

¿Cómo elijo mi proyecto de fin de semestre?

IIC3633 Sistemas Recomendadores

Denis Parra

PUC Chile 2024

Contenidos

- Etapas del proyecto
- Ejemplos de proyectos de años anteriores
- Ideas para proyectos este año
- Páginas de recursos: datasets, conferencias, etc.

Etapas del Proyecto

- **[Estudiantes]** Formar grupos y entregar propuesta (2 pags. aprox)
- [Equipo docente] Entrega feedback sobre la propuesta (Factibilidad)
- **[Estudiantes]** Envían informe de avance
- [Equipo docente] Entrega feedback sobre avance
- **[Estudiantes]** Presentación de poster (Examen)
- **[Estudiantes]** Envío de paper final y repositorio

Ejemplos de proyectos de años anteriores

- Clustering de documentos para recomendación
- Biblioteca de recomendación en Python: PyRecLab
- Recomendación en juegos MOBA
- ¿Necesitas más información? Revisa la página del curso en años anteriores

Clustering de Documentos para Recomendación

- Proyecto de Troncoso y Zorich

Hacer *clustering* de documentos permitiría:

- ▶ Agrupar los resultados de búsquedas en categorías
- ▶ Entregar diversidad en los resultados de las búsquedas
- ▶ Mejorar la presentación de la información al usuario

Clustering de documentos con *social tags*

Mauricio Troncoso, Lukas Zorich

Pontificia Universidad Católica de Chile

{mjtroncoso,lezorich}@uc.cl

27 de septiembre de 2016

El principal problema estriba en cómo detectar el **tópico** del que trata cada documento.

Usar *tags* como información adicional:

- ▶ Los *tags* entregan información semántica importante.
- ▶ Mucho contenido en internet tiene *tags*.

Documento Propuesta

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

PROPUESTA DE PROYECTO: Una comparación de diferentes técnicas de *clustering* de documentos

IIC3633 SISTEMAS RECOMENDADORES

Propuesta es corta y muy precisa, los focos principales son 1) factibilidad y 2) potencial de extensión e impacto futuro.

Índice

1. Contexto del Problema	2
2. Problema y propuesta de solución	2
3. Objetivos	2
4. Definición de experimentos	2
4.1. Conjunto de datos	2
4.2. Algoritmos a comparar	3

1. Contexto del Problema

Actualmente, los motores de búsqueda modernos están encargados de devolver los resultados más relevantes en el momento en el que un usuario hace una *query*, la que generalmente es bastante ambigua. Estos resultados más relevantes se tienen que encontrar de entre millones de documentos, lo que no es una tarea fácil.

La gran mayoría de las veces, para encontrar los resultados más relevantes el motor de búsqueda utiliza técnicas de *ranking* basadas en las conexiones entre *links*, textos de los *links*, *clicks* del usuario en los distintos datos, o buscar en el texto de la página. Sin embargo, el mayor desafío es manejar la ambigüedad que existe en la *query* del usuario. Esta *query* generalmente tiene pocas palabras y puede tener varios significados. Una manera de manejar esta ambigüedad es mediante *clustering* del contenido. Si una *query* ambigua puede tener varios significados, y asumiendo que un significado es un *cluster* de documentos de un mismo tópico, se le puede devolver al usuario documentos de varios de estos tópicos lo que aumentaría la diversidad de la respuesta. Por otro lado, si se sabe el significado de la *query* del usuario, se podrían obtener fácilmente documentos relevantes, ya que se sabría que el usuario quiere documentos de un tópico en particular. Por lo tanto, poder agrupar distintos documentos que pertenezcan a un mismo tópico es fundamental para que el usuario pueda obtener información relevante fácilmente.

2. Problema y propuesta de solución

La principal problemática que se busca atacar con este proyecto, es la de identificar tópicos de interés en documentos y poder usarlos para hacer *clusters* de documentos, lo que permitiría a los usuarios encontrar más fácilmente documentos que le interesan. El problema entonces, se representa por la pregunta: cuál es la mejor forma de hacer *clustering* de documentos?

La solución propuesta entonces, es comparar diferentes técnicas que existen hoy en día para agrupar distintos documentos.

3. Objetivos

El objetivo de este proyecto es poder comparar diferentes algoritmos de *clustering* de documentos. Además, un objetivo ideal, aunque se dejará opcional, sería el de responder la pregunta: son los tags entregados por usuarios en sitios como CiteULike una fuente significativa de información al intentar hacer un *clustering* de documentos?

4. Definición de experimentos

4.1. Conjunto de datos

Los experimentos se correrán en el *dataset* CiteEval [2]. Este *dataset* fue creado definiendo 12 áreas de interés, o categorías, dentro de la ciencia de la computación y ciencia de la información. Dentro de estas categorías se encuentran las áreas de Aprendizaje de Máquinas e IA, Bases de Datos, Arquitectura de Computadores, entre otras. Para cada una de estas categorías se definieron un subconjunto de tópicos representados como *queries*, las cuales son 99 en total. Cada una de estas 99 *queries* fueron ejecutadas en el sitio CiteULike. El resultado de cada *query* fue guardado en una colección de documentos que incluyen el id del documento, el título y el *abstract*. Para

saber la relevancia de cada documento en la *query*, se les preguntó a estudiantes de postgrado expertos en la categoría o subtópico de esa *query*. Se ocupará esta relevancia como el *ground truth* al evaluar los diferentes algoritmos.

4.2. Algoritmos a comparar

Los algoritmos que se compararán son:

- *K-means*.
- *Hierarchical clustering*. En particular, se utilizará AgglomerativeClustering.
- *Latent dirichlet allocation* (LDA).

Para el algoritmo K-means y *Hierarchical clustering*, se ocupará tfidf para representar cada palabra.

Además, dejaremos como opcional probar el algoritmo MM-LDA [1]. Para poder probar este algoritmo, será necesario extraer tags disponibles desde la página de CiteULike para los documentos que están en nuestro conjunto de datos.

Referencias

- [1] Ramage, D., Heymann, P., Manning, C. D., & Garcia-Molina, H. (2009, February). Clustering the tagged web. In Proceedings of the Second ACM International Conference on Web Search and Data Mining (pp. 54-63). ACM.
[2] Yue, Z., Harpale, A., He, D., Grady, J., Lin, Y., Walker, J., ... & Yang, Y. (2009, July). CiteEval for evaluating personalized social web search. In SIGIR 2009 Workshop on the Future of IR Evaluation (p. 23).

Paper final

Recomendación de artículos académicos utilizando etiquetas sociales

Lukas Zorich
Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile
lezorich@uc.cl

Mauricio Troncoso
Departamento de Ingeniería Matemática
Pontificia Universidad Católica de Chile
mjtroncoso@uc.cl

ABSTRACT

El almacenamiento digital de material científico ha dificultado considerablemente la tarea, de los investigadores, de encontrar trabajos pertinentes a su labor académica, luego, es pertinente ideas métodos de recomendación que se ajusten a las nuevas necesidades del mundo de la academia. Para esto se emplea un método ya utilizado para esta labor que consiste en recomendar material nuevo y antiguo en base a, respectivamente, modelación de tópicos y filtrado colaborativo. En este trabajo se extiende la labor más reciente sobre la base de datos de CiteULike, considerando la información entregada por las etiquetas que los mismos usuarios asocian a cada artículo de su interés. Se propone el modelo MM-LDA que permite agregar etiquetas de los documentos al hacer recomendaciones. De acuerdo a los experimentos, se comprueba que el uso de etiquetas permite hacer mejores recomendaciones.

1. INTRODUCCIÓN

Usualmente, los investigadores realizan las búsquedas de artículos que son de su interés siguiendo las citaciones de aquellos que ya han estudiado, esto es completamente sensato pero sus resultados están inevitablemente sesgados hacia artículos altamente citados, además, de esta forma es muy poco probable que haya movimiento entre disciplinas, esto es, que desde un trabajo, por ejemplo, de biología se lleve a otro relacionado, en el campo de la estadística.

