Proyecto de Sistemas de Recuperación de Información (SRI) - Recomendación Secuencial Colaborativa

Paula Silva Lara (C-312), Ricardo Cápiro Colomar (C-312), and Ariel González Gómez (C-312)

Facultad de Matemática y Computación (MATCOM), Universidad de La Habana, Cuba

Abstract. Different from the conventional recommender systems (RSs) including collaborative filtering and content-based filtering, SRSs try to understand and model the sequential user behaviors, the interactions between users and items, and the evolution of users' preferences and item popularity over time. SRSs involve the above aspects for more precise characterization of user contexts, intent and goals, and item consumption trend, leading to more accurate, customized and dynamic recommendations.
This project investigates the application of Recurrent Neural Networks (RNN) for collaborative sequential recommendation, and use it to predict the next most likely product a user will purchase from an e-commerce platform, specifically focusing on electronics products. A dataset from Kaggle was utilized to train a RNN model with Gated Recurrent Unit (GRU) as gating mechanism. The project evaluates the model's performance and accuracy using various metrics. The code is available at: `https://github.com/Pola03/SRI-Project/tree/main`

Keywords: Sequencial Recommendation System · Recurrent Neural Network · Gated Recurrent Unit

1 Introducción

Los sistemas de recomendación secuencial (SRS) sugieren artículos que podrían ser de interés para un usuario, principalmente modelando las dependencias secuenciales entre las interacciones usuario-artículo (por ejemplo, ver o comprar artículos en una plataforma de compras en línea) en una secuencia. Los sistemas de recomendación tradicionales (RS), incluyendo los basados en contenido y los de filtrado colaborativo, modelan las interacciones usuario-artículo de forma estática y solo pueden capturar las preferencias generales del usuario. En contraste, los SRS tratan las interacciones usuario-artículo como una secuencia dinámica y consideran las dependencias secuenciales para capturar las preferencias actuales y recientes del usuario, lo que permite obtener recomendaciones más precisas. Para mejorar la comprensión de los SRS, a continuación se presentan las motivaciones que los sustentan.

Las interacciones usuario-artículo están esencialmente interconectadas de forma secuencial. En el mundo real, los comportamientos de compra de los usuarios generalmente suceden sucesivamente en una secuencia, en lugar de forma aislada.

Tanto las preferencias de los usuarios como la popularidad de los artículos son dinámicas en lugar de estáticas con el tiempo. De hecho, la preferencia y el gusto de un usuario pueden cambiar con el tiempo. Estas dinámicas son de gran importancia para la creación de perfiles precisos de usuarios o artículos y, en consecuencia, para obtener recomendaciones más precisas, y solo pueden ser capturadas por los SRS.

Las interacciones usuario-artículo generalmente ocurren bajo un cierto contexto secuencial. Diferentes contextos conducen a diferentes interacciones de los usuarios con los artículos, lo que, sin embargo, a menudo es ignorado por los RS tradicionales como el filtrado colaborativo. En contraste, un SRS considera las interacciones secuenciales anteriores como un contexto para predecir qué artículos serán interactuados en el futuro cercano. Como resultado, es mucho más fácil diversificar los resultados de la recomendación al evitar recomendar repetidamente aquellos artículos idénticos o similares a los que ya han sido elegidos.

1.1 Antecedentes

Para brindar una visión general del progreso técnico en los sistemas de recomendación secuencial (SRS), se resume y discute brevemente el progreso de la investigación en SRS desde una perspectiva técnica en esta sección. En particular, primero se presenta una categorización de todos los enfoques para los SRS desde la perspectiva técnica y luego se destaca brevemente el progreso reciente en cada categoría.

