NVIDIA Merlin

Miroslav Hrabal

Motivation

- Training time often matters
- Scaling 1s-100s TBs of training data
- GPUs
- End-to-end solution with both high level APIs and low level APIs

NVIDIA Merlin - Overview

- Merlin NVTabular
- Merlin Models
- Merlin HugeCTR
- Merlin Transformers4Rec
- Merlin Systems
- Merlin Distributed Training

NVTabular

- Data preprocessing/feature engineering on GPU
- Fast tabular data transformation and loading
- Mostly meant to be run on GPU (uses cudf)
- NVIDIA claim: 10x-100x faster pipeline
- Some CPU support
- But: “Initializing an NVTabular Dataset in CPU mode.This is an experimental feature with extremely limited support!”

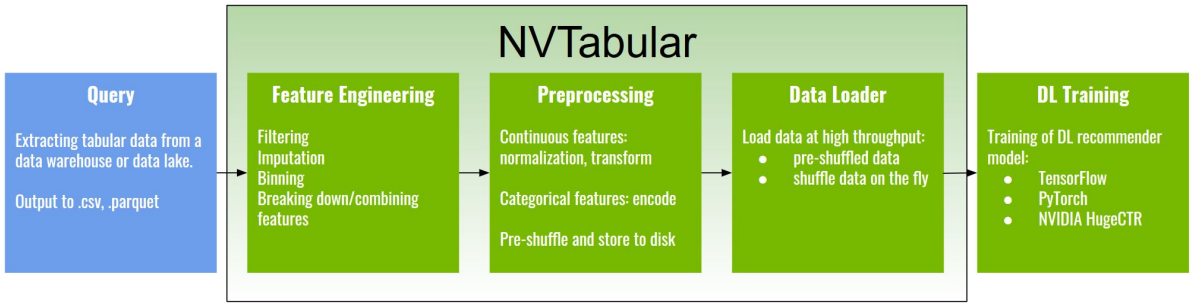


Image source: https://github.com/NVIDIA-Merlin/Merlin/blob/main/examples/getting-started-movielens/02-ETL-with-NVTabular.ipynb

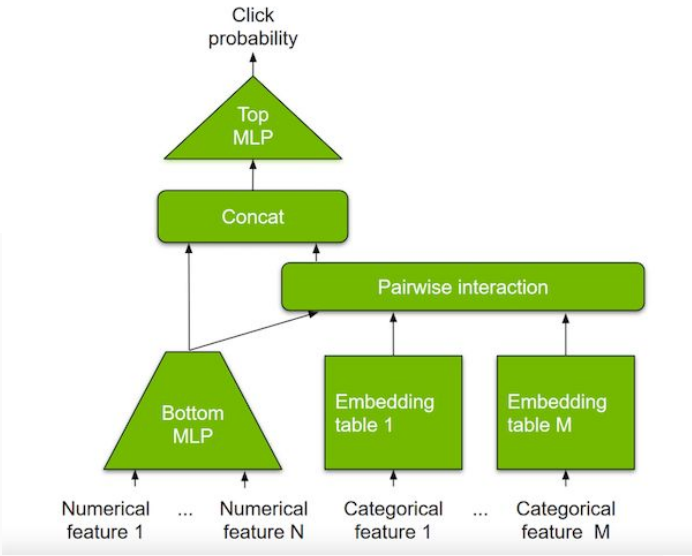
NVTabular example

```
movies = nvt.Dataset("ml-25m/movies.csv")
train = nvt.Dataset("ml25_train.csv", part_size="10M")

features = (
    ["userId", "movieId"]
    >> nvt.ops.JoinExternal(
        movies,
        on=["movieId"],
        on_ext=["movieId"],
        columns_ext=["movieId", "genres"],
    )
    >> nvt.ops.Categorify()
)
features += ["rating"] >> nvt.ops.AddMetadata(tags=[Tags.REGRESSION, Tags.TARGET])
workflow = nvt.Workflow(features)

train_transformed = workflow.fit_transform(train)
```

Merlin Models

- High level interface for training RS models
- Built on TensorFlow (and PyTorch work-in-progress)
- Retrieval models:
  - Matrix Factorization
  - Youtube DNN
  - Two Tower
- Ranking models
  - DLRM
  - DCN-v2
- (Multi-task learning models)
- Also supports creating custom model architectures
- Support for traditional models implemented by different libraries:
  - XGBoost, implicit, lightFM

