Yandex Recommender System

Michael Roizner, roizner@yandex-team.ru
Machine Learning: Prospects and Applications
2015, Berlin, Germany.

Introduction

Yandex has a number of services that provide recommendations of various types of content such as music, movies, products, news, videos, articles, apps, etc.

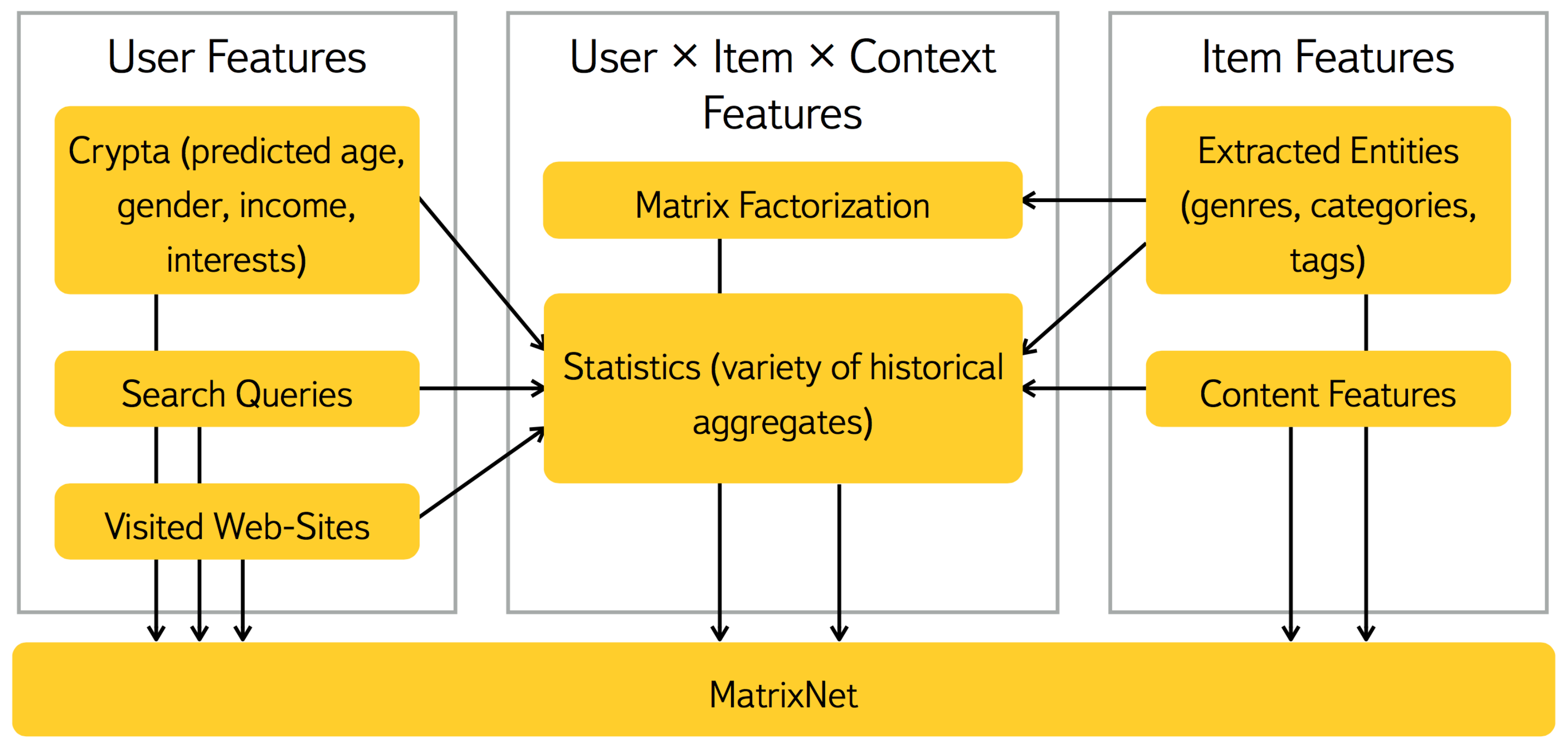
We present our technology that is used behind these services and provides personal recommendations in different domains.

Our system uses logs of interactions between users and items represented as a set of events. Each event is specified with user ID, item ID, event type (e.g., click, view, like), timestamp and some context features (e.g., location, weather, current radio station, etc.). All items may have different types, content features, and connections between each other (such as track—artist).

Recommendation Pipeline

1. A client makes a recommendation request to the system.
2. A set of candidates for recommendation is selected.
   - Sometimes the request implies a specific candidates set (e.g., all items satisfying specific filters, or all the accessories to a particular model).
   - Sometimes the set is too large (e.g., all items in the system) and needs to be pre-filtered with a cascade of algorithms (such as content-based, similarity-based, SVD, or a light-weight MatrixNet ranker).
3. All candidates are ranked by their predicted relevance.
4. Top candidates may further be reranked based on some business rules or whole page optimization algorithms to boost such properties as diversity, serendipity.
5. The resulted list of recommendations is sent to the client.