- **Modelos de Secuencia Tradicionales para SRS**: son soluciones intuitivas para los SRS al aprovechar su fuerza natural en el modelado de dependencias secuenciales entre las interacciones usuario-ítem en una secuencia. Incluyen:
  - **Minería de Patrones Secuenciales (Sequential pattern mining)**: estos primero extraen patrones frecuentes en los datos de secuencia y luego utilizan los patrones extraídos para guiar las recomendaciones subsecuentes.
  - **Modelos de Cadena de Markov (Markov chain models)**. se usan para modelar las transiciones sobre las interacciones usuario-ítem en una secuencia, para la predicción de la próxima interacción.
- **Modelos de Representación Latente para SRS**: primero aprenden una representación latente de cada usuario o ítem, y luego predicen las interacciones usuario-ítem subsecuentes utilizando las representaciones aprendidas. Como resultado, se capturan dependencias más implícitas y complejas en un espacio latente, lo que beneficia en gran medida las recomendaciones. Incluyen:

  - **Máquinas de Factorización (Factorization machines)**: los SRS basados en estas generalmente utilizan la factorización de matrices o la factorización de tensores para factorizar las interacciones usuario-ítem observadas en factores latentes de usuarios e ítems para las recomendaciones. A diferencia del filtrado colaborativo (CF), la matriz o el tensor a factorizar se compone de interacciones en lugar de las clasificaciones en CF.
  - **Embeddings**: los SRS basados en Embeddings aprenden representaciones latentes para cada usuario e ítem para las recomendaciones subsecuentes al codificar todas las interacciones usuario-ítem en una secuencia en un espacio latente. Específicamente, algunos trabajos toman las representaciones latentes aprendidas como la entrada de una red para calcular posteriormente una puntuación de interacción entre usuarios e ítems, o acciones sucesivas de los usuarios, mientras que otros trabajos las utilizan directamente para calcular una métrica como la distancia euclidiana como la puntuación de interacción.

– **Modelos de Redes Neuronales Profundas para SRSs**: las redes neuronales profundas tienen una fortaleza natural para modelar y capturar las relaciones integrales entre diferentes entidades (por ejemplo, usuarios, artículos, interacciones) en una secuencia, y por lo tanto, dominan casi por completo los SRSs en los últimos años. Generalmente, se pueden dividir en dos subclases: SRSs construidos sobre redes neuronales profundas básicas y SRSs construidos sobre redes neuronales profundas con algunos modelos avanzados incorporados.

  - **Redes Neuronales Profundas Básicas (Basic Deep Neural Networks)**: las más comúnmente usadas para SRSs son las redes neuronales recurrentes (RNN) debido a su fortaleza natural en el modelado de secuencias, pero también tienen defectos. Recientemente, las redes neuronales convolucionales (CNN) y las redes neuronales de grafos (GNN) también se han aplicado en SRSs para compensar los defectos de las RNN. A continuación, presentamos los SRSs construidos sobre estas tres redes neuronales profundas, respectivamente.
    * **SRSs basados en RNN**: Dada una secuencia de interacciones históricas entre usuario-artículo, un SRS basado en RNN intenta predecir la próxima interacción posible modelando las dependencias secuenciales sobre las interacciones dadas. A excepción de la RNN básica, también se han desarrollado RNN basados en memoria a largo plazo (LSTM) y unidad recurrente con puerta (GRU) para capturar las dependencias a largo plazo en una secuencia. En los últimos años, hemos sido testigos de la prosperidad de los SRSs basados en RNN y dominan la investigación sobre los SRSs basados en aprendizaje profundo o incluso sobre la totalidad de los SRSs. Además de la estructura básica de RNN, se propusieron algunas variantes para capturar dependencias más complejas en una secuencia, como RNN jerárquicas.

  - **SRSs basados en CNN**: A diferencia de RNN, dada una secuencia de interacciones entre usuario-artículo, una CNN primero coloca todas las incrustaciones de estas interacciones en una matriz y luego trata dicha matriz como una "imagen" en el espacio temporal y latente. Finalmente, una CNN aprende patrones secuenciales como características locales de la imagen utilizando filtros convolucionales para las recomendaciones posteriores.
  - **SRSs basados en GNN**: Recientemente, con el rápido desarrollo de GNN, se han diseñado SRSs basados en GNN para aprovechar GNN para modelar y capturar las transiciones complejas sobre las interacciones entre usuario-artículo en una secuencia. Normalmente, primero se construye un grafo dirigido sobre los datos de la secuencia tomando cada interacción como un nodo en el grafo, mientras que cada secuencia se mapea a un camino. Luego, las incrustaciones de usuarios o artículos se aprenden en el grafo para incrustar relaciones más complejas en todo el grafo.

