Sesgo de Personalidad en Recomendación de Música: objetivos *Beyond-Accurracy*

Vicente Valencia
Pontificia Universidad Católica de Chile
Santiago, Chile
vmvalencia@uc.cl

Ignacia Gonzalez
Pontificia Universidad Católica de Chile
Santiago, Chile
migonzalez5@uc.cl

ABSTRACT

En este estudio, replicamos y extendemos el estudio *Personality Bias of Music Recommendation Algorithms* [4], en el que los investigadores encontraron sesgos en recomendación de música (en términos de nDCG@k and Recall@k) en tres algoritmos de recomendación a lo largo de los cinco grandes rasgos de la personalidad de los usuarios [1]. Agregamos tres métricas nuevas *Beyond-Accurracy*: *coverage*, *diversity* y *serendipity* y estudiamos el sesgo de personalidad en ellas. Además, cada métrica fue implementada en dos versiones diferentes para poder reafirmar los sesgos: una con información solo del *dataset* de [4] y otra con *features* de Spotify.

CONCEPTOS CLAVE

sesgo de personalidad, recomendación de música, sistemas recomendadores

Formato de Referencia:
Vicente Valencia and Ignacia Gonzalez. 2020. Sesgo de Personalidad en Recomendación de Música: objetivos *Beyond-Accurracy*. In *Proceedings of RecSys 2020-2*.

1 ESTADO DEL ARTE

Este trabajo construye sobre el trabajo del artículo *Personality Bias of Music Recommendation Algorithms* [4], el que es el primero y único hasta ahora en abordar sesgos que surgen en los cinco rasgos de personalidad. En tal estudio, se propone como pregunta de investigación: ¿existen sesgos en algoritmos de recomendación de música respecto a la personalidad de los usuarios a los que se les recomienda?. Para responder esta pregunta, los autores realizaron las siguientes tareas.

(1) Recolección de *tweets* de personas que compartieron canciones que escuchaban con la API de *streaming* de Twitter.
(2) Procesamiento programático de los *tweets* para extraer canciones, artistas y personalidades relacionados con cada usuario. Las personalidades fueron obtenidas con la la API *Personality Insights* de IBM. Éstas se representan en cinco dimensiones de acuerdo al modelo OCEAN: *Agreeableness*, *Conscientiousness*, *Extraversion*, *Neuroticism* y *Openness*
(3) Uso de distintos algoritmos de recomendación — SLIM, EASE y VAE — para recomendar canciones en un experimento *offline* con el *dataset* elaborado.
(4) División de los datos por cada dimensión de personalidad, creando grupos con valores altos (grupos *high*) separados de los grupos con valores bajos (grupos *low*). Por ejemplo, para la dimensión *Openness* (apertura a la experiencia), la mitad de los usuarios con mayor valor en esta dimensión queda en el grupo de apertura alta, mientras que la otra mitad en el grupo de apertura baja.
(5) Cómputo de dos métricas de rendimiento — NDCG@K y Recall@K — sobre cada uno de los grupos generados en el paso anterior. Nota: la validación en el entrenamiento se hace respecto a NDCG.
(6) Análisis de diferencias en los valores de las métricas para los diferentes grupos y algoritmos.

Los autores de esta investigación encontraron diferencias significativas en las métricas para los diferentes grupos. Sin embargo, esto solo habla de las dos métricas usadas y de los tres algoritmos elegidos. Idealmente, se deberían probar la mayor cantidad de métricas y algoritmos, pero no es posible ser tan exhaustivo en la práctica.

En la sección de trabajo futuro de [4], se menciona que se pretende incorporar nuevas métricas sobre las que se evaluaría el sesgo de personalidad de los algoritmos. En particular, los autores proponen agregar *diversity*, *serendipity* y *coverage*. Creemos que se propusieron estas porque son de una naturaleza algo más subjetiva y menos orientada a rendimiento que las métricas de precisión. Si los sesgos observados en [4] se replicaran sobre estas nuevas métricas, consolidaría aún más la hipótesis de los autores.

