Incremental Training, Session-based Recommendation, and System Level Approaches to Online Recommender Systems

Even Oldridge, Aug 6th 2023

Sequential and Session-based Recommendation

Recommendation Systems Personalize the Internet

DIGITAL CONTENT
2.7 Billion
Monthly Active Users

E-COMMERCE
2 Billion
Digital Shoppers

SOCIAL MEDIA
3.8 Billion
Active Users

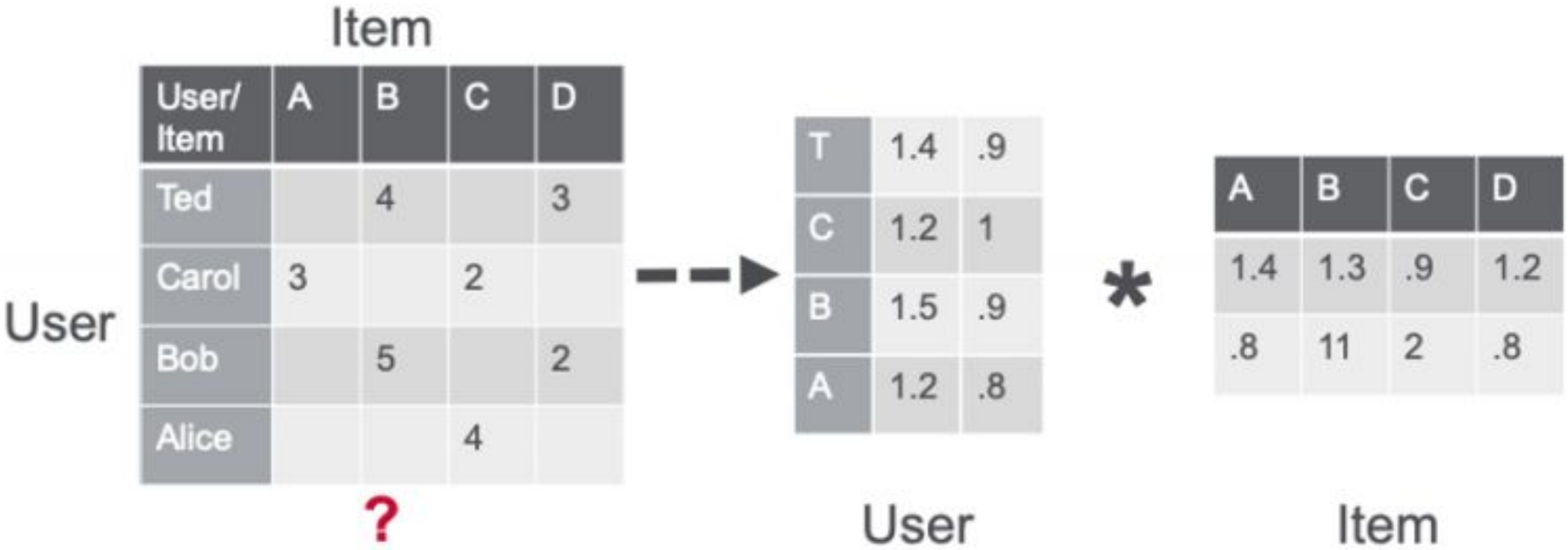
DIGITAL ADVERTISING
4.65 Billion
Users Targeted

"Already, 35 percent of what consumers purchase on Amazon and 75 percent of what they watch on Netflix come from product recommendations based on such algorithms."

Source: McKinsey

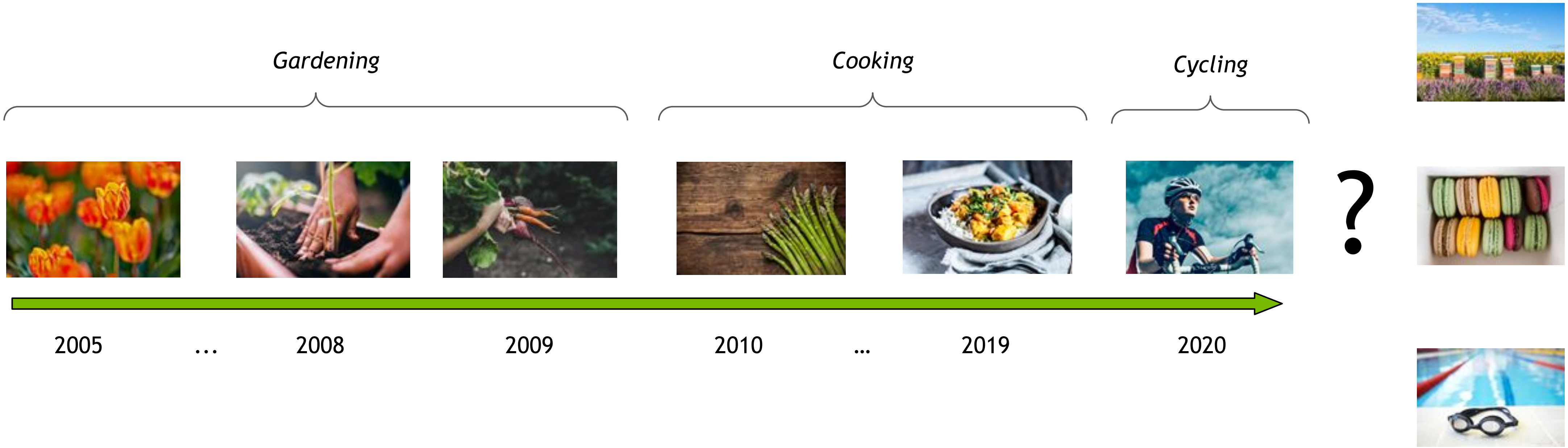
Many recommendation algorithms assume static users preference