- **Modelos Avanzados**: Para abordar las limitaciones de los SRSs construidos sobre estructuras de redes neuronales básicas, algunos modelos avanzados generalmente se combinan con algún tipo de redes neuronales profundas básicas (por ejemplo, RNN, CNN) para construir SRSs más potentes que puedan abordar desafíos particulares. A continuación, presentamos tres modelos avanzados que se utilizan comúnmente en SRSs.
  - **Modelos de atención (Attention models)**: Los modelos de atención se emplean comúnmente en SRSs para enfatizar las interacciones realmente relevantes e importantes en una secuencia mientras se restan importancia a las irrelevantes para la próxima interacción. Se incorporan ampliamente a las redes superficiales y RNN para manejar secuencias de interacción con ruido.
  - **Redes de memoria (Memory networks)**: Las redes de memoria se introducen en SRSs para capturar las dependencias entre cualquier interacción histórica entre usuario-artículo y la siguiente directamente mediante la incorporación de una matriz de memoria externa. Dicha matriz permite almacenar y actualizar las interacciones históricas en una secuencia de manera más explícita y dinámica para mejorar la expresividad del modelo y reducir la interferencia de las interacciones irrelevantes. Además, algunos trabajos incorporan una red de memoria clave-valor para almacenar y actualizar la información correspondiente de la base de conocimiento de los artículos interactuados en una secuencia para aprender la preferencia de nivel de atributo para la mejora de las recomendaciones.
  - **Modelo de mezcla (Mixture models)**: Un SRS basado en un modelo de mezcla combina diferentes modelos que sobresalen en la captura de diferentes tipos de dependencias para mejorar la capacidad del modelo completo en la captura de diversas dependencias para obtener mejores recomendaciones.

2 Preliminares

2.1 Formulación del problema

En una tarea de recomendación secuencial, se tiene un conjunto de usuarios $\mathcal{U} = \{u_1, u_2, ..., u_{|\mathcal{U}|}\}$ que han tenido interacciones históricas con un conjunto de artículos $\mathcal{V} = \{v_1, v_2, ..., v_{|\mathcal{V}|}\}$. Entre estos usuarios, el usuario i-ésimo tiene una secuencia de artículos preferidos denotada como $s_i = [v_1^{(i)}, v_2^{(i)}, ..., v_{n_i}^{(i)}]$, donde n_i es la longitud de la lista de artículos con la que el usuario i-ésimo interactúa. El objetivo es diseñar un marco eficiente y predecir el siguiente artículo con el que el usuario tiene mayor probabilidad de interactuar.

2.2 Redes Neuronales Recurrentes

Las Redes Neuronales Recurrentes están ideadas para modelar datos secuenciales de longitud variable. La principal diferencia entre las RNNs y los Feedforward Deep Models es la existencia de un estado oculto interno en las unidades que componen la red. Los RNN estándar actualizan su estado oculto h utilizando la siguiente función de actualización:

$$h_t = g(Wx_t + Uh_{t-1}) \tag{1}$$

Donde g es una función suave y acotada, como una función sigmoidea logística, x_t es la entrada de la unidad en el momento t. Una RNN genera una distribución de probabilidad sobre el siguiente elemento de la secuencia, dado su estado actual h_t.

2.3 Unidades Recurrentes Cerradas (GRU)

Las Unidades Recurrentes Cerradas (Gated Recurrent Units (GRUs)) son un modelo más elaborado de unidad RNN que tiene como objetivo abordar el problema del vanishing gradient. Las puertas GRU esencialmente aprenden cuándo y en qué medida actualizar el estado oculto de la unidad. La activación del GRU es una interpolación lineal entre la activación anterior y la activación candidata $\hat{h_t}$:

$$h_t = (1 - z_t)h_{t-1} + z_t\hat{h_t} \tag{2}$$

donde la puerta de actualización viene dada por:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{3}$$

mientras que la función de activación candidata $\hat{h_t}$ se calcula de manera similar:

$$\hat{h_t} = \tanh(Wx_t + U(r_t \odot h_{t-1})) \tag{4}$$

y finalmente la puerta de reinicio r_t viene dada por:

