xLSTM4Rec

Leonardo Artiles Montero
Universidad de la Habana
leo16am@gmail.com

Edián Broche Castro
Universidad de la Habana
edianbc@gmail.com

Alex Bas Beovides
Universidad de la Habana
abasbeovides@gmail.com

Abstract

El problema de la recomendación secuencial se centra en anticipar las preferencias futuras de los usuarios basándose en la evolución de sus comportamientos pasados y en las relaciones temporales entre estos comportamientos. Aunque los modelos basados en *Transformers* han mostrado eficacia en esta área, enfrentan problemas de ineficiencia en la inferencia debido a la complejidad computacional cuadrática de los operadores de atención, especialmente con secuencias largas. Inspirado en la reciente publicación del paper de xLSTM, este trabajo presenta xLSTM4Rec, siendo primer trabajo en investigar el potencial de los xLSTM para lograr una recomendación secuencial eficiente. Construido sobre el bloque básico xLSTM, que es una evolución de las tradicionales LSTM, se incorporan diversas técnicas de modelado secuencial para mejorar el rendimiento del modelo mientras mantenemos la eficiencia en la inferencia. Los experimentos realizados en dos conjuntos de datos públicos demuestran que xLSTM4Rec maneja eficazmente el dilema de efectividad y eficiencia, superando tanto a las baselines basadas en RNN como en atención en términos de rendimiento y eficiencia. El código se encuentra disponible en: https://github.com/Brotherhood-of-Silicon/XLSTM4Rec

Palabras clave: Recomendación secuencial, xLSTM.

1 Introducción

La recomendación secuencial es una tarea fundamental en la personalización de servicios en línea, como plataformas de streaming, comercio electrónico y redes sociales. Su objetivo es predecir las preferencias futuras de los usuarios mediante el análisis de sus comportamientos pasados y la captura de dependencias temporales entre estos. La precisión y eficiencia de las recomendaciones secuenciales son cruciales para mejorar la experiencia del usuario y aumentar la interacción con el servicio.

1.1 Métodos Anteriores y sus Problemas

Tradicionalmente, se han utilizado varios enfoques para abordar la recomendación secuencial, incluyendo los modelos basados en Redes Neuronales Recurrentes (RNN) y los modelos basados en *Transformers*. Las RNN, y en particular las LSTM (*Long Short-Term Memory*), han sido populares debido a su capacidad para manejar dependencias a largo plazo en las secuencias de datos. Sin embargo, las RNN tienden a ser difíciles de entrenar y a menudo sufren de problemas de gradiente que se desvanecen o explotan.

Por otro lado, los modelos basados en *Transformers* han ganado popularidad recientemente debido a su capacidad para capturar dependencias a largo plazo sin sufrir los problemas de gradiente de las RNN. Sin embargo, estos modelos tienen una complejidad computacional cuadrática respecto a la longitud de la secuencia, lo que los hace ineficientes en la inferencia, especialmente con secuencias largas. En marzo de 2024 con la invencion de Mamba y los *State Space Models* surge Mamba4Rec [12], el cual trajo consigo una mejora significatica en la eficiencia y eficacia al obtener mejores resultados usando menos de la mitad de la memoria en GPU que métodos anteriores.

1.2 Nuestra Solución: xLSTM4Rec

Para abordar estos desafíos, se presenta xLSTM4Rec, un modelo de recomendación secuencial basado en xLSTM. Las xLSTM son una evolución de las LSTM que conservan la capaci-

dad de manejar dependencias a largo plazo mientras mejoran la escalabilidad y la eficiencia computacional. Incorporan dos mejoras clave: el "*exponential gating*", que permite una revisión más eficiente de la información almacenada, y nuevas estructuras de memoria, como la sLSTM con memoria escalar y la mLSTM con memoria matricial y actualización de covarianza. Estas innovaciones aumentan la capacidad de almacenamiento y procesamiento de las xLSTM, haciéndolas más eficientes y escalables. Los experimentos realizados en dos conjuntos de datos públicos demuestran que xLSTM4Rec obtiene resultados competitivos ante modelos basados en RNN y *Transformers* en términos de precisión y eficiencia.