Matrix completion



From *How to Build a Winning Recommendation System* - NVIDIA Developer blog

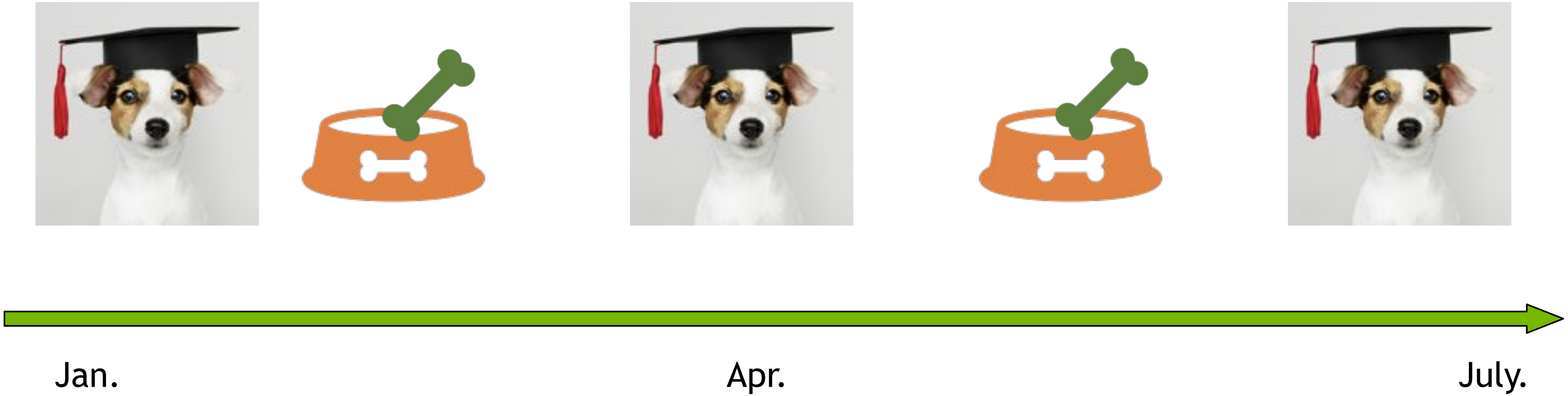
But users preference change over time...



Sequential patterns matter!

Sequential Recommendation task

Reminding / Repeated purchases



“The typical online stores gets 43% of revenue from repeat purchases”.

Session-based Recommendation task

Users behaviour for different sessions might be very distinct

Session #1 - Looking for TVs

15 days later...

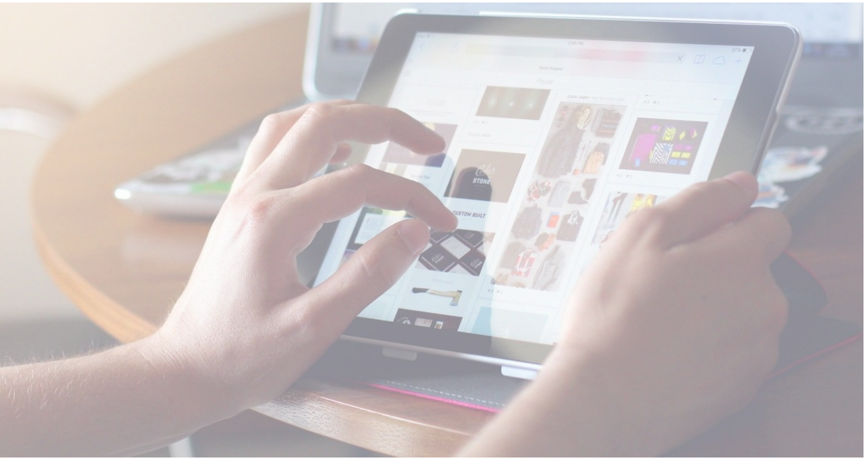
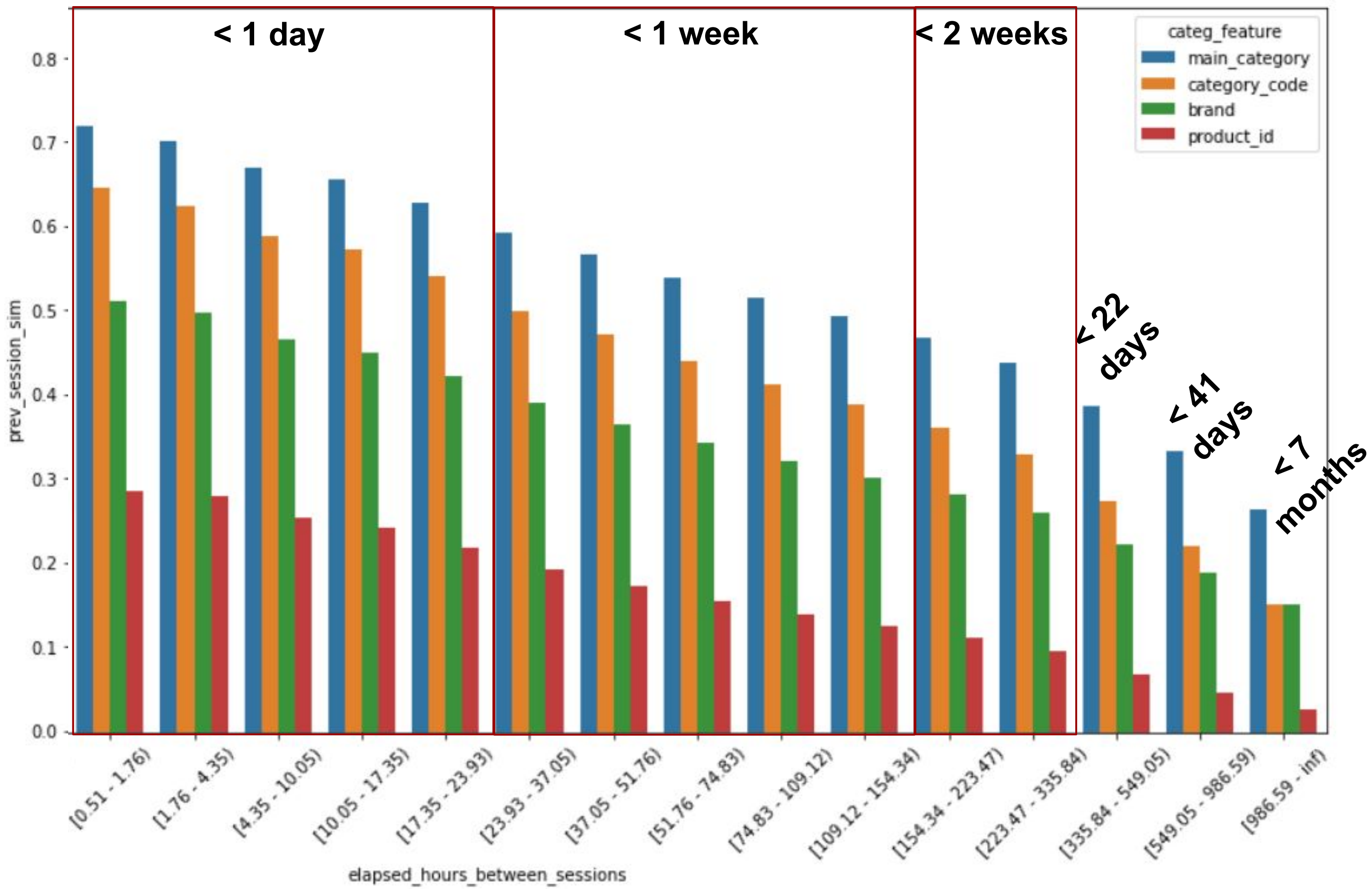