Otra alternativa es la búsqueda a través de palabras clave, que a pesar de ser una herramienta poderosa presenta problemas en los casos en los que el investigador no sabe realmente qué está esperando encontrar, además, las búsquedas son realizadas principalmente en base a contenido y no considera que un artículo valioso es uno que también es valorado por otros usuarios. Al respecto, recientemente sitios como CiteULike¹ han permitido a los investigadores crear sus propias librerías con los artículos que son de su interés agregándoles *tags* descriptivos para ser compartidos

¹<http://www.citeulike.org>

por la comunidad. El objetivo del presente trabajo es recomendar a cada usuario un conjunto de artículos que no figuran en su librería personal y dar cuenta de qué forma las etiquetas mejoran la recomendación, ya que al ser ingresadas por los mismos usuarios representan una personalización de la evaluación hecha a cada documento.

Al momento de realizar las recomendaciones se considerarán dos importantes criterios: los usuarios quieren artículos antiguos cuando están buscando introducirse en una nueva área de estudios o cuando quieren aprender los fundamentos de ésta, al respecto, los trabajos importantes estarán en las librerías de muchos usuarios. Por otro lado, los investigadores también necesitan ser informados de las publicaciones más recientes de su disciplina que, al no tener muchas lecturas, deben ser recomendadas en base al contenido de las mismas. Así, se hará uso de dos tipos de datos: el contenido de los artículos y las librerías de otros usuarios que, inherentemente, representan una evaluación de los documentos. En ese sentido, el modelo propuesto que se explicará más adelante es un modelo híbrido, ya que ocupa filtrado colaborativo y el contenido de los documentos para hacer la recomendación.

La recomendación es hecha en base a filtrado colaborativo cuando el archivo ha sido evaluado por uno o más usuarios previamente, por otro lado, para el caso de archivos sin evaluación se utilizará modelación de tópicos (*topic modeling*) para permitir recomendaciones acordes a los intereses que ya ha mostrado el usuario.

2. ESTADO DEL ARTE

El presente trabajo aborda dos casos en los que se realizan recomendaciones, primero cuando se tienen documentos que han sido evaluados anteriormente, y segundo cuando se tienen nuevos items sin evaluación previa, este caso es conocido como el *cold start problem*.

Para la recomendación de artículos académicos con y sin evaluaciones previas, en [1] se introduce el método seguido en este trabajo, pero sin la consideración de las etiquetas que permite introducir CiteULike. En [8] agregan las etiquetas y la variable temporal para realizar recomendaciones sobre, entre otras, una base de datos de CiteULike y obtener un mejor comportamiento en la mayoría de las métricas. Por otra parte, en [7], introducen etiquetas para recomendar páginas web y proponen distintos espacios para la consideración de dichas etiquetas, en este ámbito, en [9] realizan recomendaciones personalizadas de páginas web utilizando la información de las etiquetas para crear los grupos que guían la recomendación.

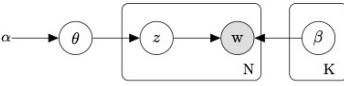


Figure 1: Modelo gráfico de LDA.

3. BACKGROUND

El método propuesto utiliza la factorización matricial usual explicada en [2] para el caso en el que el artículo al que se le desea predecir la evaluación ha sido evaluado con anterioridad, a este caso lo llamaremos, siguiendo a [1], *in-matrix prediction*. En el otro caso, en el que no hay evaluaciones previas, *out-of-matrix prediction*, la recomendación es realizada a través de un modelamiento de tópicos como se explicará en la siguiente sección.

3.1 Modelos probabilísticos de tópicos

Modelar tópicos, en este caso, tiene como objetivo dar cuenta de cómo se generan documentos, a través del descubrimiento de tópicos sobre una base de datos con muchos de ellos. El término probabilístico aparece porque cada tópico es representado como una distribución de probabilidad sobre un vocabulario que está centrada en las palabras más representativas del tópico en cuestión.

El modelo más simple de modelación probabilística de tópicos [1] es el llamado LDA por sus siglas en inglés (*Latent Dirichlet Allocation*), cuyo funcionamiento consta principalmente de dos etapas, una asignación de K tópicos para toda la base de datos, y una asignación de palabras para cada tópico, estos pasos se describen a continuación y se grafican en la Figura 1.

Para cada uno de los J documentos:

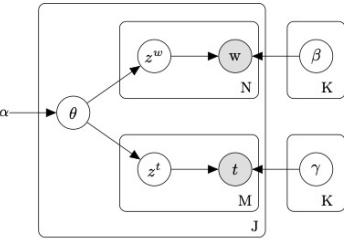


Figure 2: Modelo gráfico de MM-LDA.

- Obtener una distribución de tópicos $\theta_j \sim \text{Dir}(\alpha)$.
- Para cada tópico $k = 1, \dots, K$ obtener una distribución multinomial β_k de tamaño W . β_k es la probabilidad de observar cada palabra, dado el tópico k .
- Para cada tópico $k = 1, \dots, K$ obtener una distribución multinomial γ_k de tamaño T (cantidad total de etiquetas). γ_k es la probabilidad de observar cada etiqueta, dado el tópico k .
- Para cada palabra w_{jn} del documento j ,
 - Obtener un tópico $z_{jn}^w \sim \text{Multi}(\theta_j)$.
 - Obtener una palabra $w_{jn} \sim \beta_{z_{jn}^w}$.
- Para cada etiqueta (*tag*) t_{jm} del documento j ,
 - Obtener un tópico $z_{jm}^t \sim \text{Multi}(\theta_j)$.
 - Obtener una etiqueta $t_{jm} \sim \text{Multi}(\gamma_{z_{jm}^t})$.

3.3 Collaborative Topic Regression

El modelo *Collaborative Topic Regression* (CTR) combina filtrado colaborativo con modelamiento de tópicos. Es importante destacar que los "ítems" del filtrado colaborativo y los "documentos" de LDA se refieren a lo mismo en este trabajo, y por lo tanto se usarán ambos términos de manera intercambiable.

Al igual que en LDA, en CTR a cada documento se le asigna una proporción de tópicos θ_j , que es usado para generar las palabras. Luego, ocupa esta distribución de tópicos para generar el vector latente de los ítems v_j . El proceso generativo de CTR es el siguiente:

1. Para cada usuario i , se obtiene el vector latente del usuario $u_i \sim \mathcal{N}(0, \lambda_u^{-1} I_K)$.
2. Para cada ítem j ,
 - a) Se obtiene la distribución de tópicos $\theta_j \sim \text{Dirichlet}(\alpha)$.
 - b) Se obtiene el offset latente del ítem $e_j \sim \mathcal{N}(0, \lambda_u^{-1} I_K)$ y se asigna el vector latente del ítem el valor $v_j = e_j + \theta_j$.

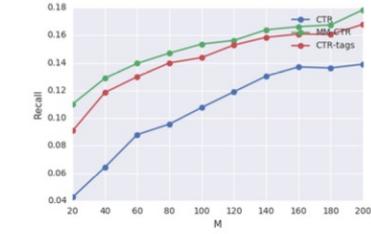


Figure 5: Comparación del recall en *out-of-matrix prediction* de los modelos CTR, MM-CTR y CTR utilizando los tags (CTR-tags) al variar el número de recomendaciones M , con $\lambda_v = 100$.

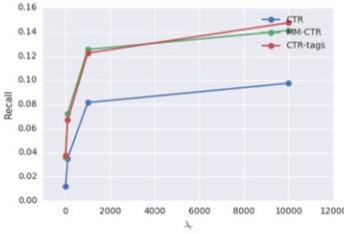


Figure 7: Comparación del recall en *out-of-matrix prediction* entre el modelo CTR, MM-CTR y CTR utilizando los tags (CTR-tags) al variar el λ_v , con $M = 50$.

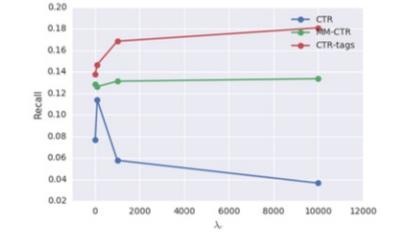


Figure 6: Comparación del recall en *in-matrix prediction* entre el modelo CTR, MM-CTR y CTR utilizando los tags (CTR-tags) al variar λ_v , con $M = 50$.

el contenido de los documentos tiene más peso.