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{5}$$

GRU utiliza la puerta de actualización para controlar el volumen de información retenida de los estados ocultos previos en el paso de tiempo actual, mientras que la puerta de reinicio controla la información que debe olvidarse. El módulo GRU (Unidad Recurrente con Puerta), equipado con puertas de actualización y reinicio en celdas GRU secuenciales, es hábil para capturar las relaciones entre los artículos a lo largo de la secuencia, mientras mantiene una complejidad computacional relativamente baja. Sin embargo, la información secuencial en GRU no se puede interactuar, y cada estado oculto se codifica principalmente a partir de elementos precedentes, lo que restringe la capacidad representativa de los modelos de recomendación basados en GRU, particularmente en la captura de dependencias complejas de artículos en toda la secuencia.

3 Descripción de la propuesta

En esta sección se explica el flujo del trabajo, y se presenta el modelo de red neuronal como propuesta de solución, dando una explicación breve de su arquitectura, funcionamiento y flujo de datos, seguida de una exploración más profunda de cada uno de sus componentes técnicos.

3.1 Flujo de la red

El flujo de la red se puede dividir en los siguientes pasos:

1. **Entrada**: La red recibe cuatro entradas para cada usuario:
   - *input_id*: Una secuencia de IDs de productos que el usuario ha comprado previamente.
   - *input_brand*: Una secuencia de marcas de los productos comprados previamente.
   - *input_cat*: Una secuencia de categorías de los productos comprados previamente.
   - *input_price*: Una secuencia de precios de los productos comprados previamente.
2. **Embeddings**: Cada una de las entradas categóricas (ID, marca, categoría) se convierte en un vector denso usando capas de embeddings:
   - *embedding_id*: Convierte los IDs de producto en vectores de dimensión *embedding_dim*.
   - *embedding_brand*: Convierte las marcas de producto en vectores de dimensión *embedding_dim*.
   - *embedding_cat*: Convierte las categorías de producto en vectores de dimensión *embedding_dim*.
3. **Concatenación**: Los vectores de embeddings para cada paso de tiempo (cada producto comprado previamente) se concatenan junto con el precio. Esto crea una secuencia de vectores de entrada para la GRU.

4. **Masking (opcional)**: Si se utiliza, la capa de masking ignora los valores de relleno en la secuencia de entrada. Esto es necesario porque las secuencias de entrada tienen diferentes longitudes (número de productos comprados previamente), y se rellenan con ceros para que todas tengan la misma longitud.
5. **GRU**: La capa GRU procesa la secuencia de entrada, teniendo en cuenta las dependencias entre los elementos de la secuencia. La GRU mantiene un estado interno que se actualiza con cada paso de tiempo, capturando la información de la secuencia hasta ese punto.
6. **Dropout**: Se aplica una capa de dropout al estado final de la GRU para prevenir el sobreajuste.
7. **Capa Densa**: La capa densa final toma el estado final de la GRU y produce una distribución de probabilidad sobre todos los productos posibles. La función de activación softmax asegura que las probabilidades sumen 1.
8. **Salida**: La salida de la red es la probabilidad de que el usuario compre cada producto. La red predice el producto con la mayor probabilidad como el siguiente producto que el usuario comprará.

3.2 Explicación de componentes principales

Capas de Embeddings: La capa de embedding es esencial para que una red neuronal pueda "entender" datos categóricos, como los IDs de producto, marcas y categorías, que no son directamente procesables por una red neuronal. La capa de embedding funciona como un diccionario que asigna a cada palabra (en este caso, cada ID de producto, marca o categoría) un vector numérico de dimensiones fijas. Este vector, llamado embedding, representa la "definición" de la palabra en un espacio multidimensional.

La capa de embedding es una tabla de búsqueda que mapea cada palabra (categoría) a un vector de tamaño fijo. La red neuronal aprende la posición de cada vector en este espacio durante el entrenamiento, y las distancias entre los vectores representan la similitud entre las categorías.

La red neuronal, durante el entrenamiento, aprende a posicionar estos vectores en el espacio de tal manera que categorías similares estén cerca y categorías diferentes estén lejos. Esto permite que la red neuronal capture la relación entre las categorías y use esta información para realizar predicciones.

