiples usuarios).
RMSE: *root mean squared error* (raíz del promedio de error al cuadrado) métrica usada para medir el error en predicción de ratings, muy común en sistemas recomendadores.

GLOSSARY

RECOMMENDER SYSTEM: a system that helps a user (or user group) choose relevant elements from a space full of information.
DATASET: set of data used as input parameters to a system or algorithm. In the first stage, it is expected to obtain a mathematical model that predicts future events not necessarily included in the original dataset.
MOVIELENS: a dataset of tuples (user, movie, ratings) created and shared by the grouplens laboratory at the University of Minnesota.
COLLABORATIVE FILTERING: a group of techniques that predict parameters (for example, ratings) belonging to a single entity or user, based on the parameters belonging to a set of other entities (multiple users).
RMSE: root rean squared error; used to measure the error in rating prediction, and a very common metric in recommender systems.

PRINCIPIO CIENTÍFICO

Los sistemas de recomendación buscan resolver el problema de cómo proporcionar recomendaciones personalizadas de ciertos ítems a un grupo de usuarios pertenecientes al sistema. Estos ítems pueden ser películas, videos, imágenes, libros, artículos de compras por internet, etc., y los usuarios son caracterizados según las preferencias manifestadas dentro del sistema. Dichas preferencias pueden ser expresadas de forma explícita a través de ratings proporcionados sobre ciertos ítems, o bien de forma implícita, según el análisis de los patrones de comportamiento que el usuario tenga dentro del sistema. El fin último de los sistemas de recomendación es la entrega de información filtrada a cada uno de los usuarios para que ellos puedan aumentar su grado de satisfacción a un menor costo temporal.

SCIENTIFIC PRINCIPLE

The recommendation systems seek to solve the problem of how to provide personalized recommendations of certain items to a group of users belonging to the system. These items can be movies, videos, images, books, shopping articles online, etc., and users are characterized according to the preferences expressed within the system. These preferences can be expressed explicitly through ratings provided on certain items, or implicitly according to the analysis of the behavior patterns of the user in the system. The ultimate purpose of the Recommendation System is the delivery of filtered information to each user so they can increase their degree of satisfaction at a lower temporary cost.

Matemáticamente, el problema de recomendación puede ser formalizado como un problema de optimización con la siguiente estructura:

$$\forall c \in C, s_c' = argmax_{s \in S}\, u(c, s)$$

donde:

u: C × S → R, función de utilidad
R: conjunto recomendado de ítems
C: conjunto de usuarios
S: conjunto de ítems

Mathematically, the recommendation problem can be formalized as an optimization problem with the structure:

$$\forall c \in C, s_c' = argmax_{s \in S}\, u(c, s)$$

where:

u: C × S → R, utility function
R: set of recommended items
C: set of users
S: set of items

REFERENCES

Gantner, Z, Rendle, S., Freudenthaler, C. & Schmidt-Thieme, L. MyMediaLite: A Free Recommender System Library, *Proceedings of the 5th ACM Conference on Recommender Systems.*

Koren, Y., Bell, R., & Volinsky, C. (2009). *Matrix factorization techniques for recommender systems. Computer IEEE Magazine, 42*(8), 30-37.

Said, A., & Bellogín, A. (2014). *Comparative recommender system evaluation: benchmarking recommendation frameworks. Proceedings of the 8th ACM Conference on Recommender systems,* 129-136.

Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). *Collaborative filtering recommender systems. In The adaptive web* (pp. 291-324). Berlin, Germany: Springer.

Sepúlveda, G.; Domínguez, V. & Parra, D. (2017). *pyRecLab: A Software Library for Quick Prototyping of Recommender Systems. Proceedings of the 11th ACM Conference on Recommender systems,* http://ceur-ws.org/Vol-1905/

EQUIPO DE INVESTIGADORES / RESEARCH TEAM

Gabriel Sepúlveda

Denis Parra