Nosotros tomaremos esta propuesta y la implementaremos sobre el código ya provisto por [4] en GitHub. Dos de nuestras preguntas de investigación (PIs) son:

PI1. ¿Existe sesgo de acuerdo a las nuevas métricas incorporadas?
PI2. Si es que hay sesgo, ¿es parecido al sesgo de las métricas anteriores?

Algo importante que hay que considerar es que hay muchas formas de calcular cualquiera de estas métricas nuevas. Nosotros proponemos que, usando *features* de artistas y canciones de la API de Spotify, se pueden elaborar métricas alternativas más dinámicas que, posiblemente, reaccionen de manera diferente a aquellas que no hagan uso de estas características o *features*.

Lo anterior nos conduce a una pregunta de investigación (PIs) adicional:

PI3. Las métricas alternativas que utilicen las *features* de Spotify ¿tienen un comportamiento diferente en relación al sesgo? ¿Cuál?

2 DATASET

El *dataset* es exactamente el mismo utilizado por [4]. Este está compuesto por 18.310 usuarios, 15.753 canciones y 395.056 interacciones. A continuación se describe el proceso para obtener las *features* de Spotify de las canciones de este dataset.

2.1 Canciones con las *features* de Spotify

Para la obtención de las features de las canciones del dataset realizamos varios pasos. El primero fue la creación de 40 listas de Spotify con 100 canciones cada una del dataset de [4]. Al ver que tomaría mucho tiempo agregar todas esas canciones manualmente, decidimos crear estas playlists utilizando la pagina web Tune My Music. Esta página web recibe un archivo CSV en donde se van agregando automáticamente las canciones a una playlist. Posteriormente usamos un codigo en jupyter notebook para la obtención de las features de cada canción de las antes mencionadas playlists a través de distintas funciones, la utilización de pandas y consultando la API de Spotify. La API de Spotify solo nos permite hacer 100 consultas de cada playlist tomando mucho tiempo en obtener todos los datos por lo que solo pudimos obtener las features de 4000 y no alcanzamos a obtener las features del dataset completo. En los experimentos, cuando una canción no tiene features asociadas, se ignora en las métricas Spotify descritas más adelante.

3 METODOLOGÍA

3.1 Herramientas

Los autores de [4] hicieron público en la plataforma GitHub el código usado para realizar los experimentos [3]. Este código utiliza el lenguaje de programación Python con las librerías Jypyter, Pytorch Lightning y Tensorboard, principalmente.

Sobre este código implementamos las métricas presentadas en 3.3: *coverage*, *diversity* y *serendipity*.

Respecto a nuestra pregunta de investigación PI3, consultamos la API Web de Spotify para obtener las características deseadas de las canciones presentes en el set de datos de [4].

3.2 Algoritmos

Como se está replicando y extendiendo un estudio, los algoritmos utilizados son los mismos que fueron usados en él. Estos son:

- SLIM (Sparse Linear Method) [5]
- EASE (Embarrassingly Shallow AutoEncoder) [7]
- VAE (Variational AutoEncoder) [2]

En este proyecto, los modelos ocupan un rol secundario. Esto no los hace menos importantes, solo no son el foco del trabajo, como puede evidenciarse en nuestras preguntas de investigación.

3.3 Métricas implementadas

Para decidir qué versiones de las siguientes métricas utilizar, nos referimos y basamos nuestras ideas en el *Handbook* de sistemas recomendadores [6]. A continuación se exponen las tres métricas implementadas, cada una en su versión simple y en su versión con las *features* de Spotify.

3.3.1 Coverage (Sección 8.3.3 del Handbook). La cobertura puede orientarse hacia usuarios o ítems. En el caso de los ítems, es una medida de la proporción de ítems que un recomendador recomienda en un experimento, simplificadamente hablando. Para el caso de los usuarios, se refiere a para cuántos usuarios el recomendador puede generar recomendaciones (o *buenas* recomendaciones).

Nosotros hemos implementado una versión simple de esta métrica — usualmente llamada cobertura de catálogo — orientada hacia ítems de la siguiente manera. Si el conjunto de ítems recomendados en un experimento e es I_e y el conjunto de todos los ítems existentes es I, entonces la cobertura de catálogo, simple o *plain* es $cov_p(e) = \frac{|I_e|}{|I|}$. Nótese que esta medida es holística y no se calcula por usuario.