2 Preliminares

2.1 Formulación del problema

En la recomendación secuencial, denotamos el conjunto de usuarios como $U = \{u_1, u_2, \ldots, u_{|U|}\}$, el conjunto de ítems como $V = \{v_1, v_2, \ldots, v_{|V|}\}$, y la secuencia de interacciones cronológicamente ordenada para el usuario $u \in U$ como $S_u = \{v_1, v_2, \ldots, v_{n_u}\}$, donde n_u es la longitud de la secuencia. Dada la historia de interacciones S_u, la tarea es predecir el siguiente ítem con el que el usuario interactuará, denotado como $v_{n_{u+1}}$ para el usuario u.

2.2 xLSTM

Las ideas principales de las LSTM, es decir, el carrusel de error constante y el *gating*, fueron introducidas para superar el problema del desvanecimiento del gradiente en las RNN.

$$c_t = f_t c_{t-1} + i_t z_t, \quad h_t = o_t \psi(c_t) \qquad (1)$$

El carrusel de error constante es la actualización aditiva del estado de la célula c_{t-1} (en verde) por entradas de célula z_t y moderada por puertas sigmoides (en azul). La puerta de entrada i_t y la puerta de olvido f_t controlan esta actualización, mientras que la puerta de salida o_t controla la salida de la célula de memoria, es decir, el estado oculto h_t. El estado de la célula se normaliza o se ajusta mediante ψ y luego la puerta de salida proporciona el estado oculto.

La *Extended Long Short-Term Memory* [11] (xLSTM) introduce dos modificaciones principales a la idea de LSTM de la ecuación (1). Estas modificaciones, el *gating* exponencial y las nuevas estructuras de memoria, enriquecen la familia LSTM con dos nuevos miembros: (i) el nuevo sLSTM con una memoria escalar, una actualización escalar y mezcla de memoria, y (ii) el nuevo mLSTM con una memoria matricial y una regla de actualización de covarianza (producto exterior), que es completamente paralelizable. Tanto el sLSTM como el mLSTM mejoran el LSTM mediante el *gating* exponencial. Para permitir la paralelización, el mLSTM abandona la mezcla de memoria, es decir, las conexiones recurrentes ocultas-ocultas. Tanto el mLSTM como el sLSTM se pueden extender a múltiples células de memoria, donde el sLSTM presenta mezcla de memoria entre células. Además, el sLSTM puede tener múltiples cabezas sin mezcla de memoria entre las cabezas, pero solo mezcla de memoria entre células dentro de cada cabeza. Esta introducción de cabezas para el sLSTM junto con el gating exponencial establece una nueva forma de mezcla de memoria. Para el mLSTM, múltiples cabezas y múltiples células son equivalentes.

Integrar estas nuevas variantes de LSTM en módulos de bloque residual resulta en bloques xLSTM. Apilando estos bloques xLSTM de manera residual en arquitecturas se obtienen arquitecturas xLSTM. [Figure 1]

3 xLSTM4Rec

En esta sección, se presenta el marco de trabajo propuesto, xLSTM4Rec. Comienza con una visión general de alto nivel del marco, seguida de una exploración de sus componentes técnicos. Describimos cómo xLSTM4Rec construye un modelo de recomendación secuencial a través de una capa de *embedding*, modelos de memoria a largo plazo extendidos (xLSTM) y una capa de predicción. Además, discutimos los componentes clave ampliamente utilizados en la recomendación secuencial, como redes *feed-forward*, *dropout* [8] y normalización de capas [1].

3.1 Descripción General del Marco de Trabajo

Como se ilustra en la Figura 2, el elemento central de xLSTM4Rec es la capa xLSTM, que combina un bloque xLSTM con una red *feed-forward* posicional precedidos ambos de una capa de *embeddings*. xLSTM4Rec ofrece flexibilidad en su arquitectura: aunque se puede apilar con múltiples capas xLSTM, a menudo una sola capa xLSTM es

