# Overview

## Essential Data

* Story Data:
  + Text of the stories, or rich metadata like:
    - Genre(s)
    - Themes
    - Plot summaries
    - Keywords
    - Level
    - Author information
    - Voice type
    - Background style
    - Number of reads per day (có thể bỏ)
* User Data:
  + Properties of the user:
    - Age
    - Country
    - Area
    - Level
    - Avg reading time per story
  + User's action:
    - List of read stories và số lần đọc mỗi lần, mức độ hoàn thành

## Position for Recommend

* Popup (When ending a story)
* Add a new "Recommendations" section to the Library
* No change at the Library

## Goals

* New user: Retention rate
* Purchased: increase learning time

## Workflow

* [Refer documents](https://drive.google.com/file/d/1mBa7mtVj-CiHwRWPOEyIudd55_WYz05n/view)

## Milestone

* [Refer documents](https://docs.google.com/spreadsheets/d/1zhsfFRCgPvcU5wezdUOAIOHiLxb6he4uIO5sR-faJd8/edit?usp=sharing)

# Cơ sở lý thuyết và thuật toán

## Define caring rate (target)

Mục tiêu của RS sẽ cung cấp được “whether user like story or not”. Vì vậy cần định nghĩa một tham số gọi là: Caring Rate (CR)

1. Define the Metrics:
   * Reading Times (R): The number of times a user has read or accessed the story.
   * Completion Rate (C): The percentage of the story that the user completes on average per read. This is a value between 0 and 1, where 1 indicates 100% completion.
2. Assumptions:
   * Higher reading times indicate higher interest, as the user returns to the story multiple times.
   * A higher completion rate suggests the user is engaged enough to read through most or all of the story.
3. Formula Construction:
   * We want a formula that increases as either the number of readings or the completion rate increases.
   * However, the completion rate should potentially have a more substantial weight because it directly measures engagement depth.
4. Proposed Formula:

* Công thức 1

(1)

* Where *α* and *β* are weights that determine the influence of reading times and completion rate, respectively.
* Choosing *α* and *β* depends on how much importance you want to give to each metric. A common starting point could be *α*=0.5 and *β*=1.5, emphasizing completion rate more.
* Công thức 2

(2)

* Công thức 3

(3)

| CT | Ưu điểm | Nhược điểm |
| --- | --- | --- |
| 1 | Đơn giản  Dễ dàng điều chỉnh được tác nhân ảnh hưởng đến mức độ quan tâm của user | Khó tìm được tham số tối ưu  Yêu cầu R, C một cách chính xác |
| 2 | Đơn giản  Hiệu quả thể hiện mối tương quan  Không cần thêm tham số | Dễ bị bias với 1 giá trị R, C bị nhiễu và noise. Vì vậy yêu câu R, C được đo đạc một cách chính xác  Scale tỉ lệ điểm rất lớn do phép nhân nên chỉ sự chênh lệch nhỏ, dẫn đến rate rất khác biệt |
| 3 | Đủ đơn giản  Loại bỏ được ảnh hưởng của nhiễu của R,C bằng phép khử lograrit | Range tỉ lệ có thể không có quá nhiều khác biệt giữa các user |