Photo by Taras Shypka on Unsplash

Session #2 - Browsing smartphones

Session-based Recommendation task

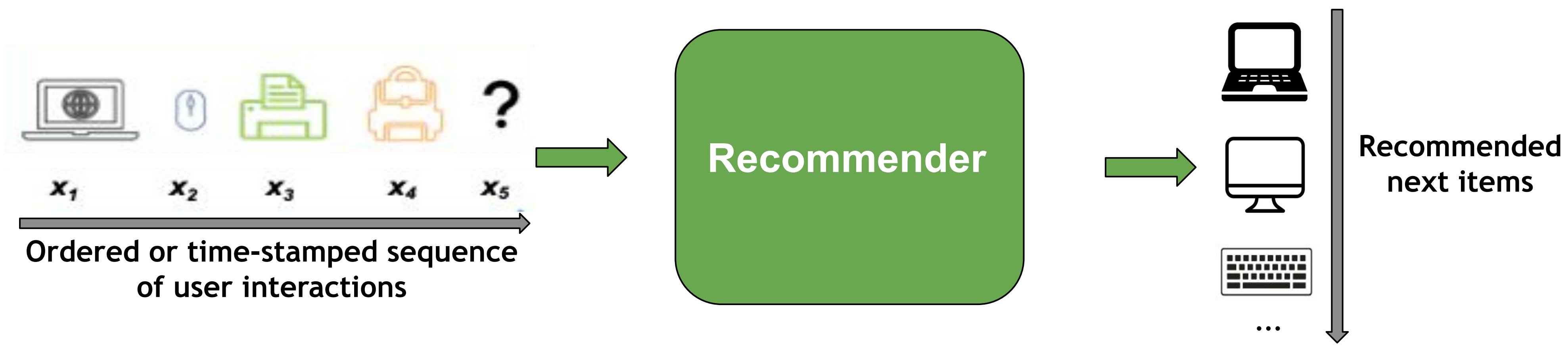
Similarity between two consecutive user sessions from an eCommerce dataset



Session-based Recommendation task

Users may browse anonymously in many online services

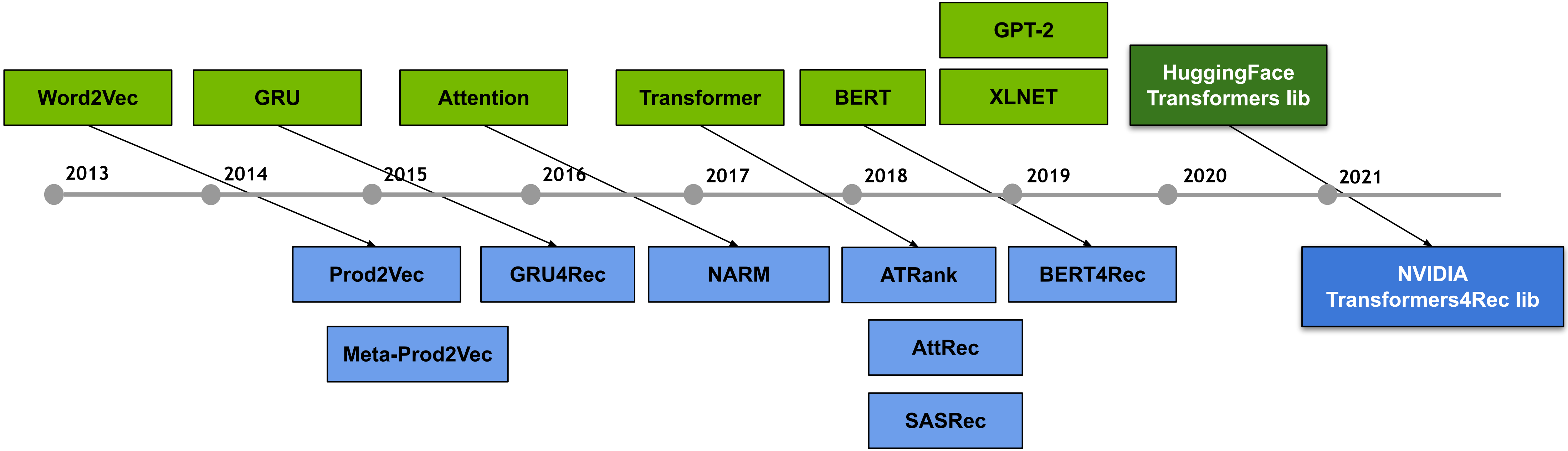
Sequential and session-based recommendation



- **Sequential recommendation** - Leverages user past interactions (usually long sequences)
- **Session-based recommendation** - Leverages only user interactions within current session (usually short sequences)

NLP x Sequential RecSys

Seminal Neural Architectures



Incremental Training

Incremental Training

Does training more frequently improve model performance?

Incremental Training

Does training more frequently improve model performance?

Features	Training Interval	like		watch	
		AUC	% improv.	AUC	% improv.
Basic ids	Daily	0.9109	-	0.7161	-
	12 hour	0.9118	+0.10%	0.7203	+0.60%
	6h	0.9126	+0.19%	0.7235	+1.03%
	1h	0.9157	+0.53%	0.7302	+1.96%
	30min	0.9174	+0.72%	0.7338	+2.48%
	5min	0.9237	+1.41%	0.7457	+4.14%
All features (Basic ids + User/item + TE)	Daily	0.9155	-	0.7219	-
	12h	0.9165	+0.11%	0.7261	+0.59%
	6h	0.9178	+0.25%	0.7304	+1.18%
	1h	0.9202	+0.51%	0.7374	+2.15%
	30min	0.9218	+0.69%	0.7404	+2.57%
	5min	0.9265	+1.20%	0.7507	+4.00%

Does this generalize?