Se corrieron experimentos con $\lambda_v = 10, 100, 1000, 10000$, con $M = 50$. Como se observa en la figura, a medida que aumenta λ_v , el modelo CTR tiene menor *performance*, lo que quiere decir que el contenido de los documentos por sí solo no basa para hacer una buena recomendación. Por otro lado, en CTR-tags y MM-CTR, el recall aumenta a medida que aumenta λ_v . Esto confirma que las etiquetas que la comunidad agrega a los documentos sí son una fuente de información importante, y pueden suprir la información de los ratings de los demás usuarios. Sin embargo, nuevamente añadir las etiquetas como parte del documento tiene mejor *performance* que el modelo propuesto.

Con la Tabla 3 y 4 se puede hacer un análisis más cualitativo de los resultados. En la Tabla 3 se muestran los tópicos más preferidos por el usuario U1, junto con las palabras y los tags que mejor describen ese tópico, y en la Tabla 4 se

muestra el resultado de una recomendación de 10 documentos para ese mismo usuario. En la tabla de los tópicos se puede observar que las palabras y los tags de ese tópico sí están relacionados. Por ejemplo, el tópico 1 tiene palabras como cerebro, señales, y actividad, y ese mismo tópico tiene etiquetas como neuroimagen (*neuroimaging* en inglés) y descanso. Esta última etiqueta puede tener que ver con la actividad del cerebro en períodos de descanso.

Con respecto a la recomendación que se le hizo al mismo usuario U1, no se tuvieron buenos resultados, ya que se recomendó solo 3 documentos que eran parte de la librería del usuario. Sin embargo, se puede ver que algunos documentos recomendados sí tenían que ver con los tópicos preferidos por el usuario de acuerdo al modelo MM-CTR. Por ejemplo, el documento 1 se refiere al aprendizaje en la sala de clases, y el tópico 2 incluye palabras como "estudiante", y "libro". Además, varios documentos recomendados están relacionados con la biología, y uno de los tópicos que el usuario prefiere tiene que ver con la actividad del cerebro, temas que están relacionados de alguna manera.

6. CONCLUSIONES

La primera conclusión del trabajo es que efectivamente las etiquetas que los usuarios agregan a los documentos son una fuente de información adicional. Al usar etiquetas se obtuvo una mejor *performance* en todos los experimentos que se corrieron.

En este trabajo se propuso un modelo que tenía la intención de ser efectivo a la hora de agregar etiquetas al modelo CTR. Sin embargo, como se vio en los resultados el modelo propuesto no fue mejor que considerar las etiquetas como parte de los documentos. En ese sentido, no se confirmaron los resultados de [7]. Es necesario hacer más experimentos y verificar que la implementación de MM-CTR fue correcta, lo que queda como trabajo a futuro. Además, es necesario ver cómo se comportan los modelos al cambiar el número de tópicos o el valor de los *priors*.

PyRecLab

- Biblioteca en Python para sistemas de recomendación de filtrado colaborativo creada por Gabriel Sepúlveda en este curso
- Presentado en la conferencia RecSys 2017 en Italia
- Escrita en C++ con un wrapper en Python
- Disponible en <https://github.com/gasevi/pyreclab>

The screenshot shows the GitHub repository page for 'pyreclab'. At the top, there are buttons for 'Unwatch' (15), 'Fork' (27), and 'Starred' (118). Below these are buttons for 'Go to file', 'Add file', and 'Code'. The 'Code' button is highlighted in green. The repository has 1 branch and 0 tags. A pull request by 'gasevi' is visible, titled 'Add ABI name for python 3.8.', with a commit hash '769f0ef' made on Sep 4, 2020, and 180 commits. The repository has 15 forks and 118 stars. The 'About' section describes pyReclab as a library for testing and prototyping traditional recommender system methods like User KNN, Item KNN, and FunkSVD. It is developed and maintained by Gabriel Sepúlveda and Vicente Domínguez. The 'algorithms' section lists various modules: algorithms, cmake_scripts, datahandlers, dataio, eval_metrics, examples, pyinterface, and pypackage. Each module has a brief description and a commit date. Below the 'About' section are several tabs: rating, svd, knn, implicit-feedback, rating-prediction, recommendation-algorithm, and alternating-least-squares.

The screenshot shows the GitHub repository page for 'pyReclab' under the 'Overview' tab. The title is 'pyReclab: Recommendation lab for Python'. It states that pyReclab is a recommendation library designed for training recommendation models with a friendly and easy-to-use interface, keeping good performance in memory and CPU usage. It is built as a Python module to provide friendly access to its algorithms and is completely developed in C++ to avoid the performance issues of interpreted languages. At this moment, the following recommendation algorithms are supported:

RecSys Algorithm	Rating Prediction	Item Recommendation	Implicit Feedback
User Average	x	x	
Item Average	x	x	
Slope One	x	x	
User Based KNN	x	x	
Item Based KNN	x	x	
Funk's SVD	x	x	
Most Popular		x	
ALS		x	x

On the right side, there are sections for 'Releases' (no releases published), 'Packages' (no packages published), 'Contributors' (4 contributors: gasevi, lalanne, ankane, juanpablo), 'Languages' (C++ 90.7%, Python 3.4%, CMake 5.9%), and 'Suggested Workflows' (based on your tech stack).

Propuesta PyRecLab

Propuesta de proyecto de Sistemas Recomendadores: Desarrollo de framework de recomendación simple, eficiente y escalable

Integrante: Gabriel Sepúlveda Villalobos.

1. Contexto del problema

Cada vez que se modela un problema de recomendación dentro de algún ámbito en particular, en el cual se definen los datos de entrada y salida, los modelos y las métricas de rendimiento a utilizar, surge inmediatamente la interrogante sobre qué herramienta de software es más apropiada para nuestras necesidades.

Para tener una referencia de las necesidades que se desea cubrir con el framework, el primer paso es conocer las funcionalidades provistas por un grupo representativo de bibliotecas en el área. Para ello, se construyen dos tablas que intentan resumir las características básicas que se plantean como deseables dentro del diseño.

Tabla 1.- Propiedades de software de bibliotecas analizadas.				
Herramienta	Plataformas	API	Línea de comando	Escalable
My Media Lite	Linux, Windows, Mac OS	C#, Clojure, F#	Sí	Sí
Lenskit	Linux, Windows, Mac OS	Java	Sí	Sí
LibRec	Linux, Windows, Mac OS	Java	Sí	Sí
Lightfm	Linux, Windows, Mac OS	Python	No	Sí
Mrec	Linux, Windows, Mac OS	Python	Sí	Sí
rrecsys	Linux, Windows, Mac OS	R	No	No
Recommenderlab	Linux, Windows, Mac OS	R	No	Sí

Como se aprecia de la tabla 1, todos las bibliotecas evaluadas tienen un alto grado de portabilidad entre sistemas operativos, sin embargo, éstas se encuentran en distintos lenguajes. Si consideramos como

Por otra parte, dado que para nuestro propósito es irrelevante el hecho de que la biblioteca cuente con una aplicación que pueda ser ejecutada directamente por consola, nuestra elección se reduce a las tres bibliotecas restantes: *Lightfm*, *Mrec* y *Recommenderlab*.

La tabla 2, resume a grandes rasgos los principales algoritmos soportados por cada una de las bibliotecas, y que son de nuestro interés como capacidades básicas deseables.

Tabla 2.- Resumen de algoritmos implementados por cada biblioteca.							
	MyMediaLite	Lenskit	LibRec	Lightfm	Mrec	Rrecsys	Recommenderlab
Ratings	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Implicit feedback	Sí	No	Sí	Sí	Sí	Sí	No
Most Popular	Sí	No	Sí	No	Sí	Sí	Sí
User KNN	Sí	Sí	Sí	No	No	No	Sí
Item KNN	Sí	Sí	Sí	No	Sí	Sí	Sí
Slope One	Sí	Sí	Sí	No	No	No	No
M.F.	Sí	Sí	Sí	Sí	Sí	Sí	Sí
MAE	Sí	Sí	Sí	No	No	Sí	Sí
RMSE	Sí	Sí	Sí	No	No	Sí	Sí
nDCG	Sí	Sí	Sí	No	No	Sí	No
MAP	Sí	No	Sí	No	No	No	No
Pre@recall	Sí	No	Sí	Sí	Sí	Sí	Sí

De la tabla anterior, podemos ver que ninguno de los algoritmos que pasaron el primer filtro impuesto, tiene implementados todos los algoritmos requeridos. Este hecho es la base de la presente propuesta.