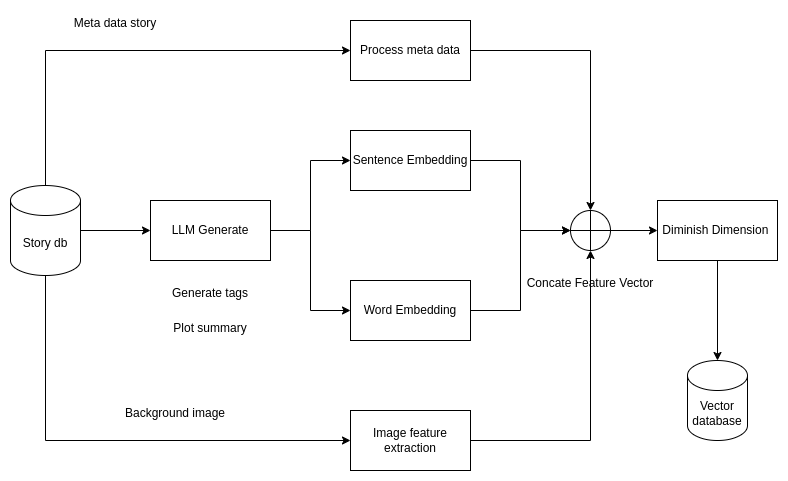
## Process data

2.1 Process story data

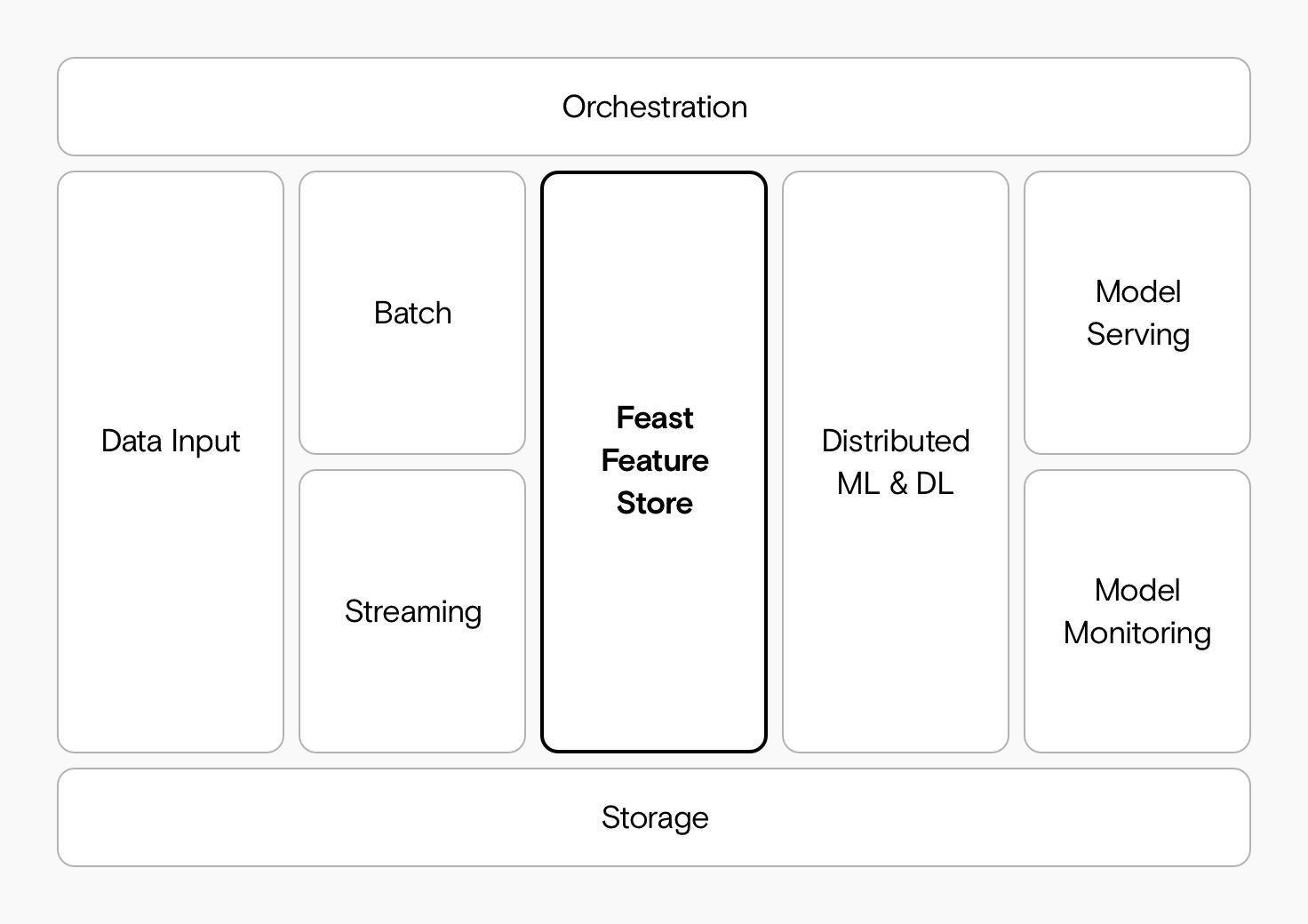
* Phân tích data và pp xử lý

| Attr | Đặc điểm | PP xử lý |
| --- | --- | --- |
| Genres | Thuộc loại data: Category | Use one-hot encoding to transform |
| Themes | Thuộc loại data: Category | Use one-hot encoding to transform |
| Plot summaries | Thuộc kiểu dữ liệu text | Sentence Embedding |
| Keywords | Thuộc kiểu dữ liệu: Text | Word2Vec TF-IDF |
| Level | Thuộc loại data: Category | Use one-hot encoding to transform |
| Author | Thuộc loại data: Category | Use one-hot encoding to transform |
| Voice | Thuộc loại data: Category | Use one-hot encoding to transform |
| Background Style | Thuộc loại data: Category | Use one-hot encoding to transform |
| Background | Thuộc kiểu dữ liệu: Image | Use CNN embedding: RestNet, VGG Use AE to diminish dimensions |

* Luồng process data



\* Vòng đời sử dụng của data



2.2 Process user data

* Phân tích data và pp xử lý

| Attr | Đặc điểm | PP xử lý |
| --- | --- | --- |
| Age | Thuộc loại data: Numerical |  |