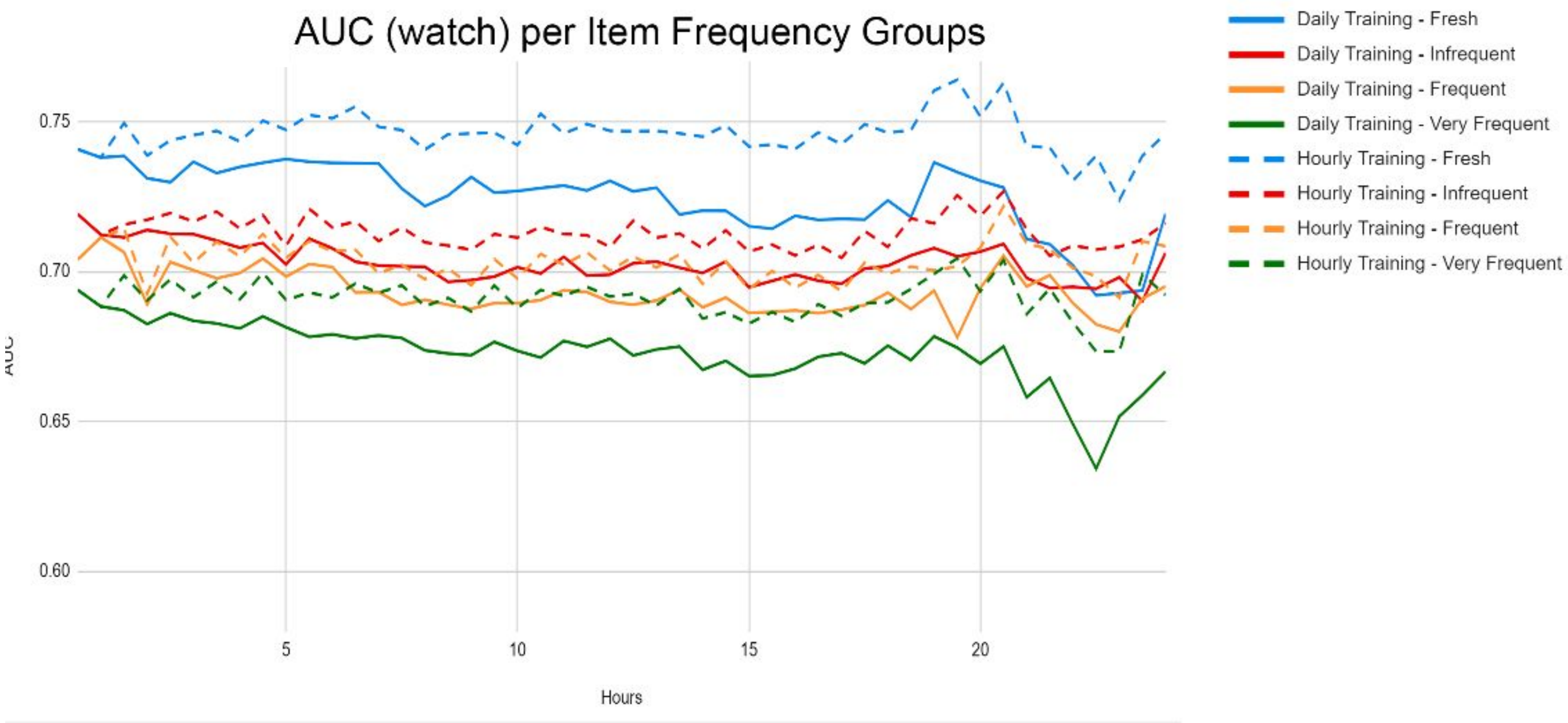
Across both feature representations and across model type we consistently see an improvement

like AUC

Model	Features	Sampled dataset Daily	Sampled dataset 30min	Sampled dataset %	Full dataset Daily	Full dataset 30min	Full dataset %
MLP	Basic ids	0.9078	0.9142	+0.69%	0.9109	0.9174	+0.72%
	Basic ids + User/item	0.9098	0.9157	+0.65%	0.9103	0.9179	+0.83%
	User/item + TE features	0.9072	0.9084	+0.13%	-	-	-
	All features	0.9130	0.9184	+0.60%	0.9155	0.9218	+0.69%
DLRM	All features	0.9103	0.9181	+0.86%	0.9152	0.9215	+0.68%
DCN-v2	All features	0.9131	0.9194	+0.68%	0.9160	0.9227	0.73%

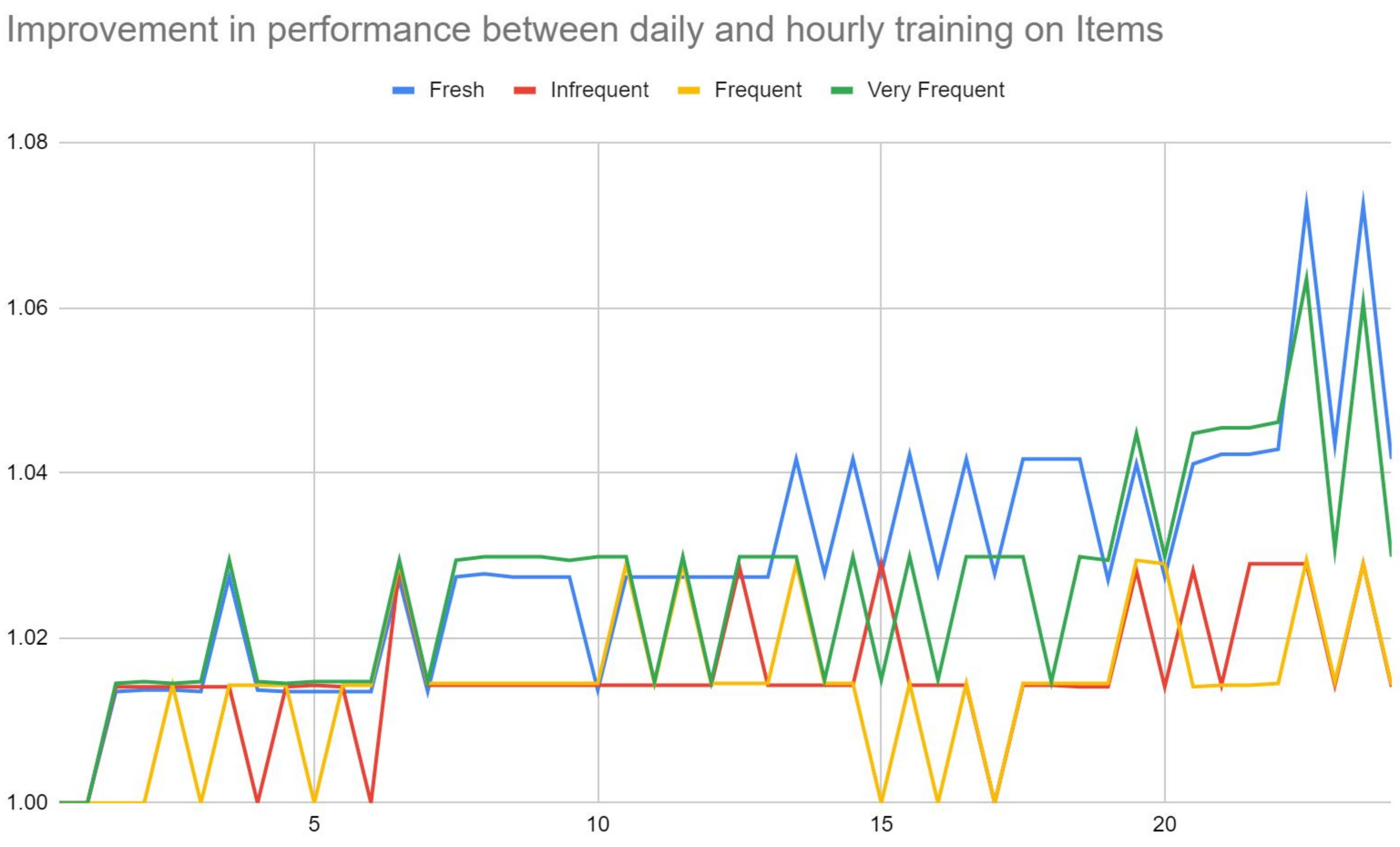
What items are most impacted?