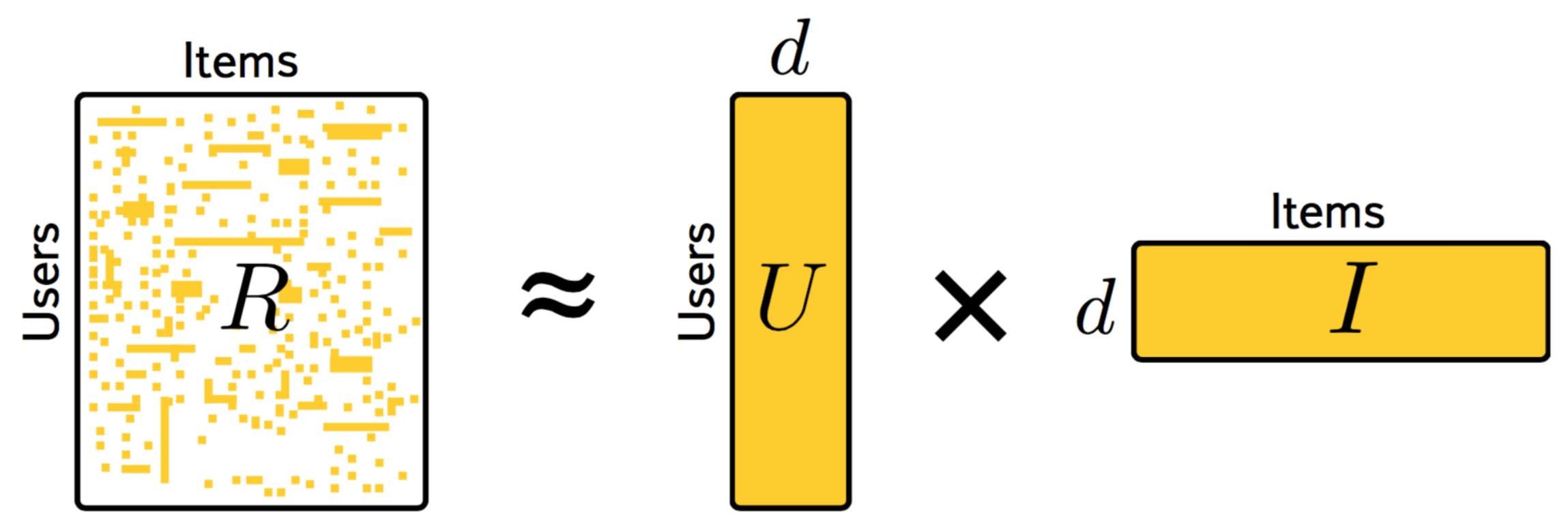
Core Relevance Prediction


User Features
Crypta (predicted age, gender, income, interests)
Search Queries
Visited Web-Sites
User × Item × Context Features
Matrix Factorization
Statistics (variety of historical aggregates)
Item Features
Extracted Entities (genres, categories, tags)
Content Features
MatrixNet

The system combines collaborative filtering with content-based recommendations.

- Relevance is predicted for different items and particular user in particular context.
- MatrixNet is used as a top-level machine learning algorithm.
- Search queries and visited web-sites can be used as external sources of user activity.
- Crypta provides such features as predicted user age, gender, income and interests.
- Different content features are available in different domains (e.g., track length, model price). Categorial features (e.g., artist, genre, category) may be represented as tags.
- Some entities may be retrieved from the content of the item (e.g., celebrities or brands mentioned in the text). They are also represented as tags.
- These tags may be used in matrix factorization and statistical aggregates.
- Overall, we use several hundreds of numeric features.

Matrix Factorization



Our implementations of matrix factorization algorithms are based on Singular Value Decomposition (SVD). All users and items are associated with their latent vectors. The predicted rating is calculated as

$$\hat{r}_{ui} = \mu + b_u + b_i + \langle p_u, q_i \rangle$$

The objective function optimised during learning is

$$\sum_{u,i} w_{ui}(\hat{r}_{ui} - r_{ui})^2 + \sum_{u}(\lambda_u \|p_u\|^2 + \lambda'_u b_u^2) + \sum_{i}(\lambda_i \|q_i\|^2 + \lambda'_i b_i^2)$$

Some specialities:

- Different aggregating functions based on history of interactions between the corresponding user and item are used to determine r_{ui} and w_{ui}.
- We use implicit feedback approach (Hu et al., 2008), i.e. we use $r_{ui} = 0$ and very small w_{ui} in a case of no interactions between the user and the item.
- Alternating Least Squares is used for optimization (Bell, Koren, 2007). It takes approximately 30 minutes to train a model using hundreds of millions of events.
- Items vectors are updated periodically (e.g., every night).
- Users vectors are updated in realtime. When we process a recommendation request we use the optimal user vector calculated using their latest history of events.
- We use information about connections between items and tags. For every pair of connected items i and j we add $\|q_i - q_j\|^2$ to the objective function.
- Since there are a lot of hyperparameters such as latent dimension and different regularizers, hyperparameter tuning is used to optimize desired metrics.