Beneficios de la capa de embedding:

- Conversión de datos categóricos a numéricos: Permite que la red neuronal procese datos categóricos.
- Representación semántica: Captura relaciones entre las categorías.
- Reducción de dimensionalidad: La capa de embedding reduce la dimensionalidad del espacio categórico, simplificando el aprendizaje.
- Mejora del rendimiento: Permite a la red neuronal aprender representaciones más ricas y precisas de los datos categóricos, lo que conduce a una mejor precisión predictiva.

Capa Masking: La capa Masking se utiliza en redes neuronales cuando se hace necesario procesar secuencias de diferente longitud, como es común en problemas de procesamiento de secuencias. Esta identifica los valores de padding (ceros en este caso) en las secuencias de entrada y marca estos valores para que las capas posteriores, los ignoren. Su uso es crucial para que el modelo no interprete estos valores como información válida y no aprenda patrones incorrectos basados en ellos.

Capa GRU: Explicada en la sección de Preliminares. El módulo GRU contribuye a la tarea de recomendación al capturar las dependencias a largo plazo entre los artículos en la secuencia y ajustar dinámicamente el contenido de su memoria a través de diferentes partes de la secuencia.

Capa Dropout: La capa Dropout es una técnica de regularización utilizada en redes neuronales para reducir el sobreajuste (overfitting). En redes profundas, es común que el modelo se ajuste demasiado a los datos de entrenamiento, lo que provoca una disminución del rendimiento en datos no vistos. Dropout mitiga este problema desconectando aleatoriamente una fracción de las neuronas durante el entrenamiento, lo que obliga a la red a no depender de conjuntos específicos de características y aprender representaciones más generales.

Propósito: Prevenir el sobreajuste haciendo que la red no dependa demasiado de ciertos patrones o combinaciones específicas de pesos.

Cómo funciona: Dropout desconecta aleatoriamente una proporción de neuronas en cada paso de entrenamiento (en este caso, el 30% con Dropout(0.3)). Esto significa que, en cada paso, las neuronas que queden activas deben compensar la información ausente, lo que fortalece el modelo general al hacerlo menos sensible a patrones ruidosos o irrelevantes.

Beneficio: Permite que el modelo generalice mejor, especialmente cuando se tiene un conjunto de datos limitado o cuando hay muchas características que pueden llevar a sobreajuste.

Capa Dense: La capa Dense (o capa completamente conectada) es una capa fundamental en las redes neuronales, donde cada neurona de la capa está conectada con todas las neuronas de la capa anterior. Esta capa es responsable de aprender combinaciones no lineales de las características extraídas por capas anteriores (en este caso, de las capas GRU y Dropout) y tomar decisiones sobre la salida.

Propósito: La capa Dense actúa como un "decisor" en la red, combinando la información aprendida en capas anteriores y produciendo una salida final. En este caso, se usó una capa con activación softmax para producir una distribución de probabilidad sobre las posibles clases de productos que pueden ser recomendados.

Cómo funciona: Cada neurona en la capa Dense toma una combinación ponderada de las entradas (en el proyecto, la salida de la capa GRU y Dropout), aplica una función de activación (en este caso, softmax para convertir las salidas

en probabilidades) y genera una predicción. El modelo ajusta estos pesos durante el entrenamiento para minimizar la pérdida.

Beneficio: Las capas Dense permiten realizar una clasificación efectiva en un espacio de características de alto nivel. En el proyecto, la última capa Dense con $n\_products$ unidades y activación $softmax$ devuelve las probabilidades de cada producto, lo que permite al modelo predecir cuál es el producto más probable que un usuario compre.

Relación entre las capas Dropout y Dense: Dropout regulariza las salidas de la capa GRU, haciendo que la red sea más robusta y menos propensa al sobreajuste. Dense utiliza estas salidas regularizadas para realizar la tarea final de clasificación, determinando cuál es el siguiente producto que debe recomendarse al usuario. La combinación de ambas capas mejora el rendimiento y la capacidad de generalización del modelo. Estas dos capas son comunes en arquitecturas de redes neuronales por su capacidad de combinar regularización (Dropout) y clasificación efectiva (Dense).