Fresh items and infrequently accessed items



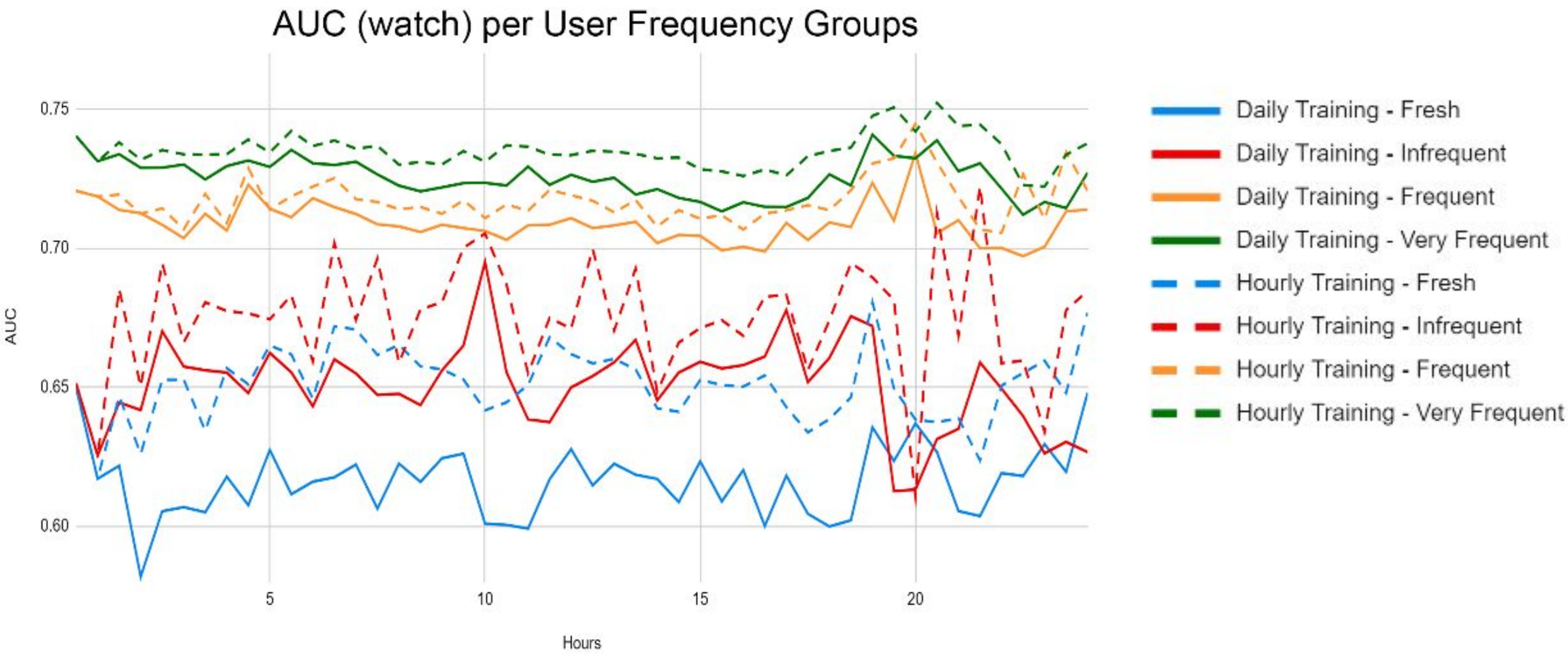
What items are most impacted?