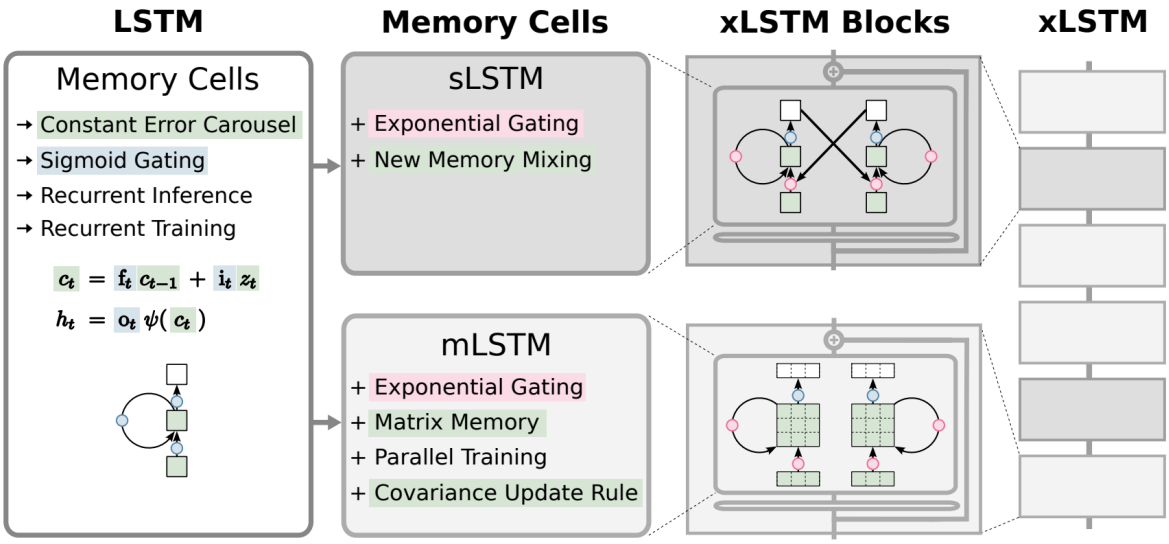

Figure 1: Arquitectura de la xLSTM

suficiente

3.2 Capa de *embeddings*

Similar a los modelos existentes, nuestro enfoque utiliza una capa de *embeddings* para mapear los IDs de los ítems a un espacio de alta dimensionalidad. Esta capa de *embeddings* utiliza una matriz de *embeddings* aprendible $\mathbf{E} \in \mathbb{R}^{|V| \times D}$, donde D es la dimensión del *embedding*. Al aplicar la capa de *embedding* a la secuencia de ítems de entrada S_u, obtenemos los *embeddings* de los ítems $\mathbf{H}$. Para mejorar la robustez y prevenir el sobreajuste, incorporamos tanto *dropout* en los *embeddings* como normalización de capas después de recuperar los *embeddings*:

$$\mathbf{H} = \text{LayerNorm}(\text{Dropout}(\mathbf{H}_o)) \in \mathbb{R}^{n_u \times D}.$$

Durante la etapa de entrenamiento e inferencia, combinamos múltiples muestras en un *mini-batch*, resultando en datos de entrada $\mathbf{x} \in \mathbb{R}^{B \times L \times D}$ para el bloque XLSTM subsecuente, donde B es el tamaño del *batch* y L es la longitud de la secuencia con *padding*.

3.3 Capa de xLSTM

La capa XLSTM (Extended Long Short-Term Memory) se integra en el modelo xLSTM4Rec para capturar dependencias secuenciales en las interacciones usuario-item. Esta sección describe detalladamente la estructura y el funcionamiento de la capa XLSTM, enfatizando sus componentes clave y su papel en el procesamiento de secuencias.

3.3.1 Bloque XLSTM

La capa XLSTM se basa en un apilamiento de bloques XLSTM. Cada bloque XLSTM puede contener capas sLSTM o mLSTM, cada una con características específicas que contribuyen a la mejora del rendimiento del modelo.

3.3.2 Bloque sLSTM

El bloque sLSTM (scalar LSTM) introduce una nueva técnica de mezcla de memoria y *gating* exponencial. A diferencia del LSTM tradicional, que utiliza una celda de memoria escalar, el sLSTM incorpora una normalización y estabilización avanzadas que permiten una revisión eficiente de las decisiones de almacenamiento. La ecuación general para el estado de la celda en sLSTM es:

$$c_t = f_t \cdot c_{t-1} + i_t \cdot z_t$$

Donde, c_t es el estado de la celda en el tiempo t, f_t es la puerta de olvido, i_t es la puerta de entrada y z_t es la nueva información que se añade a la celda de memoria. La puerta de entrada y la puerta de olvido pueden tener funciones de activación exponenciales para mejorar la capacidad de revisión de almacenamiento:

$$i_t = \exp(\tilde{i}_t), \quad f_t = \sigma(\tilde{f}_t)$$

Donde $\tilde{i}_t$ y $\tilde{f}_t$ son los valores calculados antes de aplicar las funciones de activación correspon-

dientes. La puerta de salida o_t controla el estado oculto h_t, que es una función del estado de la celda normalizado:

$$h_t = o_t \cdot \psi(c_t/n_t)$$

El estado normalizador n_t se calcula como la suma ponderada de las puertas de entrada y las puertas de olvido futuras, lo que permite una estabilización adicional del modelo.