| Country | Thuộc loại data: Category | Use one-hot encoding to transform |
| Area | Thuộc loại data: Category | Sentence Embedding |
| Level | Thuộc loại data: Category | Use one-hot encoding to transform |
| Reading time per day | Thuộc loại data: Numerical |  |

2.3 Xây dựng Utility Matrix

**Utility Matrix** là một bảng hai chiều được sử dụng trong các hệ thống **gợi ý** để biểu diễn mức độ yêu thích của người dùng đối với các sản phẩm (item). Mỗi ô trong ma trận tương ứng với một cặp người dùng (user) và sản phẩm (item), chứa giá trị thể hiện mức độ yêu thích của người dùng đó đối với sản phẩm đó.

| Người dùng | Story 1 | Story 2 | Story 3 | ... | Story 10 |
| --- | --- | --- | --- | --- | --- |
| User 1 | 5 | 3 | 4 | ... | 1 |
| User 2 | 4 | 2 | 5 | ... | 3 |
| User 3 | 1 | 2 | 3 | ... | 5 |
| ... | ... | ... | ... | ... | ... |
| User 5 | 3 | 5 | 2 | ... | 4 |

Trong ma trận này, giá trị 5 trong ô giao giữa User 1 và Story 1 cho biết User 1 rất thích Story 1, trong khi giá trị 1 trong ô giao giữa User 3 và Story 10 cho biết User 3 không thích Story 10. -> Giá trị ngày được tính toán từ các công thức CR