3. Objetivos

Para lograr el objetivo planteado, se propone la implementación de una biblioteca que cumpla con las siguientes características:

- Simplicidad de utilización y capacidad de ejecución en línea: Para esto se define *Python* como lenguaje de interfaz con el usuario.
- Implementación eficiente y escalable: Para ello se evaluará la implementación de los algoritmos base en C o C++, o al menos, utilizando alguna biblioteca de *Python* que utilice código compilado. Además se asegurará el soporte de matrices ralas.
- Soporte para los algoritmos vistos en el curso: Se implementarán los algoritmos que aparecen en la tabla 2, y al menos, utilizando el método de *Non-negative Matrix Factorization*.
- Soporte para el cálculo de métricas *RMSE*, *nDCG* y *MAP*.
- Capacidad de almacenamiento de modelos para su posterior evaluación.
- Soporte para múltiples formatos de entrada y salida: csv, json, texto plano.

4. Definición de experimentos (datos, métodos, evaluación)

Para evaluar el cumplimiento de la propuesta, se utilizarán el conjunto de datos proporcionados en la tarea 1. Con ellos se obtendrán las métricas de recomendación soportadas, y también se evaluará el rendimiento en cuanto al uso de recursos de computación: Memoria y CPU.

Luego de obtenidos estos resultados, serán comparados con los logrados por la biblioteca *librec* bajo las mismas condiciones.

De ser posible, se intentará extender las pruebas utilizando el *dataset* de *MovieLens*, para tener una medida comparativa estándar tal como se realiza en [1] y [2].

Paper final PyRecLab

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.

3.1.1 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 cuando 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.

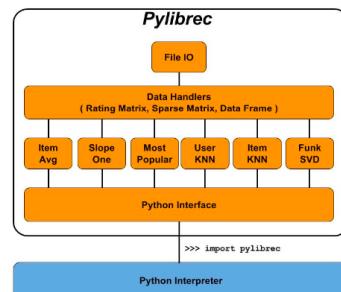


Figura 1. Arquitectura y componentes de biblioteca Pylibrec.

3.1.2 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.

3.1.3 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*.

3.1.4 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.

A continuación, se detalla la interfaz que tiene el usuario a su disposición.

3.2. Detalles de implementación.

Las clases contenidas por el módulo *pylibrec*, son listadas a continuación.

- *pylibrec.ItemAvg*
- *pylibrec.SlopeOne*
- *pylibrec.MostPopular*
- *pylibrec.UserKnn*
- *pylibrec.ItemKnn*
- *pylibrec.SVD*

Al ser instanciadas, cada una de ellas debe recibir como parámetro obligatorio, el archivo que contiene los datos de entrenamiento. Además, se pueden establecer otros parámetros que tienen relación con el carácter delimitador de campos en el archivo, existencia de cabecera informativa y posición de columnas correspondientes a usuario, ítem y rating. A continuación se presenta un ejemplo de dicha situación.

- <https://arxiv.org/abs/1706.06291>

arXiv > cs > arXiv:1706.06291

Computer Science > Software Engineering

[Submitted on 20 Jun 2017 (v1), last revised 11 Jul 2017 (this version, v2)]

pyRecLab: A Software Library for Quick Prototyping of Recommender Systems

Gabriel Sepulveda, Vicente Dominguez, Denis Parra

This paper introduces pyRecLab, a software library written in C++ with Python bindings which allows to quickly train, test and develop recommender systems. Although there are several software libraries for this purpose, only a few let developers to get quickly started with the most traditional methods, permitting them to try different parameters and approach several tasks without a significant loss of performance. Among the few libraries that have all these features, they are available in languages such as Java, Scala or C#, what is a disadvantage for less experienced programmers more used to the popular Python programming language. In this article we introduce details of pyRecLab, showing as well performance analysis in terms of error metrics (MAE and RMSE) and train/test time. We benchmark it against the popular Java-based library LibRec, showing similar results. We expect programmers with little experience and people interested in quickly prototyping recommender systems to be benefited from pyRecLab.

Recomendación en Juegos en línea

- Cuatro papers sobre recomendación de juegos han nacido a partir de proyectos finales de este curso:
 - Araujo, V., Rios, F. & Parra, D. (2019). Data mining for item recommendation in MOBA games. In *Proceedings of the 13th ACM Conference on Recommender Systems* (pp. 393-397).
 - Cheuque, G., Guzmán, J. & Parra, D. (2019). Recommender systems for Online video game platforms: The case of STEAM. In *Companion Proceedings of The 2019 World Wide Web Conference* (pp. 763-771).
 - Villa, A., Araujo, V., Cattan, F. & Parra, D. (2020). Interpretable contextual team-aware item recommendation: Application in multiplayer online battle arena games. In *Proceedings of the 14th ACM Conference on Recommender Systems* (pp. 503-508).
 - Araujo, V., Salinas, H., Labarca, A., Villa, A. & Parra, D. (2022). Hierarchical Transformers for Group-Aware Sequential Recommendation: Application in MOBA Games. In *Adjunct Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization* (pp. 293-301).

Araujo y Ríos 2018 : Recomendar ítems en LoL

Recomendador de items para League of Legends

Vladimir Araujo, Felipe Ríos

Contexto

League of Legends

El objetivo general del juego es derribar la base del equipo:

- Enfrentamiento de equipos de 5 vs 5.
- Amplia selección de personajes.
- Gran disponibilidad de ítems que generan utilidad o mejoras en dicho personaje en variadas formas.

Solución propuesta



Dado un campeón en un equipo conocido y enemigos conocidos

Encontrar relaciones entre ellos para recomendar ítems

Metodología

Análisis Dataset

- Revisión de la data
- Selección de características.
- Adecuación de datos.

Recomendador

- Recomendador basado en contenido.

Alternativas:

- Basado en contexto
- Machine learning

Evaluación

- Evaluación por relevancia de ítems recomendados.

Alternativas:

- Evaluación por Win/Lose

Paper final

Data Mining for Item Recommendation in MOBA Games

RecSys '19, September 16–20, 2019, Copenhagen, Denmark

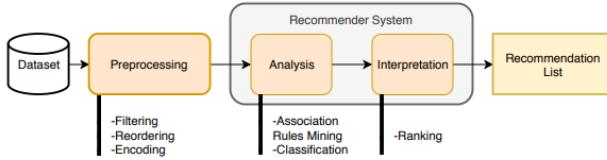


Figure 2: Recommendation framework based on data mining process.

On the one hand, all new *champions* released in the game after season 7 were removed, leaving 136. On the other hand, only finished items were used, which means that basic, advanced, and consumable items were eliminated, resulting in 93 items. This final file was divided into 631, 590 instances for training and 70, 405 instances for testing.

4.2 Recommender System Based on Association Rules Mining

The association rules have already been used for the recommendation of items for DotA 2 [15], but our approach is slightly different. We filtered the dataset for each champion of the game for rules mining (including both the enemies confronted and the items used as a transaction). We did this because item purchasing strategies tend to be champion-specific [5] and also so that the item recommendation will be context-aware.

This system consists of an association rules algorithm module that extracts the frequent itemset of each champion. We used the Apriori and Eclat algorithms for this task. Also, a search engine module is responsible for generating a recommendation through the search in the files stored by the previous module.

Through $X, Y = \phi_{AR}(D | supp_{min}, conf_{min})$, the algorithm generates rules with the form $X \rightarrow Y$, where Y is an enemy *champion* and X is a subset of items. D is the transactional data of a picked champion composed of the enemy team and the items used.