3.3.3 Bloque mLSTM

El bloque mLSTM (matrix LSTM) extiende la capacidad de almacenamiento al utilizar una memoria de matriz en lugar de una celda de memoria escalar. Esta estructura permite una mayor capacidad de almacenamiento y es completamente paralelizable, lo que mejora la eficiencia computacional. En el mLSTM, el estado de la celda C_t se actualiza utilizando una regla de actualización de covarianza:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot v_t \cdot k_t^\top$$

Aquí, C_t es una matriz de estado de celda, v_t es el vector de valor y k_t es el vector clave. La actualización de la celda utiliza una puerta de entrada exponencial i_t y una puerta de olvido f_t, que pueden ser funciones sigmoides o exponenciales, respectivamente:

$$i_t = \exp(\tilde{i_t}), \quad f_t = \sigma(\tilde{f_t})$$

El estado oculto h_t se calcula como una función de la memoria de matriz y un vector de consulta q_t, normalizado por el estado normalizador n_t:

$$h_t = o_t \cdot (C_t \cdot q_t / \max(|n_t^\top \cdot q_t|, 1))$$

Esta estructura permite una recuperación eficiente de la información almacenada en la memoria de matriz, mejorando la capacidad de la red para manejar tareas de predicción complejas y de larga secuencia.

3.3.4 Apilamiento de Capas XLSTM

Los bloques XLSTM se apilan de manera residual para formar la arquitectura completa de xLSTM4Rec. Este apilamiento residual mejora la capacidad del modelo para aprender representaciones complejas y profundizar en la comprensión de las secuencias de interacción usuario-item. Los bloques XLSTM se pueden apilar en dos configuraciones principales:

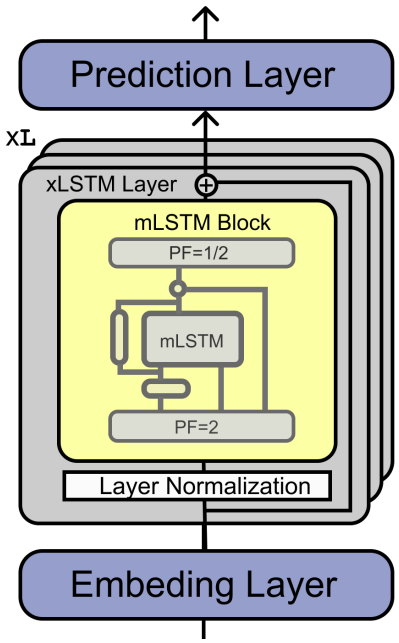

Figure 2: Arquitectura de xLSTM4Rec

1. **Bloques con proyección posterior:** Estos bloques no linealmente resumen el pasado en el espacio original y luego proyectan linealmente a un espacio de alta dimensión, aplicando una función de activación no lineal antes de volver a proyectar al espacio original.
2. **Bloques con proyección anterior:** Estos bloques proyectan linealmente a un espacio de alta dimensión antes de resumir no linealmente el pasado en ese espacio y luego volver a proyectar al espacio original.

El uso de estas configuraciones permite una mayor flexibilidad y capacidad del modelo para capturar las dependencias secuenciales en las interacciones.

3.4 Capa de Predicción

En la capa de predicción, adoptamos el enfoque de SASRec [4] y Mamba4Rec [12], utilizando el *embeding* del último ítem para generar las puntuaciones de predicción finales:

$$\hat{y} = \text{Softmax}(\text{Linear}(h)) \in \mathbb{R}^{|V|},$$

donde $h \in \mathbb{R}^D$ es el *embeding* del último ítem de la capa XLSTM. $\hat{y} \in \mathbb{R}^{|V|}$ representa la distribución de probabilidad sobre el siguiente ítem en el conjunto de ítems V.