Fresh items and infrequently accessed items



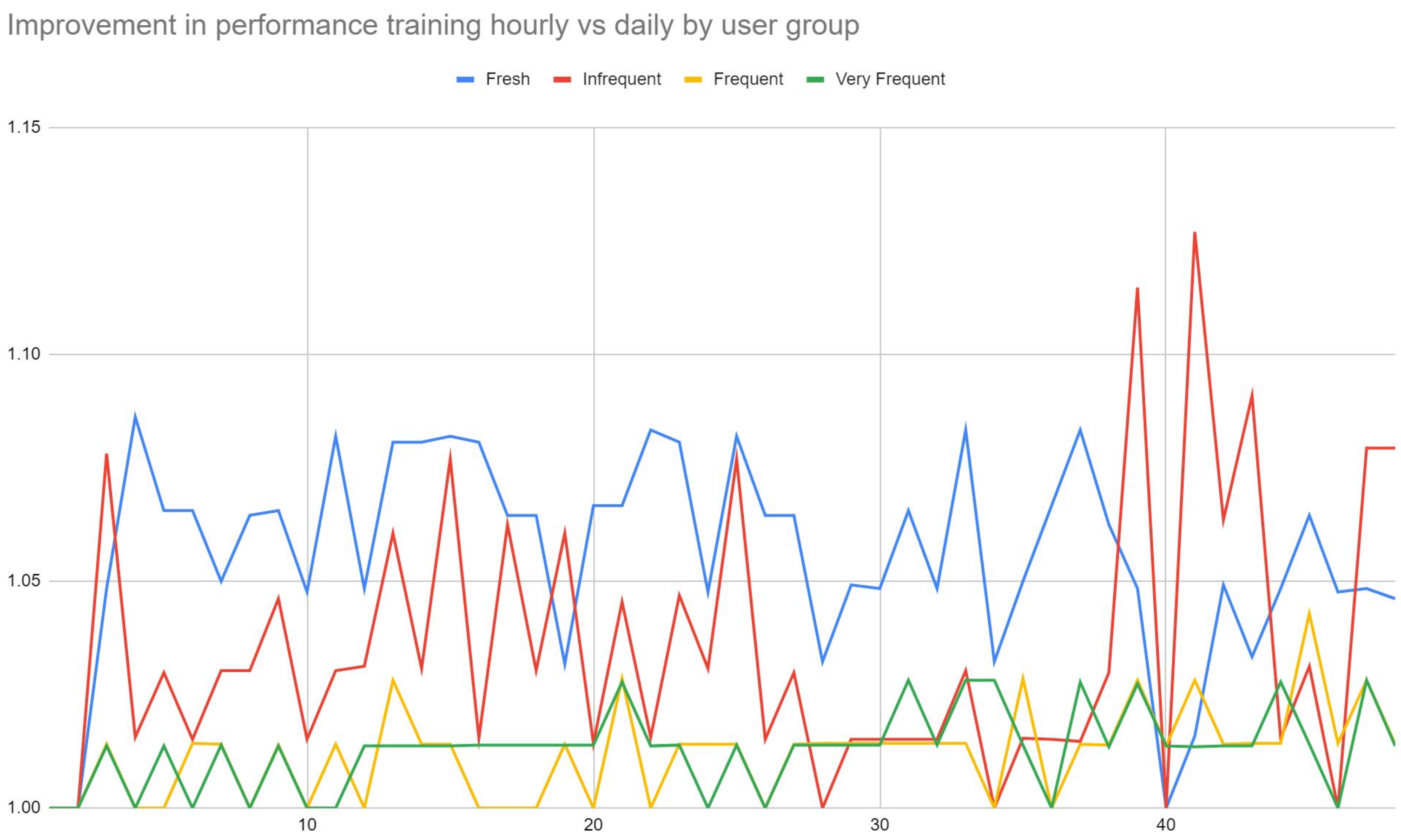
Which User Groups Are Impacted?

Across both feature representations and across model type we consistently see an improvement



Which User Groups Are Impacted?

Across both feature representations and across model type we consistently see an improvement



NVIDIA Merlin

Develop, deploy and maintain recommender systems

	Model Development	System Deployment	Production Maintenance
Who:	Data Scientists / ML Engineers	ML Engineers / Product Engineers	Product Engineers / ML Ops
Needs:	Quick iteration over feature engineering and model training	Easily deploying new models and workflows into production	Monitoring and maintaining many recommender systems
Merlin:	▪ Accelerates pipelines for fast experimentation cycle ▪ Integrates ETL and model training ▪ Implements common architectures, loss functions, sampling strategies, etc. ▪ Flexibility to build your own	▪ Simple API to push to production ▪ Deploys ETL and multi-stage models as ensemble ▪ Supports retrieval, filtering, and other common pipeline stages ▪ Scalable and accelerated components	▪ Standardize production workflow for all use cases ▪ Integration to other components for logging, feature storage, etc.

Four Common Components of Recommender Systems

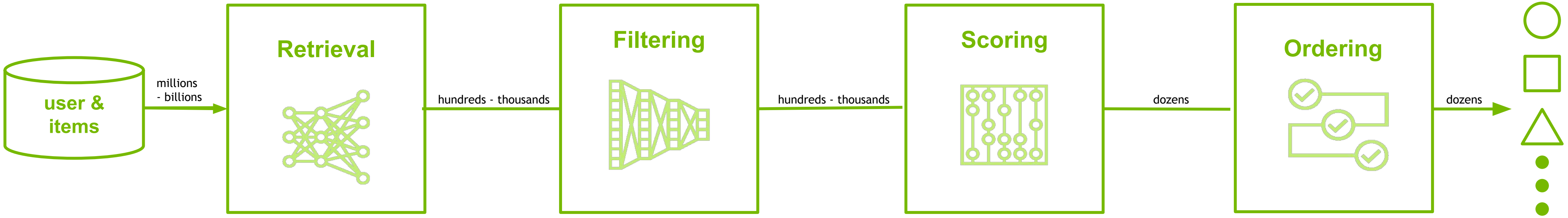
1. **Retrieval**: Fetch a **small set of candidate items** from the massive item catalog relevant for the current user

2. **Filtering: Remove candidate items that aren't appropriate** or available

3. **Scoring**: **Assign a** relevance **score** to each remaining item

4. **Ordering:** Choose which of the candidate items to include in the final list of recommendations and **put them in an optimal order**

Complex Multi-stage Recsys Pipeline



Recommender system consists of multi-stage pipeline

Each stage has multitudes of models and toolings, maintained by separate teams, with their individual KPIs.

Different stages are subsequently “chained” together with complex system engineering and MLOps.

It takes a whole village for an end-to-end system running smoothly!