On the one hand, *support* is defined as the perceptual frequency of observation of the same item in all the transactions. On the other hand, *confidence* is defined as the probability $P(Y | X)$. For both it is important to set a minimum value $supp_{min}$ and $conf_{min}$. Both Apriori and Eclat were set up with minimum support of 4 and minimum confidence of 80%.

In this way, it is possible to find a frequent itemset $I_f = X$. However, because it is quite likely that I_f could contain a subset of frequent itemset J_f , we use the maximal itemset that is defined as $I_m = I_f \mid I_f \wedge \nexists J_f \supset I_f$. The itemset I_m retrieved with the highest confidence were used to generate the recommendation.

4.3 Recommender System Based on Classifiers

Classifiers have been used frequently for mining tasks [16]. For this work, we applied a supervised approach for multi-label classification to obtain a mapping between features space (*champions*) and label space (items). On the one hand, the input, which represents a

match played, was coded with one-hot, where the selected *champion* was represented with 1, the enemies with -1, and the rest with 0. On the other hand, the output presents the items used with 1 and 0 for the others. Though we evaluated three types of classifiers, the task can be carried out with other models.

4.3.1 Decision Trees. This technique generates a classifier in the form of a tree structure based on the features of the inputs. In this work, each decision node evaluates the *champion* selected, and the enemies presented, while leaf nodes indicate the item used in that context. We have implemented a decision tree with the Gini impurity function to measure the quality of a split because it is good at dealing with classification errors and is not computationally exhausting. Also, we do not use any maximum depth value for the tree to capture special cases of the context.

4.3.2 Logistic Regression. Logistic regression is a linear model used to predict the probability that an input belongs to one class or another. Typically, this model is trained with gradient descent, but due to the amount of this dataset, we decided to use stochastic gradient [22]. Besides, to use this model as a multi-label classifier, we adopted the one-vs-all strategy, which consists of fitting one classifier per class. The class is then fitted against all the other classes for each classifier.

4.3.3 Artificial Neural Network. Multi-Layered Perceptron (MLP) is frequently used to approximate nonlinear relationships existing between an input and the corresponding output. We used a fully connected architecture with 2 hidden layers of 150 neurons each, the input layer corresponds to the 136 *champions* and the output layer corresponding to the 93 items. Because this is a multi-label classification, we used binary cross-entropy loss function and sigmoid activation function to evaluate each class independently.

4.4 Interpretation by ranking

In a general view, in order to generate the itemset recommendation for the user, we use a sort function over the output of the recommender systems based on its probabilities $RecList = sort(OutputRecSys, P)$.

On the one hand, association rules methods generate several frequent itemsets with a confidence value associated. We organized these itemsets from highest to lowest, and then we take the first one. If the highest confidence itemset does not have a high enough amount for generating the recommendation required, the next itemset is merged.

RecSys '19, September 16–20, 2019, Copenhagen, Denmark

Vladimir Araujo, Felipe Rios, and Denis Parra

Table 1: Results for @N-itemset

Method	Precision	Recall	F1-Score	MAP	MRR
Apriori@1	0.23	0.11	0.18	0.43	0.43
Eclat@1	0.44	0.12	0.18	0.44	0.44
D Tree@1	0.65	0.17	0.27	0.64	0.64
Logit@1	0.67	0.18	0.28	0.67	0.67
ANN@1	0.71	0.19	0.30	0.71	0.71
Apriori@3	0.42	0.33	0.36	0.59	0.63
Eclat@3	0.42	0.33	0.36	0.60	0.61
D Tree@3	0.47	0.37	0.41	0.71	0.73
Logit@3	0.53	0.42	0.46	0.74	0.76
ANN@3	0.60	0.48	0.52	0.78	0.79
Apriori@6	0.40	0.62	0.48	0.58	0.63
Eclat@6	0.41	0.62	0.48	0.59	0.64
D Tree@6	0.32	0.50	0.38	0.69	0.75
Logit@6	0.37	0.59	0.43	0.71	0.78
ANN@6	0.44	0.69	0.53	0.74	0.81

On the other hand, because of the implemented classifiers have a multi-label output, we apply sort function over the list of probabilities. This function returns the most probable item for the context at the beginning, and so on.

5 RESULTS AND DISCUSSION

In order to measure the quality of our systems, we adopt an evaluation per itemset size using metrics like precision, recall, and F1-score.

It must be taken into account that it is possible that a *champion* has not completed its items-build during a match. Because of that, the first evaluation was done by generating a fixed itemset size recommendation for each instance in the test set.

Table 1 shows the results, and we also have included the mean average precision (MAP), which evaluates relevant items from 1 to k , and the mean reciprocal rank (MRR), which evaluates the rank of the first correct item. Typical behavior occurs in all models: precision decays and recall rises while the list of recommendations increases. Therefore, F1 grows, transmitting the balance between precision and recall.

As we expected ANN model outperforms the other approaches [1], achieving F1 of 53%, and MAP of 74%, while the worst was Apriori approach reaching F1 of 48%, and MAP of 58%, showing a difference of ~15% between them. Also, ANN obtained 71%, 79%, and 81% of MRR, which might indicate that this model recommends the first best situational-item.

For the second evaluation, we take into account the dimension of the label vector. We divide the dataset of testing into six groups, according to the size of the label vector, and each group is evaluated with itemset recommendations of the corresponding size. Figure 3 shows the results. In this evaluation, we also add Random and *champion-based* Most Popular recommendation because there is not a previous baseline. It is possible to see that the ANN recommender has the highest performance once again, reaching 80% of MAP@6.

In contrast, as we expected, the Random recommendation achieved lower performance with almost 15% of MAP@6. Remarkably, Most

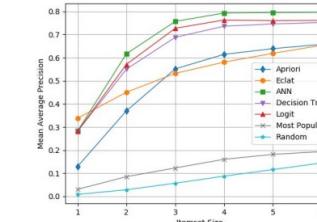


Figure 3: Evaluation with MAP by itemset size for the second evaluation.

Popular recommendation achieved only 20% in spite of being a *champion-based* version. This result may suggest that items used during a match are dependent on the current situation.

As seen in the previous evaluation, ANN, Logit, and Decision trees outperform Apriori and Eclat algorithms in almost all itemset sizes. There was an unexpected case with the Eclat algorithm, which could predict the first best situational-item (best performance of MAP@1). However, we assume that it was a coincidence because it then shows the lowest performance than the rest.

The results are promising. However, we only use the finished items. The inclusion of the remaining items could reduce the performance of our models, so more sophisticated systems that take into account more information from the dataset may be necessary.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we presented a simple framework for the recommendation of items for LoL. It is based on a data mining methodology using five different methods. The evaluation showed promising results with the best model based on ANN, which seems to be able to make recommendations according to the context (enemy team).

This work shows that the recommendation systems in the MOBA games have relevant challenges to face, encouraging to explore this field to improve these types of systems.

Qualitative evaluation by an expert would be desirable to analyze the list of recommendations in an experimental setting. It might ensure that the suggested items are reliable and that they include context information. We will also continue with the context approach, but use additional information such as the allied team, characteristics of the items, and the role of the characters.

It would be desirable to have a system that can run during a match. Future work will focus on assert which technique could run in real-time without losing accuracy.

ACKNOWLEDGMENTS

This work has been partially funded by Millennium Institute for Foundational Research on Data (IMFD).

