Implementación de RecGan en Anime

Verner Codoceo
Pontificia Universidad Católica de Chile
Santiago, Chile

Dunkan Torres
Pontificia Universidad Católica de Chile
Santiago, Chile

Marcelo Escudero
Pontificia Universidad Católica de Chile
Santiago, Chile

ABSTRACT

Modelar las variables latentes de la evolucion temporal en los gustos de los usuarios es una cualidad buscada en los sistemas recomendadores. Permitiendo al sistema tener mayor flexiblidad para modelar tendencias temporales. Usando un dataset de anime, en este trabajo buscamos implementar un simil del modelo RecGAN, usando una Gated Recurrent Unit (GRU) general para capturarestas variables latentes mencionadas y usando Generative Adversarial Network (GAN). Para evaluar nuestro trabajo el modelo será comparado con un ALS, en las métricas novelty y diversity. Los resultados indican que ALS no es un modelo ideal para este tipo dedataset y que trabajar con un modelo RecGAN es algo difícil de hacer.

KEYWORDS

Datasets, Neural Networks, GAN, GRU

1 INTRODUCCIÓN

1.1 Contexto del problema

En el mundo de la animación japonesa (anime) existen muchos sitios de streaming para ver estas animaciones. Sin embargo, no todas estas animaciones son conocidas o han sido vistas por mucha gente. Esto supone que dentro de estas páginas o en algunos foros, se hagan discusiones respecto a recomendaciones de animes ya que hay algunas personas que se están iniciando en este mundo o simplemente ya no saben que ver.

1.2 Problema y su justificación

Tras una búsqueda de los principales sitios de streaming, se descubrió que no existen recomendadores de anime o simplemente son filtros básicos aplicados sobre un dataset. Algunas a destacar son:

- Crunchyroll: un servicio de streaming pagado de anime.
- MyAnimeList: una página llevar track de los animes o mangas que uno está siguiendo. Una suerte de IMDb para anime.
- Animeblix: un sitio de streaming que tiene un recomendador que supuestamente usa machine learning, pero que realmente se ve como un Most Popular filtrado por género.

Resulta extraño que no exista un recomendador dentro de estas plataformas dado el aumento en su popularidad. Cada vez más gente está viendo anime y quiere encontrar nuevos animes que puedan ser de su agrado. Por esta razón, es que nos planteamos realizar un recomendador para anime y así suplir la necesidad que se ha generado.

IIC3613, Diciembre 2020, Santiago, Chile

1.3 Objetivos

Elaborar un sistema recomendador que en base a la actividad de un usuario genere una lista de recomendaciones para el usuario. Esta actividad contempla los animes vistos por el usuario y la fecha en que se vieron.

1.4 Solución propuesta

Este trabajo se basó en el paper "RecGAN: Recurrent Generative Adversarial Networks for Recommendation Systems" [1], el cual enfatiza la importancia en modelar latent features detrás de la evolución temporal de preferencias del usuario y el estado del ítem. Estas latent features son utilizadas para hacer recomendaciones relevantes, considerando también modas/tendencias de usuarios e ítems mediante una Red Adversaria Generativa Recurrente (RecGAN), **propuesta para capturar características latentes de usuarios e ítems observables de perfiles temporales a corto y largo plazo.**

2 TRABAJO RELACIONADO

En [1] se propone y se mide el rendimiento de un sistemas recomendadores en base a una RecGAN sobre un *dataset* de películas. El modelo utilizado funciona similar al propuesto en este estudio, solo con la diferencia de hacer uso de una GRU modificada y de la técnica utilizada para recomendar a partir de la matriz generada por el *generator*. En [2] se detallan los elementos más relevantes de un anime a tener en cuenta al momento de recomendar anime.