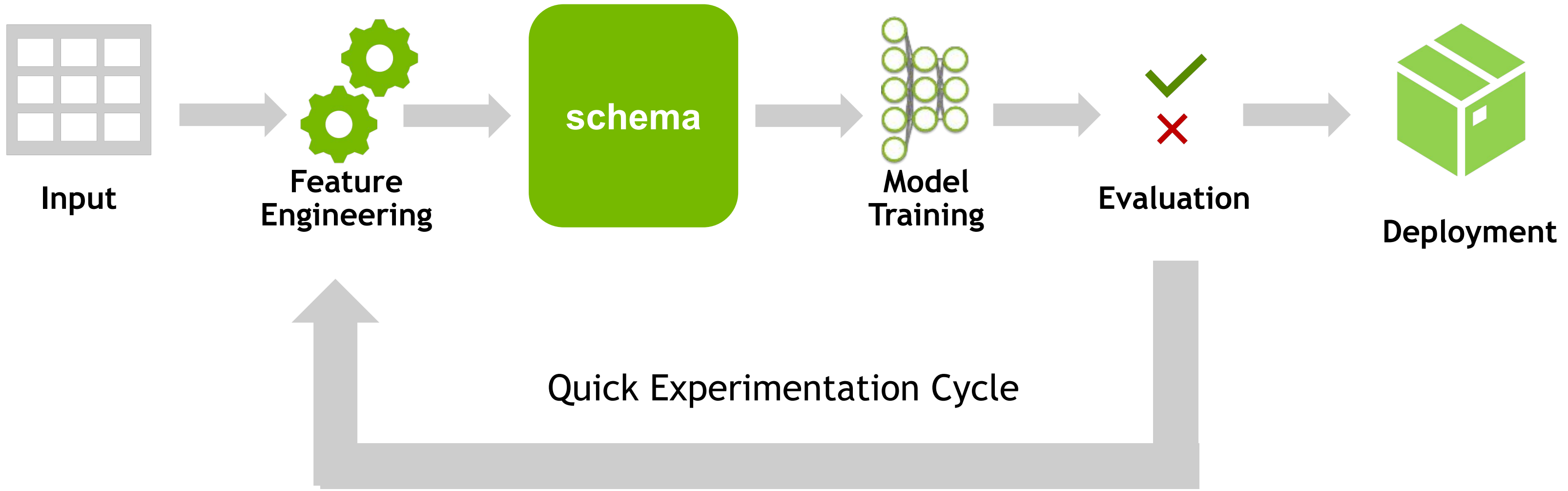
MERLIN MODELS

User Needs:

- Quickly and easily iterate over features and models to determine the best model for user's data and use-case

What is it:

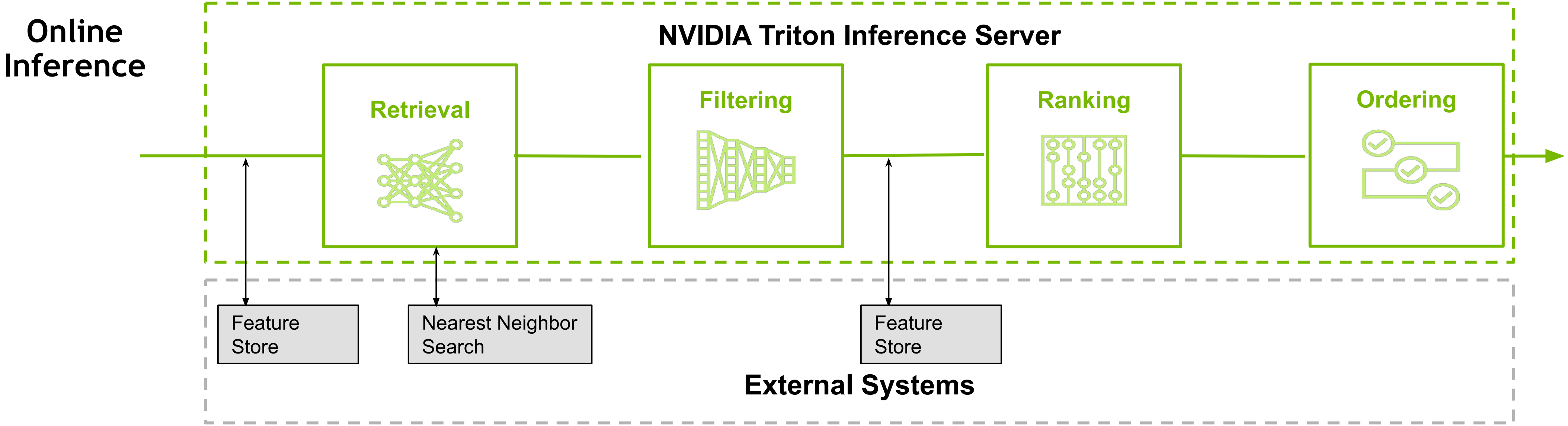
- Model training library w/ pre-defined model implementations and reusable building blocks
- Schema export from NVTabular during pre-processing
- High level model building block APIs - to build and train ML or DL models with 10 lines of code
- Cross FW (TF/XGBoost/Implicit/LightFM) model evaluation



MERLIN SYSTEMS

User Needs:
- Easily deploy pipeline with feature transforms, retrieval, & ranking as microservice w/ Triton with few lines of code

What is it :
- Triton ensemble configs to connect different stages together for on-prem/cloud deployment
- Create a pipeline with 50 lines of python code



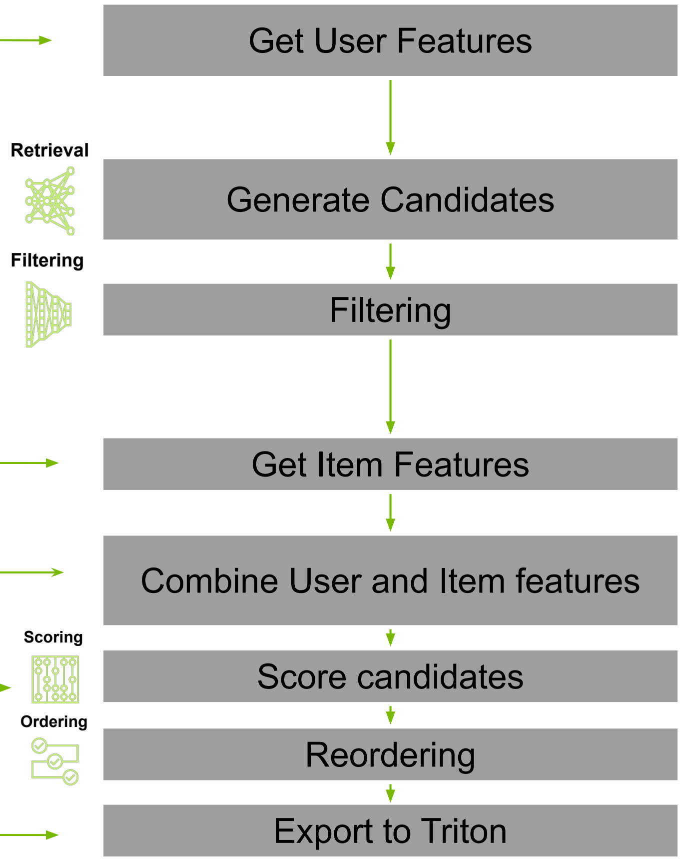
Merlin system abstracts all the ◆ into high level APIs to build an entire pipeline in < 50 lines of code