## Các phương pháp trong recommend System

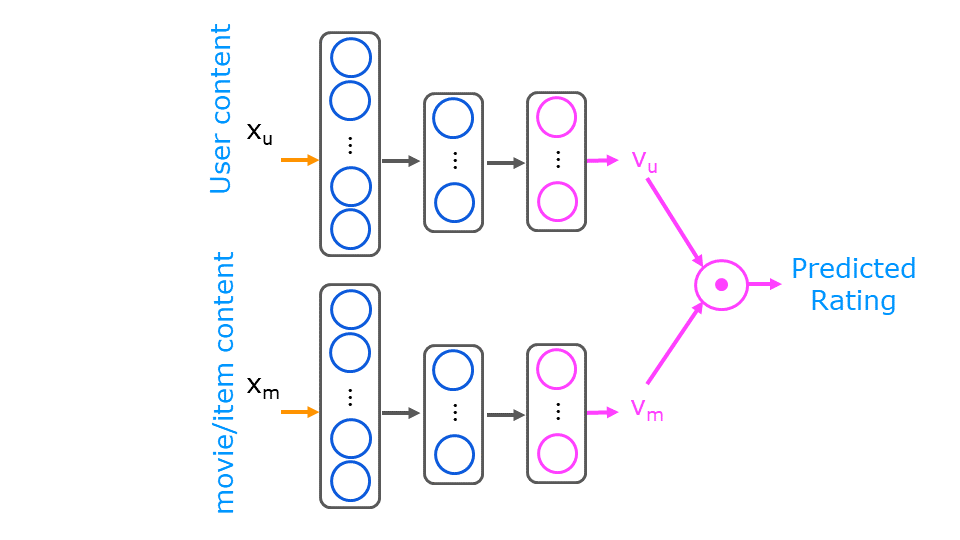
* Content filtering

**Content-based filtering** (Lọc theo nội dung) là một kỹ thuật **gợi ý** được sử dụng trong các hệ thống **đề xuất** để dự đoán mức độ yêu thích của người dùng đối với các sản phẩm (item) dựa trên các thuộc tính của sản phẩm đó và sở thích của người dùng. Nói cách khác, phương pháp này sẽ phân tích các đặc điểm của sản phẩm, chẳng hạn như chủ đề, thể loại, phong cách, v.v., và so sánh chúng với sở thích của người dùng được thể hiện qua lịch sử tương tác của họ (ví dụ: sản phẩm đã mua, phim đã xem, bài hát đã nghe, v.v.) để đưa ra những gợi ý phù hợp.

Method 1: None ML Approach phương pháp này sẽ sử dụng các feature vector để tính toán độ tương đồng giữa các item bằng các measure Cosin, Euclu

Method 2: ML -> sử dụng ma trận utility matrix kết hợp các thuật toán ML như SVM, LR, SVD cho các bài toán Linear Regression or bài toán Classification

Method 3: Deep Learning Appoarch:

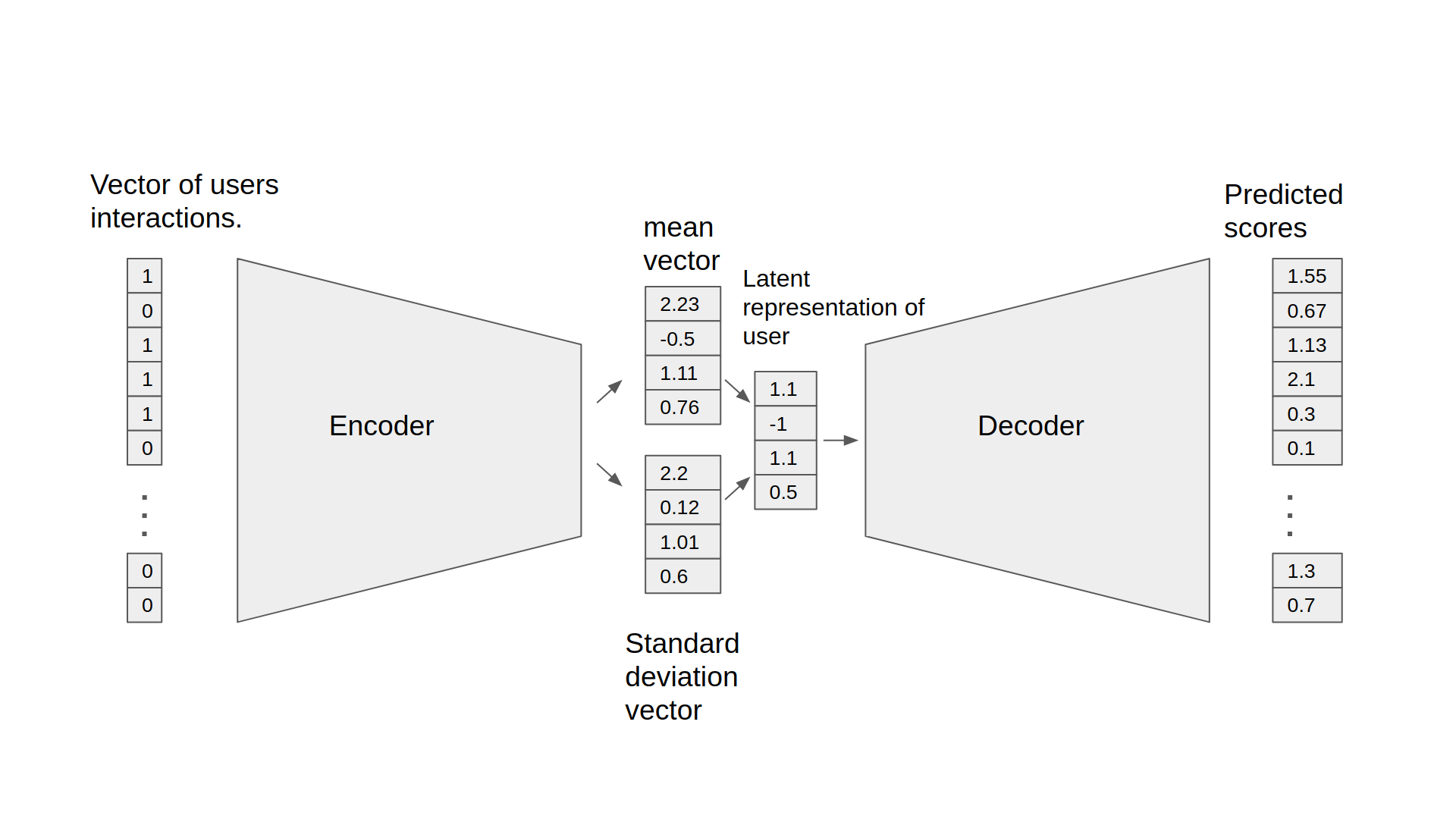


* Collaborative filtering

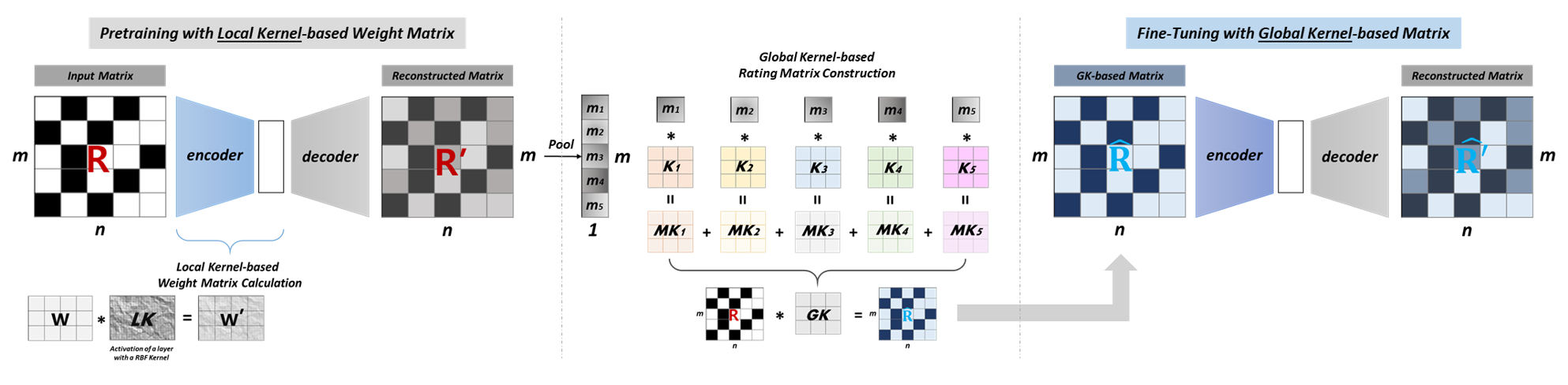
An autoencoder neural network reconstructs the input layer at the output layer by using the representation obtained in the hidden layer. An autoencoder for collaborative filtering learns a non-linear representation of a user-item matrix and reconstructs it by determining missing values.