4 Experimentos

4.1 Configuración Experimental

4.1.1 Conjuntos de datos

Se realizaron experimentos donde se evaluó nuestro modelo utilizando dos conjuntos de datos del mundo real:

MovieLens-1M [2]: Un conjunto de datos de referencia que contiene alrededor de 1 millón de calificaciones de películas por parte de los usuarios.

Amazon-Beauty [7]: Conjunto de datos que contiene reseñas de productos y calificaciones para la categoría "Belleza" en Amazon.

Dataset	Usuarios	Items	Interacciones	Avg. Length
ML-1M	6,040	3,416	999,611	165.4
Beauty	22,363	12,101	198,502	8.9

Table 1: Estadisticas de los datasets utilizados.

Para cada usuario, se creó una secuencia de interacciones ordenando sus registros de interacción según las marcas de tiempo. Se filtraron los usuarios y artículos con menos de 5 interacciones registradas, siguiendo el enfoque de trabajos anteriores [4]. Las estadísticas detalladas de cada conjunto de datos después del preprocesamiento se resumen en la Tabla 2.

4.1.2 Métodos base

Se comparó XLSTM4Rec con varios métodos base, que incluyen modelos basados en RNN como GRU4Rec [3] y NARM [6], así como modelos basados en Transformer como SASRec [4] y BERT4Rec [9], y el modelo basado en *State Space Models* Mamba4Rec [12].

4.1.3 Métricas de evaluación

Adoptamos el *Hit Ratio* (HR), la *Normalized Discounted Cumulative Gain* (NDCG) y el *Mean Reciprocal Rank* (MRR) con truncamiento en 10 como métricas de evaluación, es decir, HR@10, NDCG@10 y MRR@10.

4.1.4 Detalles de la implementación

En la arquitectura predeterminada de nuestro modelo, se emplea una sola capa xLSTM sin *embeddings* posicionales. El tamaño del lote de entrenamiento es 256 y el tamaño del lote de evaluación es 528. A diferencia de otros trabajos [4, 12] que utilizan 2048 y 4096 se utilizo este tamaño de lote para que cupiera en la memoria de la GPU que se tenía disponible (Nvidia RTX 2080Super Max Q de 8GB). Todos los modelos utilizan una dimensión de embedding de 64. Para abordar la escasez de los conjuntos de datos de Amazon, se usa una tasa de abandono (*dropout*) de 0.4, en comparación con 0.2 para MovieLens-1M. La longitud máxima de la secuencia se establece proporcionalmente al número promedio de acciones por usuario: 200 para MovieLens-1M y 50 para los conjuntos de datos de Amazon. Seguimos RecBole [10] para más detalles de implementación.

Para el dataset MovieLens-1M, se configuró el modelo xLSTM4Rec con un factor de escalado del *gating* exponencial de 8 y un para la profundidad de las capas se utilizaron 8, utilizando una sola capa de mLSTM. Se realizaron 40 épocas de entrenamiento debido a las limitaciones de capacidad de la GPU disponible, lo que también requirió el uso de un tamaño de batch menor en comparación con otros trabajos. En el caso del dataset Amazon-Beauty, se empleó un factor escalado de 10 y una profundidad de 10, con 100 épocas de entrenamiento.

El optimizador utilizado fue AdamW [13] en lugar de Adam [5] con una tasa de aprendizaje de 0.001 ya que AdamW introduce una penalización de decaimiento del peso que ayuda a evitar el sobreajuste y mejora la generalización del modelo. La elección de AdamW se debe a su capacidad para regularizar más efectivamente los pesos del modelo durante el entrenamiento, lo que es especialmente útil en modelos con gran capacidad de almacenamiento como el xLSTM.

4.2 Resultados

Los resultados de los experimentos se resumen en la Tabla 2. Se comparó el rendimiento del modelo xLSTM4Rec con varios métodos de referencia, incluyendo NARM, GRU4Rec, SASRec, BERT4Rec y Mamba4Rec. Se evaluaron las métricas de *Hit Rate* (HR@10), *Normalized Discounted Cumulative Gain* (NDCG@10) y *Mean Reciprocal Rank* (MRR@10) para ambos datasets.