MERLIN SYSTEMS

Merlin Systems Python API (~50 lines)

```python
user_features = ["user_id"] >> QueryFeast(
    feast_repo_path,
    entity_view="user_features",
    entity_id="user_id",
    entity_column="user_id",
    features=["movie_id_count"],
    mh_features=["movie_ids", "genres", "search_terms"],
    input_schema=feast_user_in_schema,
    output_schema=feast_user_out_schema,
)

retrieval = (
    user_features
    >> PredictTensorflow(
        retrieval_model_path,
        custom_objects={"sampled_softmax_loss": sampled_softmax_loss},
    )
    >> QueryFaiss(faiss_index_path, query_vector_col="output_1", topk=100)
)

filtering = user_features["movie_ids_1"] + retrieval["candidate_ids"] >> FilterCandidates(
    candidate_col="candidate_ids", filter_col="movie_ids_1"
)

item_features = filtering >> QueryFeast(
    feast_repo_path,
    entity_view="movie_features",
    entity_id="movie_id",
    entity_column="filtered_ids",
    features=["tags_nunique"],
    mh_features=["genres", "tags_unique"],
    input_schema=Schema([ColumnSchema("filtered_ids", dtype=np.int32)]),
    output_schema=feast_item_out_schema,
    include_id=True,
    output_prefix="movie",
)

combined_features = user_features + item_features >> UnrollFeatures(
    "movie_id", feast_user_out_schema.column_names, unrolled_prefix="user"
)

ranking = combined_features >> PredictTensorflow(ranking_model_path)

ordering = (combined_features + ranking)["movie_id", "output_1"] >> SoftmaxSampling(
    "movie_id", relevance_col="output_1", topk=10, temperature=20.0
)

export_path = str("/nvtabular/test_poc/")

ensemble = Ensemble(ordering, request_schema)
ens_config, node_configs = ensemble.export(export_path)
```

Triton Pipeline



Core Merlin Principles

Easy to add to: All of our libraries are designed so that you can add your own components

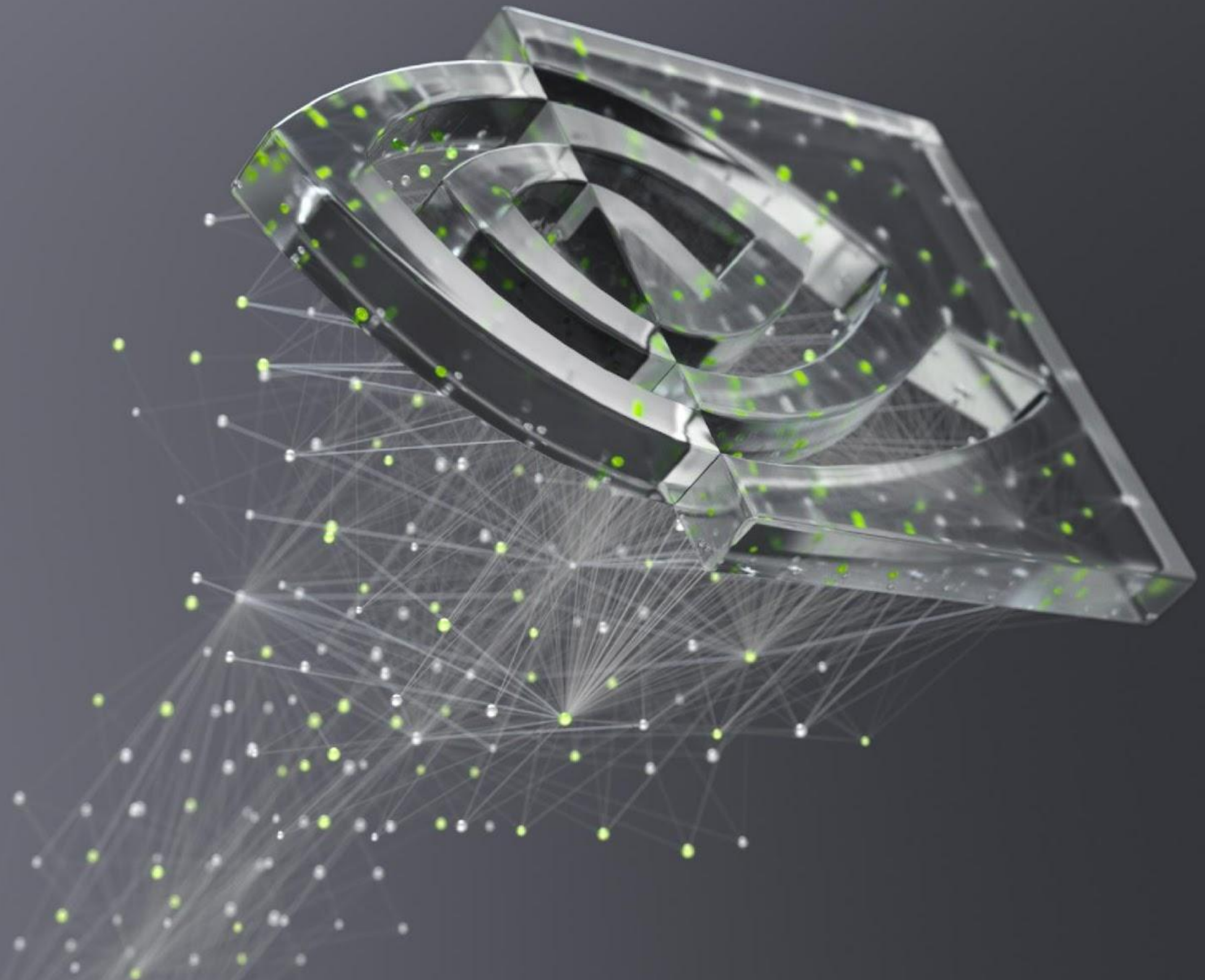
Composability: Components are easy to connect together into a more complex system

Industry Standard Design Patterns: Best practices are already implemented for you, allowing you to iterate quickly

NVIDIA-Merlin

NVIDIA Merlin Team

Providing easy to develop, performant, end to end recommender systems on the GPU



Thank You!

Even Oldridge
eoldridge@nvidia.com

Even_Oldridge

NVIDIA-Merlin

NVIDIA