Merlin Models - API example

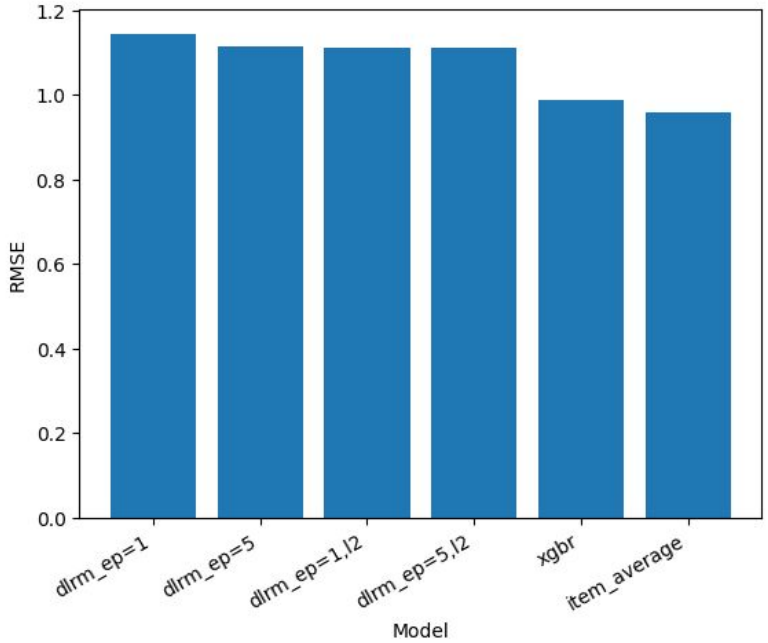


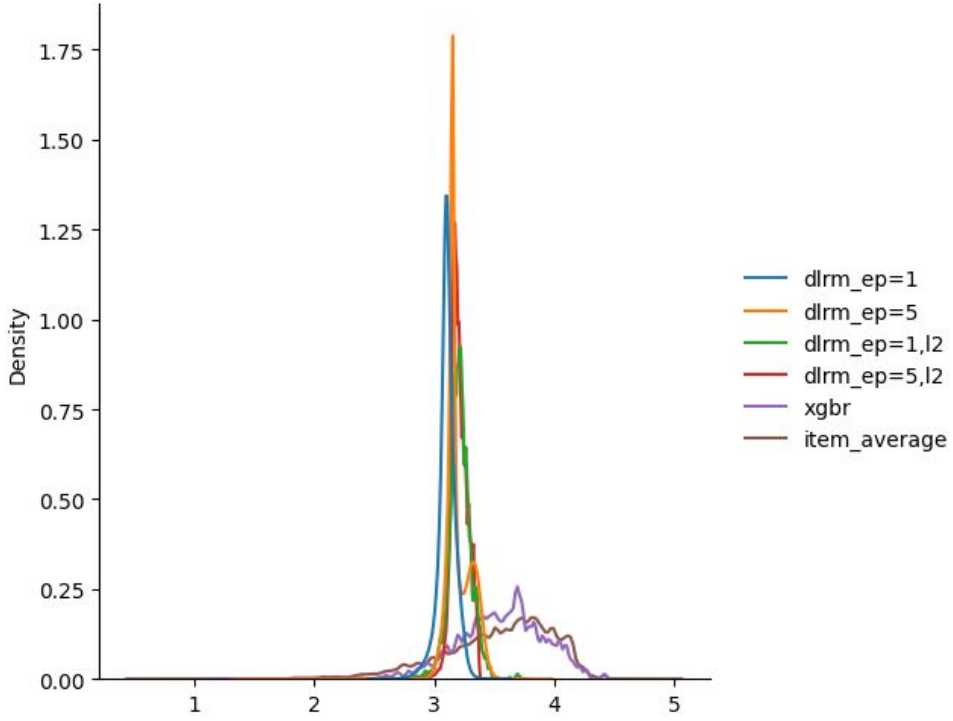
Img: https://nvidia-merlin.github.io/models/v0.10.0/models_overview.html

```
model = mm.DLRMModel(
        schema=train_transformed.schema,
        embedding_dim=64,
        top_block=mm.MLPBlock([128, 64, 32]),
        bottom_block=mm.MLPBlock([128, 64]),
        prediction_tasks=mm.RegressionTask("rating"),
    )
model.compile(optimizer="adam")
model.fit(train, validation_data=test, batch_size=1024, epochs=1)
```

Experiment

- Ranking models: DLRM (Merlin Models), scikit-learn+XGBoost on CPU and naive item average rating
- MovieLens 25M





Other libraries/frameworks

- Merlin Systems
- HugeCTR (click-through rate)
  - Another framework (alternative to Tensorflow, Pytorch)
  - API quite similar to Keras
  - Very large embedding tables, multi-node training, …
  - Used by Tencent
  - Learn how Tencent Deployed an Advertising System on the Merlin GPU Recommender Framework (2021) https://www.nvidia.com/en-us/on-demand/session/gtcspring21-s31820/
  - https://nvidia-merlin.github.io/HugeCTR/master/hugectr_talks_blogs.html
  - https://nvidia-merlin.github.io/HugeCTR/master/performance.html
- Merlin Transformers4Rec
- NVIDIA Triton Inference Server
  - Serving models/pipelines in production
  - Can load a model or ensemble exported by Merlin Models or Merlin Systems

Getting GPU Support

- CUDA + cuDNN
- cudf hard to install with pip
- Anaconda/Miniconda
- Docker containers offered by NVIDIA
  - merlin-tensorflow
  - merlin-pytorch
  - merlin-hugectr
- import cudf; cudf.set_allocator("managed")
- Dataset part_size

Additional resources

- https://nvidia-merlin.github.io/HugeCTR/master/hugectr_talks_blogs.html
- https://developer.nvidia.com/nvidia-merlin
- Example notebooks and documentation
- https://www.nvidia.com/en-us/on-demand/