<https://dl.acm.org/doi/abs/10.1145/3298689.3346986>

<https://dparra.sitios.ing.uc.cl/pdfs/MOBArecsys2019-preprint.pdf>

https://www.youtube.com/watch?v=HI9ftgGUlmE&ab_channel=YonseiEsportsLab

Propuesta: Recomendación adversaria

- Proyecto de curso del año 2021
- Estudia el uso de modelos adversariales (GANs) para recomendación y propone una mejora para cold-start y diversidad en recomendación
- Reporte final: <https://github.com/PUC-RecSys-Class/proyecto-juan-banach/blob/main/paper.pdf>
- Poster <https://github.com/PUC-RecSys-Class/proyecto-juan-banach/blob/main/Poster/Poster.pdf>
- Presentado este año en el workshop FashionXRecSys 2023 de la conferencia RecSys

Program (Singapore time)

Fashion x RecSys 2023 - Schedule

Time ¹	Talk	Speaker	Title
14:00-14:10	Intro	Organizers	Opening Remarks
14:10-14:45	Keynote	Prof Calvin Wong	Acceleration with AI: From Fashion Design to Quality Control
14:45-15:00	Paper (pdf)	Himanshu Arora	[Desaki et al.] Automated Material Properties Extraction For Enhanced Beauty Product Discovery and Makeup Virtual Try-on
15:00-15:15	Paper (pdf)	Mario Mallea	[Mallea et al.] Overcoming popularity bias in adversarial pairwise learning for fashion recommendations
15:15-15:30	Paper (pdf)	Alexandre Candeias	[Candeias et al.] Tailor: Size Recommendations for High-End Fashion Marketplaces
15:30-15:40	Early Results (pdf)	Sahar Mbarek	[Mbarek et al.] Exploring Fit and Shape for Predicting Garment Size Issues in Fashion with Multi-Task Learning
15:40-15:55	Break (aligns with RecSys coffee break)		
15:55-16:30	Keynote	Tian Su	Fashion, Emotion and Innovation
16:30-16:45	Paper (pdf)	Nour Karessli, Vladimir Vlasov	[Dibak et al.] UNICON: A unified framework for behavior-based consumer segmentation in e-commerce
16:45-17:00	Paper (pdf)	Veronika Shilova	[Shilova et al.] AdBooster: Personalized Ad Creative Generation using Stable Diffusion Outpainting
17:00-17:15	Paper (pdf)	Eden Dolev	[Dolev et al.] Efficient Large-Scale Visual Representation Learning and Evaluation

<https://fashionxrecsys.github.io/fashionxrecsys-2023/>

Proyectos de años anteriores del curso

- Revisa la sección “Proyectos finales” de años anteriores
- <https://github.com/PUC-RecSys-Class/RecSysPUC-2020>
- <https://github.com/PUC-RecSys-Class/RecSysPUC-2021>

Grupo	Proyecto	Poster	Paper
1	Recomendación a conjuntos de usuarios en grupos heterogeneos Cartagena, Huerfano, Toscano	poster	paper
2	Personality bias in music recommendation: Beyond accuracy Objectives Valencia, Gonzalez	poster	paper
3	Attack learning: a method using GANs Castro, Casassus	poster	paper
4	Metrica Beyond Accuracy: Personal Labarca, Fuentes	poster	paper
5	Sequential Recommenders for MeLiDataChallenge 2020 Aguilera, Everke	poster	paper
6	Exploración de recomendadores híbridos para música Suarez, Carreño, Alipanah	poster	paper

Proyectos finales

L@s estudiantes trabajan en grupo sobre proyectos finales de curso, produciendo un poster, paper y repositorio con código para cada uno:

Grupo	Proyecto	Poster	Paper
1	VAE based model for single-target cross-domain recommendation De Diego, Hernández, Schuit	poster	paper
2	Spotify Playlist Continuation Carstens, López, Mendoza	poster	paper
3	Hybrid recommender of articles based on Topic Modeling and Collaborative Filtering Flores	poster	paper
4	Efficiency of Shilling Attacks in Modern Recommenders Brancoli, Gazali, Murtagh	poster	paper
5	Métricas de Sesgo de Posición para Recomendaciones de Artículos Científicos Brancoli, Gazali, Murtagh	poster	paper

Proyectos de años anteriores del curso II

- Revisa la sección “Proyectos finales” de años anteriores
- <https://github.com/PUC-RecSys-Class/RecSysPUC-2023/tree/master>

Data augmentation for recommender systems using LLMs

Rodrigo Pozo - Oscar Loch - Luis Miranda - Cristóbal Vásquez
Grupo 11

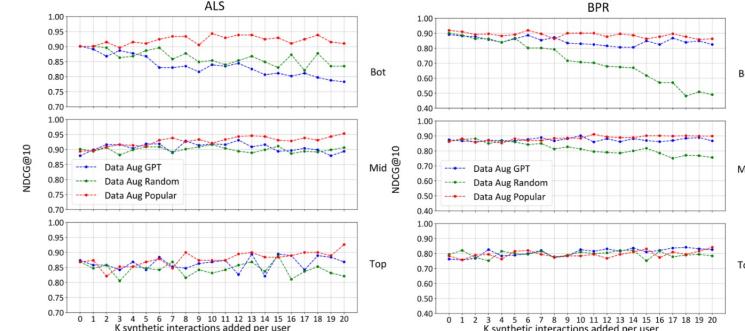
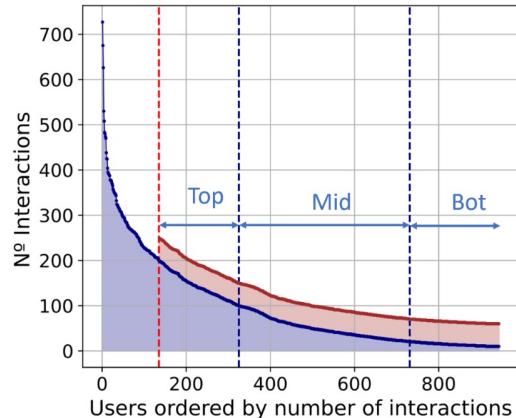


Figura 5: Resultados para NDCG@10 usando ALS y BPR

Recomendación Multi Modal para Juegos de Mesa

Jueves 14 de Diciembre
Sara Godoy
Álvaro Postigo

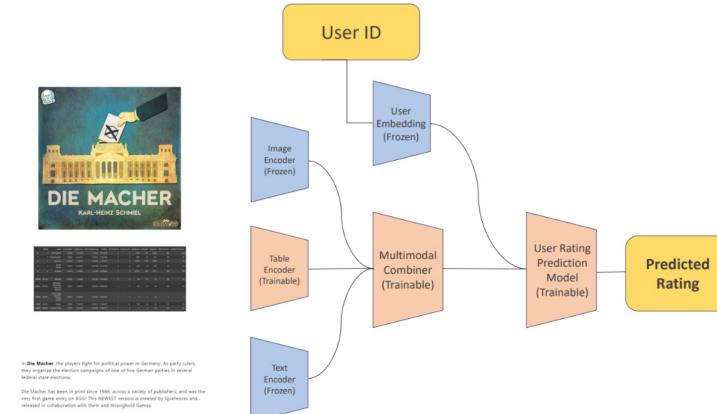


Figura 1: Diagrama Modelo

Ideas para proyectos este año

1. Modelos de Lenguaje (ChatGPT, Llama)
2. Modelos multimodales: CLIP, CLIP Lite, Pixel
3. Self-supervised learning para RecSys: Masked Autoencoders
4. Reproducción y Reproducibilidad
5. PyRecLab v2: agregar algoritmos
6. Estudiar simuladores de RL para recomendación

Modelos de Lenguaje (ChatGPT, Llama)

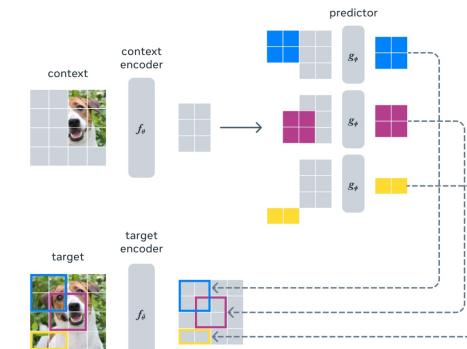
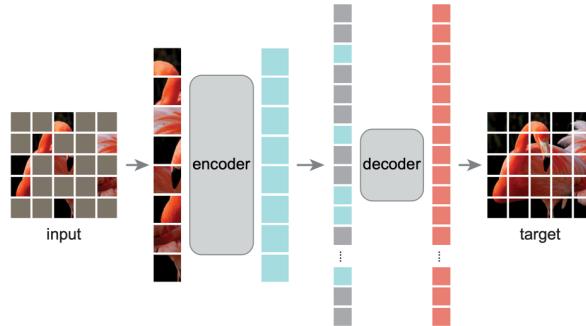
- Se podrían usar modelos de lenguaje (LMs) para hacer zero-shot recommendations, es decir, dar un prompt ejm. “al usuario X le gusta A y B, que le recomendarías”:
 - Se podrían comparar modelos en nube chatGPT versus locales Llamav2
 - Se podría estudiar diferentes formas de finetunear y el efecto en el recomendador sobre varias dimensiones: diversidad, novedad, ranking
 - Se podría usar el LM para obtener etiquetas y disminuir el cold-start
 - Se podría estudiar el efecto de bias (sesgo) que podría tener la estrategia anterior
 - Etc.