The [NVIDIA GPU-accelerated](https://github.com/NVIDIA/DeepLearningExamples) Variational Autoencoder for Collaborative Filtering ([VAE-CF](https://github.com/NVIDIA/DeepLearningExamples/tree/master/TensorFlow/Recommendation/VAE-CF)) is an optimized implementation of the architecture first described in [Variational Autoencoders for Collaborative Filtering](https://arxiv.org/abs/1802.05814). VAE-CF is a neural network that provides collaborative filtering based on user and item interactions. The training data for this model consists of pairs of user-item IDs for each interaction between a user and an item.

The model consists of two parts: the [encoder and the decoder.](https://developer.nvidia.com/blog/deep-learning-nutshell-sequence-learning/) The encoder is a feedforward, fully connected neural network that transforms the input vector, containing the interactions for a specific user, into an n-dimensional variational distribution. This variational distribution is used to obtain a latent feature representation of a user (or embedding). This latent representation is then fed into the decoder, which is also a feedforward network with a similar structure to the encoder. The result is a vector of item interaction probabilities for a particular user.

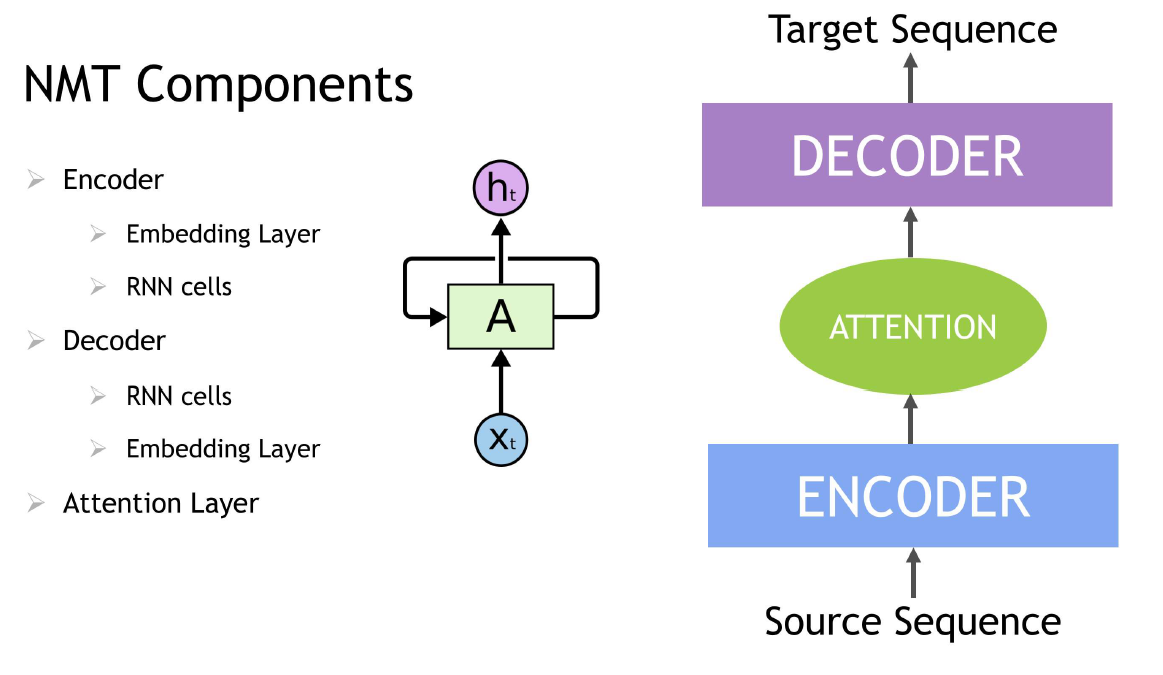


The state of the art model base on AE for collabrative filtering: Global K recommend System