3 DATASET

El dataset obtenido desde Kaggle consta de 9 archivos .csv, entre los cuales se consideraron útiles los siguientes:

(1) **'animelists_cleaned.csv':** Este archivo contiene las interacciones de los usuarios con determinados animes y sus timestamps. Además, este archivo estaba anteriormente procesado para eliminar datos con errores o datos repetidos.
Se revisó que no tuviera valores vacíos y se eliminaron las columnas que no resultaban importantes, dejando solamente (userId, itemId, rating, timestamp) donde timestamp es el momento de la última edición del usuario del estado de este anime.
Por último como los modelos RNN, GRU, LSTM usan la columna de timestamp, se puso énfasis en que esta tuviera un rango acotado. Se descubrió que aquellos animes donde esta columna no se completaba eran rellenados por default con una fecha en 1970, por ello se eliminaron todos los usuarios que tuvieran al menos un anime de esta época, como la base de datos es muy grande (~ 2.7GB) no se perdió tanta información.

(2) **'anime_cleaned.csv':** Este archivo contiene información del anime como por ejemplo género, estado, fecha de emisión, capítulos, etc.
(3) **'users_cleaned.csv':** Este archivo contiene información de los usuarios. Anteriormente fue procesado para eliminar a los usuarios que les falta información importante o que estuviesen repetidos.

Se generó un archivo llamado actualmente *230_users.csv* el cual contiene las interacciones de 230 usuarios usuarios sobre 5.518 animes diferentes y contiene un total de 92.595 interacciones. La razón por la cual se creó este archivo es debido a que se tuvo problemas utilizando el dataset completo ya que contenía demasiada información.

Este archivo contiene las columnas (userId, itemId, rating, timestamp). Al archivo generado se le transformó la columna timestamp a un numero desde 0 que represente el semestre en el que se actualizo por última vez el rating de un anime.

4 METODOLOGÍA UTILIZADA

El primer paso en el estudio fue buscar un *dataset* en el que pudieramos trabajar para cumplir con el objetivo propuesto. Una vez encontrado, se analizó la información disponible para asegurar su calidad antes de ingresarla a cualquier modelo. En este proceso, se encontró los errores en el *dataset* descritos anteriormente.

Luego de preprocesar los datos, se procedió a generar recomendaciones mediante un modelo clásico, ALS, con el fin de tener una base de comparación para las recomendaciones generadas con RecGAN. En estas recomendaciones se midió tanto el *novelty* como *diversity* de los *items* obtenidos.

Una vez obtenidas las recomendaciones con ALS, el siguiente paso fue estudiar sobre RecGAN, modelo propuesto por el equipo docente, con el fin de aplicarlo en nuestro *dataset*. Debido a la falta de detalles en algunas implementaciones de [1], se debió adaptar el modelo bajo suposiciones del equipo de desarrollo. Esto llevó a cambiar la salida del generador desde una lista de recomendaciones a un $Y^u = [y_1^u, ..., y_t^u, ..., y_T^u]$, donde y_t^u es un vector con todos los *ratings* del usuario u en la temporada t.

Una vez adaptado el generador, se procedió a entrenarlo junto al discriminador, el cual corresponde a una RNN basada en GRU que debía indicar si el Y que se le entregaba era uno generado o uno proveniente del *dataset*. Teniendo ambas RNN entrenadas, es posible generar $Y's$ lo suficientemente veraces para poder basar recomendaciones en base a ellas. Para un usuario a recomendar u se genera un conjunto de $Y's$ y se obtiene el más similar al Y del usuario. Luego de esto, basta con elegir los *items* con mejor *rating* del tiempo del que se quiere obtener la recomendación para obtener los *items* recomendados.

5 ANÁLISIS DE PARÁMETROS

En el caso de ALS se entrenaron 18 modelos distintos, es decir, con distintos parámetros. La idea detrás de esto era tener muchas opciones de dicho recomendador e intentar encontrar aquella configuración que nos diera los mejores resultados. Una vez obtenida la mejor versión de ALS, se compararían las métricas obtenidas por dicho modelo con las métricas obtenidas por el RecGan para ver efectivamente si había una diferencia de desempeño en cuanto a la novedad y diversidad de las recomendaciones obtenidas. Los valores de los parámetros fueron:

- **Factores latentes:** 50, 100, 200.
- **Regularización:** 0.1, 0.01.
- **Iteraciones:** 10, 50, 100.