Modelos multimodales: CLIP, CLIP Lite, BLIP2, Pixel

- Usando un dataset que tenga al menos dos modalidades (texto e imágenes o audio e imágenes) además de ratings/likes, testear el efecto de diversos modelos multimodales para la recomendación personalizada
- Estudiar efecto de finetuning versus zero-shot
- Estudiar diferentes formas de combinar las modalidades para hacer recomendación
- <https://github.com/ialab-puc/VisualRecSys-Tutorial-IUI2021>
- Dataset: <https://github.com/westlake-repl/PixelRec>

Self-supervised learning

- Adaptar Masked autoencoders / I-JEPA para la tarea de recomendación
- Sería un encoder visual distinto para luego ponerlo en el pipeline de un recomendador de imágenes
<https://github.com/facebookresearch/mae>
- <https://ai.meta.com/blog/yann-lecun-ai-model-i-jepa/>



Reproducción y Reproducibilidad

- No todos los paper publicados tienen una implementación pública
- Una muy buena idea de proyecto es implementar el modelo del paper ver si es posible replicar resultados
- Posteriormente, si la replicación fue posible, testear lo mismo pero con otros datasets para ver si el resultado es consistente (reproducible)
- Ejemplo: estudio de reproducibilidad de GRU4Rec, algoritmo para sesión-based recommendation

The screenshot shows a Cornell University logo and the text "We gratefully acknowledge s member inst". Below it is the arXiv logo with the URL "arXiv > cs > arXiv:2307.14956". The title "Computer Science > Information Retrieval" is followed by the submission date "[Submitted on 27 Jul 2023]". The main title is "The Effect of Third Party Implementations on Reproducibility" by "Balázs Hidasi, Ádám Tibor Czapp". The abstract discusses the reproducibility of recommender systems research, mentioning the examination of six third-party implementations of a popular algorithm across five datasets.

<https://arxiv.org/abs/2307.14956>

PyRecLab v2

- El proyecto PyRecLab solo tiene los siguientes algoritmos implementados
- Podrías agregar nuevos métodos y actualizar los que ya están implementados (requiere conocimiento C++ y Python libraries)

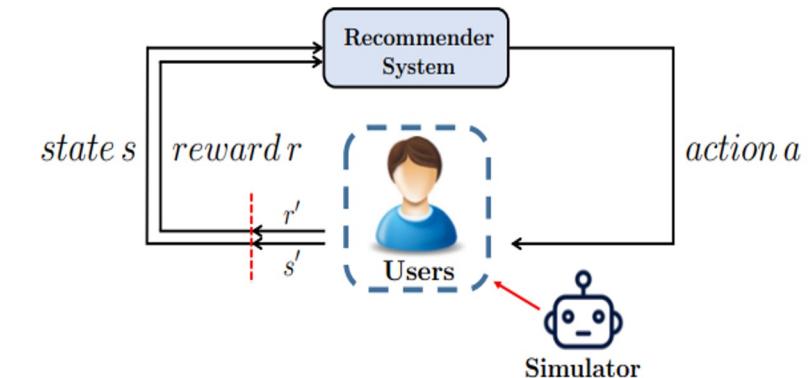
RecSys Algorithm	Rating Prediction	Item Recommendation	Implicit Feedback
User Average	x	x	
Item Average	x	x	
Slope One	x	x	
User Based KNN	x	x	
Item Based KNN	x	x	
Funk's SVD	x	x	
Most Popular		x	
ALS		x	x
ALS with Conjugate Gradient		x	x
BPR for Matrix Factorization		x	x

Simuladores de RL para Recomendación

Definición: Uno de los principales desafíos del Aprendizaje Reforzado (RL) en Sistemas Recomendadores es que un agente de RL necesita actualizar su política de recomendación a medida en que recibe **feedback de los usuarios**. Esto significa que entrenar el agente con datos históricos genera sesgos, ya que las **acciones** con las que se entrena no corresponden a la **política** actual del agente.

Por otro lado, entrenar desde un comienzo con usuarios reales perjudica la experiencia del usuario, ya que el agente se tardará en encontrar una política óptima.

Una de las soluciones que se ha propuesto para este problema es el uso de **Simuladores**. Un simulador busca replicar el comportamiento de usuarios reales frente a acciones, así permitiendo entrenar modelos de manera **offline**.



Simuladores de RL para Recomendación

- Algunos de los ambientes de simulación disponibles para RL en Sistemas Recomendadores son:
 1. RecoGym
 2. RecSim
 3. MARS-Gym
 4. PyRecGym
 5. VirtualTB
 6. Recsimu
 7. SOFA
- ¿en qué estado de actualización/desarrollo están?

Simuladores de RL para Recomendación

Algunas **propuestas** para el desarrollo de un proyecto son:

- Generar un Survey de las distintas alternativas, comparando su rendimiento y resultados.
- Evaluar efectividad de simulación replicando resultados reales en ambientes simulados.
- Proponer mejoras a ambientes existentes.

Motivación:

1. Existen diversas investigaciones respecto al uso de simuladores en Aprendizaje Reforzado, pero no existe un consenso respecto a la manera de evaluar la utilidad de estas herramienta ni un análisis exhaustivo de las distintas herramientas
2. Muchos de los modelos de simuladores tienen código abierto disponible para su uso y testeo.

RL: what are we learning?

On the Unexpected Effectiveness of Reinforcement Learning for Sequential Recommendation

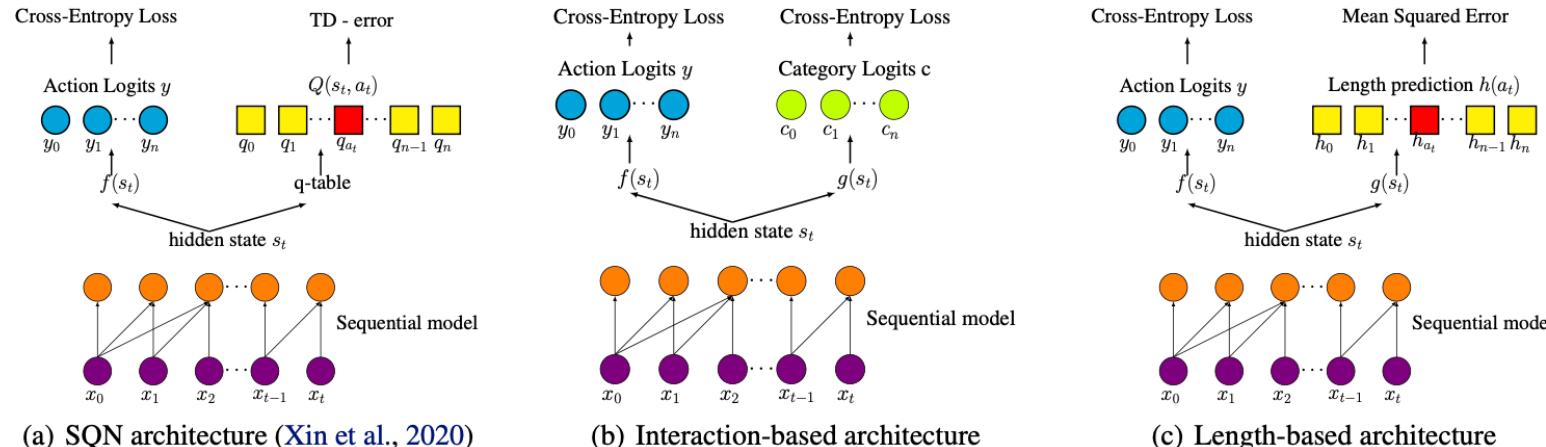


Figure 1. The SQN and proposed architectures

Table 2. List of features used for linear regression analysis.