* Contextual Sequence Learning

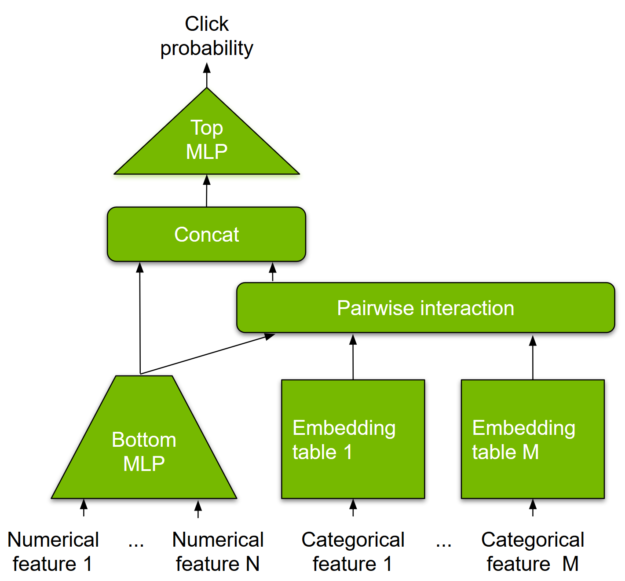
A [Recurrent neural network](https://developer.nvidia.com/discover/recurrent-neural-network) (RNN) is a class of [neural network](https://developer.nvidia.com/discover/artificialneuralnetwork) that has memory or feedback loops that allow it to better recognize patterns in data. RNNs solve difficult tasks that deal with context and sequences, such as natural language processing and are also used for contextual sequence recommendations. What distinguishes sequence learning from other tasks is the need to use models with active data memory, such as [LSTMs](https://developer.nvidia.com/discover/lstm) (Long Short-Term Memory) or [GRU](https://developer.nvidia.com/discover/recurrent-neural-network) (Gated Recurrent Units) to learn temporal dependence in input data. This memory of past input is crucial for successful sequence learning. Transformer deep learning models, such as BERT (Bidirectional Encoder Representations from Transformers), are an alternative to RNNs that apply an attention technique—parsing a sentence by focusing attention on the most relevant words that come before and after it. Transformer-based deep learning models don’t require sequential data to be processed in order, allowing for much more parallelization and reduced training time on GPUs than RNNs.



* DLRM

[DLRM](https://developer.nvidia.com/blog/optimizing-dlrm-on-nvidia-gpus/) is a DL-based model for recommendations introduced by Facebook [research](https://arxiv.org/pdf/1906.00091.pdf). It’s designed to make use of both categorical and numerical inputs that are usually present in recommender system training data. To handle categorical data, embedding layers map each category to a dense representation before being fed into multilayer perceptrons (MLP). Numerical features can be fed directly into an MLP.

At the next level, second-order interactions of different features are computed explicitly by taking the dot product between all pairs of embedding vectors and processed dense features. Those pairwise interactions are fed into a top-level MLP to compute the likelihood of interaction between a user and an item pair.



Compared to other DL-based approaches to recommendation, DLRM differs in two ways. First, it computes the feature interaction explicitly while limiting the order of interaction to pairwise interactions. Second, DLRM treats each embedded feature vector (corresponding to categorical features) as a single unit, whereas other methods (such as Deep and Cross) treat each element in the feature vector as a new unit that should yield different cross terms. These design choices help reduce computational/memory costs while maintaining competitive accuracy.

DLRM forms part of NVIDIA [Merlin](https://developer.nvidia.com/nvidia-merlin), a framework for building high-performance, DL-based recommender systems, which we discuss below.

## **System Design**

* **Overview Architecture**



Reference Suggestion from Nvidia: <https://docs.nvidia.com/deeplearning/performance/recsys-best-practices/index.html>

* **Tech Stack**

| STT | TechStack | Purpose | Lý do |
| --- | --- | --- | --- |
| 1 | Raw Data | Warehouse | Warehouse của Monkey |
| 2 | Essential Data | Clickhouse | Các data từ ware sẽ được select clean và save vào clickhouse phục vụ cho real-time query |
| 2 | Extract & Transform: | NV Tabular  Pandas | Sử dụng NV tabular từ Nvidia để process data với GPU -> boost tối đa hiệu năng Pandas cho parse ban đầu của data |
| 3 | Feature store | FEAST | Feature store đơn giải và hiệu quả |
| 4 | Model Repository | S3 | Chi phí rẻ |
| 5 | Model AI framework | Huge CTR | Framework nằm trong hệ sinh thái đc Nvidia recommend cho RS |
| 6 | Serving Inference | Triton Serving |  |