Con respecto al model RecGan, se probaron varias combinaciones de parámetros con el fin de lograr una convergencia del modelo. Además, se aplicaron ciertas estrategias para ayudar a la convergencia del modelo. Finalmente, los parámetros fueron elegidos en base a la rápidez y aproximación a lo que esperaba realizar. Estos se listan a continuación:

- **Número de épocas:** 5.
- **Tamaño del batch:** $\frac{\text{número de usuarios}}{4}$.
- **Learning rate:** 0.0001.
- **Número de capas:** 2.
- **Tamaño del input:** número de ítems.
- **hidden_size:** número de ítems.

6 RESULTADOS OBTENIDOS

En cuanto a los resultados obtenidos, solo pudimos hacer la medición de las métricas para los 18 modelos de ALS que fueron entrenados. Estas métricas fueron calculadas para 20 y 50 recomendaciones por usuario (ver Figura 1).

Rec.	Factores	Reg.	Iteraciones	Diversity	Novelty
20	50	0.01	50	0.004513	0.040130
50	100	0.01	50	0.004506	0.040130

Figure 1: Tabla con mejores resultados de las métricas.

Tras haber realizado el estudio se cree que estos resultados fueron bajos debido a la forma en la que se construyó la matriz de interacciones. Esta matriz fue construida utilizando el id máximo de un anime. Esto generó una matriz mucho más vacía y por lo mismo, se cree esto se debe al problema conocido como *cold start* que sufren algunos sistemas recomendadores.

Para el modelo RecGan no se obtuvieron resultados. Sin embargo, se cree importante mostrar como fueron los resultados de las funciones de pérdida tanto para el discriminador como para el generador a lo largo de las 5 épocas (ver Tabla 1).

7 CONCLUSIONES

En este trabajo nos proponiamos comparar RecGAN con ALS, con las metricas de novelty y diversity, esto no se logro debido a que el modelo RecGAN no convergió. Por lo tanto, las recomendaciones generadas por el generador del modelo no eran fiables. Algunas razones que creemos fueron las causantes de este problema son:

- Vanishing Gradients: nosotros usamos binary cross entropy loss, pero se ha reportado de que GAN puede funcionar mejor con Wasserstein loss o con Modified minimax loss.
- GRU personalizada: no implementamos la GRU personalizada que usa Relu como funcion de activacion, esta ayuda a la convergencia del modelo segun el paper
- BPTT: no pudimos hacer un BPTT completo ya que se disparaba el tiempo que demoraba en entrenar el algoritmo (no logramos que terminara de entrenar)

Época	Batch	Loss D	Loss G	D(x)
0	0	1.3878	0.7920	0.4999
0	1	1.3051	1.1162	0.4998
0	2	1.1137	1.5764	0.4992
1	0	0.9510	2.0752	0.4980
1	1	0.8325	2.5633	0.5066
1	2	0.7163	3.0189	0.5451
2	0	0.5050	3.4427	0.7113
2	1	0.4291	3.6915	0.8683
2	2	0.4257	3.7935	0.8928
3	0	0.4139	3.8220	0.8934
3	1	0.4005	3.8421	0.8834
3	2	0.3916	3.8911	0.8700
4	0	0.3867	3.9769	0.8595
4	1	0.3830	4.0893	0.8552
4	2	0.3792	4.2112	0.8566
5	0	0.3755	4.3291	0.8605
5	1	0.3723	4-4369	0.8640
5	2	0.3695	4.5354	0.8654

Table 1: Tabla con información del entrenamiento del modelo RecGan por épocas.

8 BIBLIOGRAFÍA

[1] Homanga Bharadhwaj, Homin Park, and Brian Y. Lim. 2018. RecGAN: Recurrent Generative Adversarial Networks for Recommendation Systems. In *Twelfth ACM Conference on Recommender Systems (RecSys '18), October 2–7, 2018, Vancouver, BC, Canada.* ACM, New York, NY, USA, Article 4, 5 pages. https://doi.org/10.1145/3240323.3240383

[2] Hyerim Cho, Marc L. Schmalz, Stephen A. Keating, and Jin Ha Lee. 2017. Information needs for anime recommendation: analyzing anime users' online forum queries. In *Proceedings of the 17th ACM/IEEE Joint Conference on Digital Libraries (JCDL '17). IEEE Press, 305–306.*