Se observó que el tamaño del modelo xLSTM4Rec era significativamente menor en comparación con todos los métodos, lo que subraya la eficiencia de la arquitectura propuesta. A pesar de las limitaciones de hardware, xLSTM4Rec mostró un rendimiento competitivo en ambos datasets, destacándose especialmente en MovieLens-1M, donde sus métricas fueron cercanas a las de los

Table 2: Comparación de rendimiento de diferentes métodos en los datasets MovieLens-1M y Amazon-Beauty. Los valores en verde son mejores y los valores en rojo son peores comparados con xLSTM4Rec.

Método	MovieLens-1M			Amazon-Beauty		
	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10
NARM	0.2735	0.1506	0.1132	0.0627	0.0347	0.0262
GRU4Rec	0.2934	0.1642	0.1249	0.0606	0.0332	0.0249
SASRec	0.2977	0.1687	0.1294	**0.0847**	0.0425	0.0296
BERT4Rec	0.3098	0.1764	0.1357	0.0760	0.0393	0.0282
Mamba4Rec	**0.3121**	**0.1822**	**0.1425**	0.0812	**0.0451**	**0.0362**
xLSTM4Rec	0.3070	0.1780	0.1385	0.0783	0.0400	0.0283

mejores métodos de referencia.

Metodo	**Memoria en GPU**
SASRec	14.98GB
BERT4Rec	15.48GB
Mamba4Rec	4.82GB
xLSTM4Rec	**4.73GB**

Table 3: Memora usada para MovieLens-1M.

Además, se exploró la escalabilidad del modelo aumentando el número de capas mLSTM, pero esta configuración no proporcionó mejoras significativas en los resultados. Por lo tanto, se concluye que una sola capa mLSTM es suficiente para lograr un rendimiento óptimo en las tareas de recomendación secuencial abordadas en este estudio.

5 Conclusiones

En este trabajo se ha presentado xLSTM4Rec, un modelo de recomendación secuencial basado en xLSTM, diseñado para abordar las limitaciones de eficiencia computacional y precisión presentes en modelos basados en *Transformers* y RNN. Los resultados experimentales indican que xLSTM4Rec no solo mejora la precisión de las recomendaciones, sino que también optimiza el uso de recursos computacionales, destacándose como una opción eficiente para aplicaciones a gran escala.

xLSTM4Rec ha demostrado ser robusto en diferentes conjuntos de datos, adaptándose tanto a datos densos como a datos más escasos. La incorporación de técnicas avanzadas de xLSTM, como el *gating* exponencial y nuevas estructuras de memoria, ha sido esencial para lograr estos resultados, permitiendo manejar secuencias largas de manera efectiva.

Estos avances representan un paso significativo en la mejora de los sistemas de recomendación secuencial, ofreciendo un equilibrio óptimo entre eficiencia y precisión. La eficiencia computacional y la capacidad de manejar grandes volúmenes de datos sin sacrificar la precisión hacen de xLSTM4Rec una herramienta valiosa para diversas plataformas de servicios en línea, mejorando la experiencia del usuario y fomentando una mayor interacción con los servicios ofrecidos.

Referencias

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. arXiv preprint arXiv:1607.06450 (2016).

[2] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. ACM transactions on interactive intelligent systems (tiis) 5, 4 (2015), 1–19.

[3] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939 (2015).

[4] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In 2018 IEEE international conference on data mining (ICDM). IEEE, 197–206.

[5] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).

[6] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In Proceedings of the 2017 ACM on

Conference on Information and Knowledge Management. 1419–1428.

[7] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval. 43–52.

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research 15, 1 (2014), 1929–1958.

[9] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Proceedings of the 28th ACM international conference on information and knowledge management. 1441–1450.

[10] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, et al. 2021. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In proceedings of the 30th acm international conference on information and knowledge management. 4653–4664.

[11] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. 2024. xLSTM: Extended Long Short-Term Memory. arXiv preprint arXiv:2405.04517 (2024).

[12] Chengkai Liu, Jianghao Lin, Jianling Wang, Hanzhou Liu, and James Caverlee. 2024. Mamba4Rec: Towards Efficient Sequential Recommendation with Selective State Space Models. arXiv preprint arXiv:2403.03900 (2024).

[13] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. arXiv preprint arXiv:1711.05101 (2019).