Feature	Description	β
hist-length	Number of past user interactions in the sequence.	0.0638
fut-length	Number of future user interactions in the sequence.	0.0261
Q-Value (eval)	The expected return following the sequence in the history log.	0.0159
hist-buys	Number of items the user bought in past interactions.	0.0282
fut-buys	Number of items the user will buy in future interactions.	0.0110

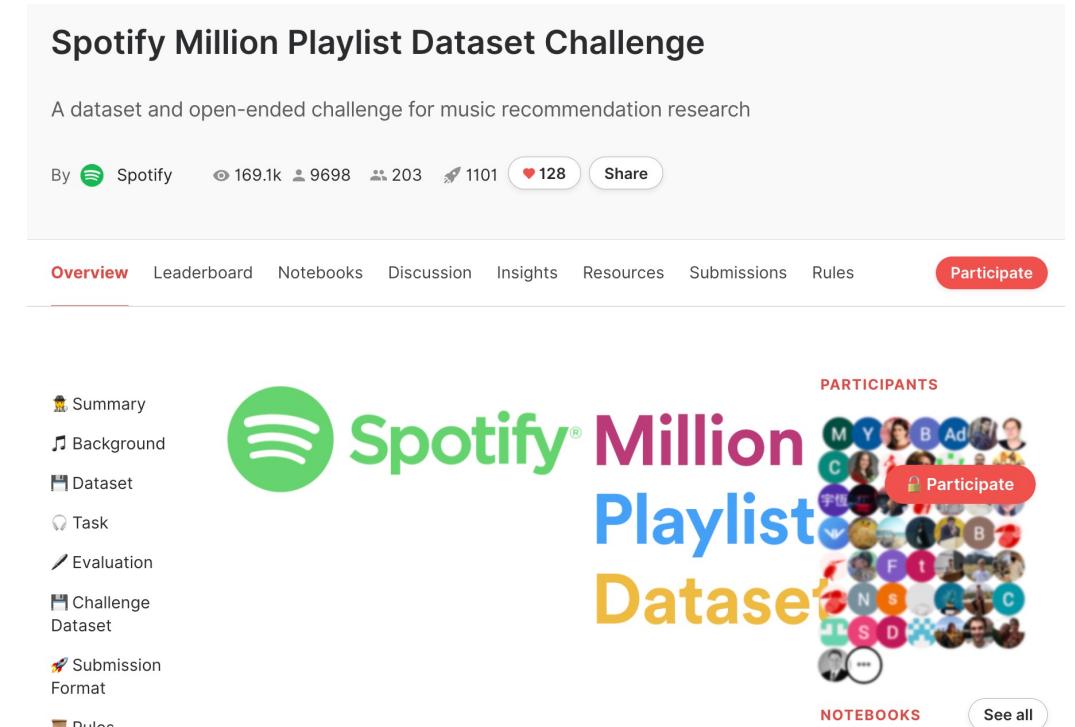
- Re-escribir código tensorflow a pytorch
- Correr experimentos en otros datasets secuenciales

Páginas de recursos

- Datasets
- Conferencias para revisar artículos e ideas recientes
- Páginas para revisar

Datasets

- Página Julian McAuley:
<https://cseweb.ucsd.edu/~jmcauley/datasets.html>
- Kaggle y similares:
<https://www.kaggle.com/search?q=recommender+in%3Adatasets+sortBy%3Adate>
- <https://www.aicrowd.com/challenges/spotify-million-playlist-dataset-challenge>
- RecSys/KDD/WSDM Challenges de años anteriores



The screenshot shows the main page of the Spotify Million Playlist Dataset Challenge on Kaggle. At the top, it displays the title "Spotify Million Playlist Dataset Challenge" and a subtitle "A dataset and open-ended challenge for music recommendation research". Below this, there are statistics: "By Spotify 169.1k views 9698 participants 203 teams 1101 submissions 128 likes Share". A red "Participate" button is located on the right. The page features a large Spotify logo and the challenge title in colorful, stylized letters. To the left, there's a sidebar with links: "PARTICIPANTS" (with a "Participate" button), "NOTEBOOKS" (with a "See all" button), "Summary", "Background", "Dataset", "Task", "Evaluation", "Challenge Dataset", "Submission Format", and "Rules".

Conference Challenges

- KDD cup 2024 <https://www.aicrowd.com/challenges/amazon-kdd-cup-2024-multi-task-online-shopping-challenge-for-langs>

amazon KDD Cup 2024

Multi-Task Online Shopping Challenge for LLMs

🏆 31,000 💲 10,500

By  Amazon Search

👁 49.6k

👤 998

👥 364

🚀 974

❤ 60

Share



Conferencias para revisar artículos e ideas

- Páginas de algunas conferencias definen scope (alcance) con temas relevantes, así como listas de papers aceptados, best papers, etc:
 - RecSys <https://recsys.acm.org/>
 - IUI <https://iui.acm.org/2023/>
 - UMAP <https://www.um.org/umap2023/>
 - SIGIR <https://sigir.org/sigir2023/submit/call-for-full-papers/>
 - KDD <https://kdd.org/kdd2023/>
 - CIKM <https://uobevents.eventsair.com/cikm2023/>

Resumenes de la conferencia RecSys

<https://www.urinieto.com/2020/09/recsys2020/>



ORIOL (URI) NIETO

PhD in Music Data Science



Carousel Personalization in Music Streaming Apps with Contextual Bandits

Cool application of contextual bandits in a "Carousel" music application, tested in a large-scale scenario by the good people at Deezer. They not only report how effective their model is, but they also release a publicly available dataset and an open-source carousel personalization environment to play with. The code and dataset should be here "by the end of September 2020." This makes me both happy and sad: happy because I see that I'm not alone in terms of doing research at the very last minute (it's September 28th and the repo is still empty); sad because the code isn't available yet and the world would be a much better place if researchers wouldn't do things at the very last minute.

Content-Collaborative Disentanglement Representation Learning for Enhanced Recommendation

Do you want to combine content and collaborative features together? Disentangle them first! This paper proposes a two-level disentanglement model that makes use of the KL divergence to ensure that the learned features are statistically independent. Results obtain state-of-the-art on 3 different datasets.

Exploring Longitudinal Effects of Session-based Recommendations

Interesting "reinforcement effect" investigation on music recommendation. The authors run several algorithms in a repeated manner and report how, in the long term, recommenders tend to pick smaller pools of songs because they keep recommending more and more similar tracks due to the reinforcement effect. By running a re-ranking method they are able to reduce this potential bias in the recommendations.

KRED: Knowledge-Aware Document Representation for News Recommendations

- Saito: Doubly Robust Estimator for Ranking Metrics with Post-Click Conversions
- Wang et al.: Causal Inference for Recommender Systems
- Guo et al. (Twitter): Deep Bayesian Bandits: Exploring in Online Personalized Recommendations

Dominant Topics: Biases, Fairness, Causality, Bandits and Reinforcement Learning

In my personal view, the outcry of recent years regarding the narrow-minded focus on accuracy was heard by the community. Acknowledging biases and developing de-biasing techniques, looking beyond correlation and trying to model causal effects as well as addressing fairness and accountability are among the dominant topics of the conference. I believe that the RecSys research community is much more aware of these topics than general society gives it credit for. However, my view is biased towards what I saw at the conference and not what happens in general behind each system. But there is also evidence that addressing these issues drives beneficial long-term business goals and is therefore grounded in industry's own interest.

<https://www.inovex.de/blog/recsys-2020-highlights/>

Resumenes RecSys 2021-2022

- RecSys 2021
 - <https://www.yusp.com/blog-posts/recsys-2021-impressions-summary/>
 - <https://www.the-odd-dataguy.com/2021/10/01/recsys-2021/>
- RecSys 2022
 - <https://eugeneyan.com/writing/recsys2022/>
 - <https://www.shaped.ai/blog/day-2-of-recsys2022-our-favorite-5-papers-and-talks>
 - <https://www.shaped.ai/blog/day-3-of-recsys2022-our-favorite-5-papers-and-talks>