Para la versión Spotify de la métrica, se consideró la distancia euclideana entre las dos canciones más lejanas (en el espacio de características de Spotify) recomendadas en un experimento. Esta distancia se normaliza por la distancia entre las dos canciones más distantes en el conjunto de todas las canciones. Si $S(c)$ es la representación de la canción c en el espacio de Spotify, $c1$ y $c2$ son las canciones más lejanas en I_e y $c3$ y $c4$ son las más lejanas en I, entonces la cobertura de Spotify es:

$$cov_s(e) = \frac{||S(c1) - S(c2)||}{||S(c3) - S(c4)||}$$

La intuición sobre la que se basa esta medida es que las distancias euclideanas en las que depende pueden interpretarse como díametros de esferas que envuelven o cubren las canciones.

3.3.2 Diversity (Sección 8.3.8 del Handbook). *Diversity* o diversidad intenta medir cuán diferentes o variados son los ítems que se le recomiendan a un usuario. También existen varias formas de calcular esta métrica. La idea general es computar una distancia $d(i, I)$ de un ítem i a un conjunto de ítems I ($i \notin I$) para todas las particiones de la forma $\{\{i\}, I\}$ de una lista de recomendación y luego promediar estas distancias para obtener una medida de diversidad para un usuario. Después, para un experimento en que se calcula diversidad de las listas de recomendación de todos los usuarios, se promedian las diversidades de todos para obtener una versión general de la métrica.

La fórmula de distancia utilizada es

$$d(c, C) = \frac{1 + n_C - n_{C,a(c)}}{1 + n_c}$$

en la que n_C es $\max_a n_{C,a}$, $n_{C,a}$ es la cantidad de canciones del artista a en C y $a(c)$ es el artista de la canción c. De esta manera resulta una métrica de distancia que es 0 cuando todos los artistas de las canciones en C son el mismo artista a y 1 cuando en C no hay una canción del artista $a(c)$.

La versión Spotify de esta métrica es idéntica, excepto en la forma de computar distancia. Esta sería:

$$d(c, C) = \frac{\sum_{c_i \in C} ||S(c) - S(c_i)||}{|C|}$$

Es decir, la distancia promedio a las canciones de C.

3.3.3 Serendipity (Sección 8.3.7 del Handbook). *Serendipity* trata de medir sorpresa en recomendación. Por ejemplo, quizás a alguien le guste música de rock clásico, pero solo ha escuchado pop actual en una plataforma de *streaming* de música. Una recomendacion de, por ejemplo, la canción *Back in Black* de AC/DC, sería una recomendación alta en *serendipity* para ese usuario.

Trait	Metric	Coverage			Diversity			Serendipity		
		All	High	Low	All	High	Low	All	High	Low
Agr.	Spot.	0.9749	0.8456	0.9749	0.0413	0.0386	0.0440	5.16e-4	6.11e-4	4.19e-4
	Plain.	0.3601	0.2844	0.2909	0.7779	0.7813	0.7744	7.80e-3	8.58e-3	7.01e-3
Con.	Spot.	0.9749	0.9749	0.8456	0.0413	0.0421	0.0405	5.16e-4	3.05e-4	7.38e-4
	Plain.	0.3601	0.2888	0.2892	0.7779	0.7817	0.7738	7.80e-3	7.07e-3	8.57e-3
Ext.	Spot.	0.9749	0.9749	0.8456	0.0413	0.0425	0.0402	5.16e-4	4.94e-4	5.36e-4
	Plain.	0.3601	0.2857	0.2874	0.7779	0.7806	0.7753	7.80e-3	8.33e-3	7.31e-3
Neu.	Spot.	0.9749	0.8456	0.9749	0.0413	0.0385	0.0441	5.16e-4	5.90e-4	4.40e-4
	Plain.	0.3601	0.2773	0.2885	0.7779	0.7807	0.7750	7.80e-3	8.35-3	7.25e-3
Ope.	Spot.	0.9749	0.9749	0.8456	0.0413	0.0449	0.0376	5.16e-4	5.91e-4	4.39e-4
	Plain	0.3601	0.2973	0.2785	0.7779	0.7841	0.7715	7.80e-3	8.16e-3	7.44e-3

Tabla 1: Métricas *Beyond-Accuracy*@50 para el modelo Struct-VAE

En este trabajo, *serendipity* se implementó dividiendo la *ground truth* o las etiquetas en dos grupos: uno de canciones observadas y otro de canciones ocultas. Al momento de *testing*, por cada canción recomendada para un usuario que esté en la lista oculta, se calcula su distancia o similaridad a la lista observada, de manera que la *serendipity* aumenta más mientras más lejos esté o menos similar sea una canción recomendada oculta que la lista observada.