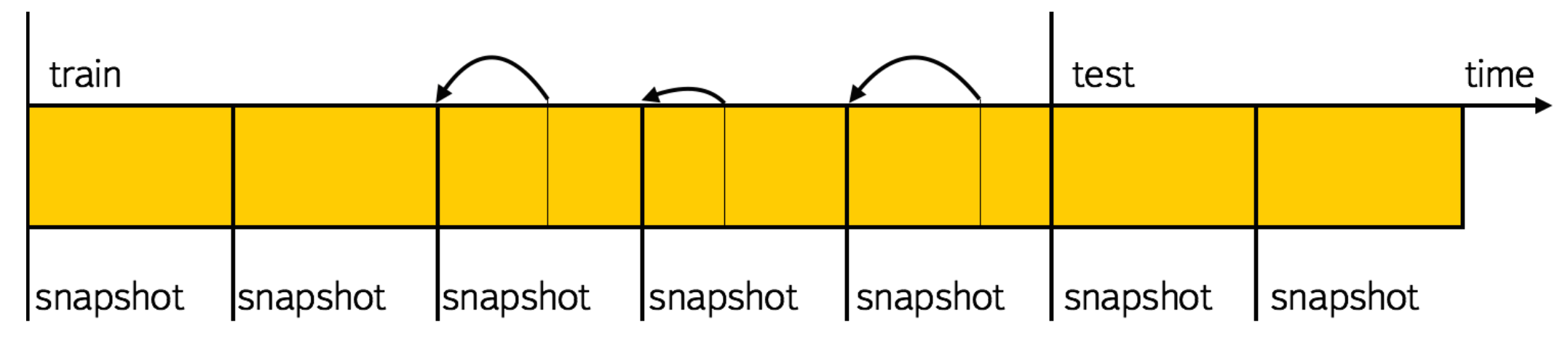
Statistical Aggregates

Here are some types of statistical aggregates of event history we use:

- Total number of events for the item (item popularity)
- Ratio of events of particular type (CTR, ratio of likes, etc.)
- Number of events for a specific time period (last day, last week, last month, etc.)
- Personal statistics: considering the events the requesting user had with the item
- Time features: when was the last event
- Using connections: considering the events with the connected items (e.g., number of likes of tracks of the same artist)
- Statistics of features: for any user property such as predicted age, we measure the average predicted age of all users interacted with this item, CTR for all users in a given age range, etc. The same for other user, item and context features.

Time Machine

All features using the latest user history are computed in realtime. All other features (such as items latent vectors and non-personal statistics) are stored in model snapshots. Model snapshots are trained periodically (e.g., every night). Using only the latest snapshot and entire users history for features in all the events in the MatrixNet training set would lead to overfitting. We solve this problem with Time Machine.



- Time Machine is used for MatrixNet training set creation and model evaluation.
- Time Machine basically simulates the system working in realtime on historical data.
- For every event in the set, the latest available snapshot and user history for that moment is used.

Optimized Metrics

In different cases the relevance predictor is trained in different modes and different metrics are measured and being optimized.

- When explicit predicted rating is needed (e.g., for a movie) the regression mode is used (MSE metric).
- For ranking based on predicted rating the ranking mode is used (NDCG metric).
- For optimizing block CTR, click probability is predicted with binary classification mode (AUC and log-likelihood metrics).
- Probability of a composite event (such as purchase) may be predicted as $\hat{P}(purchase) = \hat{P}(click) \cdot \hat{P}(purchase|click)$
- Average cost (of a click or a purchase) may be predicted as $\hat{E}(cost) = cost\text{-}of\text{-}purchase \cdot \hat{P}(purchase)$
- If no negative feedback is given we can rank positive feedback (or implicit feedback, e.g., web-pages user actually visited) against random items in the same context.

Results

We use our system in several Yandex services. Major results:

- On Yandex.Music and Yandex.Radio services average listening time per user approximately increased by **+25%** and user retention increased by **+50%**.
- On Yandex.Market the block with popular items got more than **+30%** in CTR and in revenue from personalization by our system.
- Recently launched Yandex.Zen is based entirely on recommendations from our system. After launch CTR increased by **+75%**.

Conclusion

We built a system for personal recommendations that has several key features:

- It learns in realtime.
- It combines collaborative filtering algorithms with content-based features.
- It is extensible. One can add new models to the ensemble.
- It supports different domains with different specifics and tasks.
- History of user events outside the recommendation service can be used.
- Different objective metrics for different tasks can be optimized.
- It supports applying business rules for those properties that can hardly be optimized with machine learning algorithms.
- It boosts discovery.

Acknowledgements

The system was designed and developed by Michael Roizner, Victor Lamburt, Igor Lifar, Andrey Danilchenko, Alexander Gutman with help from our colleagues at Yandex.