La medida de distancia para esta métrica es la misma que para diversidad, en la versión simple usando la distancia de la versión simple de diversidad y en el cado de Spotify, usando la distancia de la versión Spotify de diversidad. Lo fundamentalmente distinto aquí es el uso de la partición (parte oculta y parte observada) de las etiquetas. La distancia ya no es calculada entre las canciones recomendadas, sino que entre ellas y la parte observada de la partición.

4 RESULTADOS

En la tabla 1 pueden observarse los valores para todas las métricas para cada rasgo de personalidad. Lo importante aquí son las direcciones de los sesgos. La mayoría de las direcciones coinciden. Por ejemplo la dirección del sesgo en cobertura para Apertura a la Experiencia u *Openness* coincide para las dos versiones de la métrica, pero, en otros casos, como para diversidad en neuroticismo, esta disminuye de grupo alto a grupo bajo para la versión simple, pero aumenta para la versión Spotify.

A modo general, sin embargo, las direcciones se mantienen. También se mantienen para diferentes valores de K y para los otros dos modelos.

5 CONCLUSIONES

El sesgo de personalidad existe y parece prevalecer a través de diferentes algoritmos de recomendación y diferentes maneras de calcular una misma métrica. Sin embargo, hay algunos casos en que la dirección del sesgo de una métrica se invierte en sus dos versiones. Esto es un aspecto que requiere más investigación. Por ejemplo, podrían agregarse *tests* estadísticos para determinar cuán significativa es cada diferencia (cada sesgo).

También somos conscientes de que es muy difícil presentar la cantidad de resultados que tenemos en tablas. La tabla 1 solo muestra los valores para un solo modelo y un solo valor de K. Para mejorar el análisis de resultados, sería deseable tener una visualización acorde que permita agrupar o presentar las métricas de manera agregada.

Volviendo al sesgo, es interesante notar que este puede ser diferente (cambiar de dirección) para cada métrica. Por esta misma razón es que es muy importante analizar varias métricas y en diferentes versiones si se quiere comprender a cabalidad los sesgos. Este estudio avanza en esa dirección.

REFERENCIAS

[1] Lewis R Goldberg. "The structure of phenotypic personality traits." In: *American psychologist* 48.1 (1993), p. 26.

[2] Dawen Liang et al. "Variational autoencoders for collaborative filtering". In: *Proceedings of the 2018 World Wide Web Conference*. 2018, pp. 689–698.

[3] Alessandro B. Melchiorre, Eva Zangerle, and Markus Schedl. *GitHub repository: Personality Bias in Music Recommendation*. Aug. 2020. URL: https://github.com/CPJKU/pers_bias.

[4] Alessandro B. Melchiorre, Eva Zangerle, and Markus Schedl. "Personality Bias of Music Recommendation Algorithms". In: *Fourteenth ACM Conference on Recommender Systems*. RecSys '20. Virtual Event, Brazil: Association for Computing Machinery, 2020, pp. 533–538. ISBN: 9781450375832. DOI: 10.1145/3383313.3412223. URL: https://doi.org/10.1145/3383313.3412223.

[5] Xia Ning and George Karypis. "Slim: Sparse linear methods for top-n recommender systems". In: *2011 IEEE 11th International Conference on Data Mining*. IEEE. 2011, pp. 497–506.

[6] Francesco Ricci et al. *Recommender Systems Handbook*. 1st. Berlin, Heidelberg: Springer-Verlag, 2010. ISBN: 0387858199.

[7] Harald Steck. "Embarrassingly shallow autoencoders for sparse data". In: *The World Wide Web Conference*. 2019, pp. 3251–3257.