**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**School of Information and Communications Technology**

**PROJECT REPORT**

**SESSION-BASED RECOMMENDER SYSTEM FOR RETAIL PRODUCTS**

**Course**: Web Mining - Fall 2024

**Instructor**: Ph.D. Nguyen Kiem Hieu

**Group 1**

Đoàn Ngọc Cường - 20210141

Võ Đình Đạt - 20214890

Lê Trung Kiên - 20214907

Phạm Quang Trung - 20214935

Đoàn Thế Vinh - 20210940

*Hanoi, 2024*

[**1. PROBLEM INTRODUCTION 4**](#_xpoq5oxbzd1b)

[1.1. Background & Motivation 4](#_orzzxtrdq50p)

[1.2. Problem Description 5](#_ce0td020e5al)

[**2. DATA DESCRIPTION 6**](#_gqhk4g80tuqi)

[2.1. Overview 6](#_urf78oiu9ajw)

[2.2. Data Format 7](#_cucrl4qhfad3)

[2.3. Exploratory Data Analysis 7](#_lf2coebjqkxg)

[2.4. Data Preprocessing 11](#_706zrwsd4pgr)

[2.4.1. Labels Generation 11](#_m3izuvx407xw)

[2.4.2. Data Splitting 12](#_nvarwdtvocyf)

[2.4.3. Transformation to Parquet 12](#_hsagfxlzv4ul)

[2.4.4. Use of Polars 13](#_bc0kc05mxbdq)

[**3. METHOD DETAILS 14**](#_lqf0ftpvofkc)

[3.1. Retrieval - Reranker Approach 14](#_869899uidbig)

[3.1.1. Candidate generation 14](#_m1mez1j88tl5)

[3.1.2. Feature Engineering 15](#_q6y25qi8irgv)

[3.1.3. Ranking 18](#_39pe4d1c2bg5)

[3.1.4. Prediction 19](#_xcbtw0w65rh7)

[3.2. Graph-based Approach 19](#_77d64b9hb5n)

[3.3. RNN-based Approach 21](#_1iwlsb1u9ug9)

[**4. EVALUATION 23**](#_qpmhdmq83352)

[4.1. Metrics 23](#_ufc0lhjfa1mg)

[4.2. Results 25](#_cf2ivropocby)

[**5. DISCUSSION 26**](#_casqtf8uy9fr)

[5.1 Key Insights 26](#_xaxylkbqgqyi)

[5.2 Limitations 26](#_dug98w5goliw)

[5.3 Future Directions 27](#_l5sepa7d17cb)

[**6. CONCLUSION 28**](#_211e8vfzkjd2)

[**7. CONTRIBUTION 29**](#_4i3ryzo5ibls)

[**8. REFERENCES 30**](#_b7izlew16ejk)

# **1. PROBLEM INTRODUCTION**

## **1.1. Background & Motivation**

Digitalization is on the rise for all aspects of everyday life, and people’s shopping habits are no exception. Nowadays, more and more brands of various industries, along with supermarkets, convenience stores, and retail chains, offer an online alternative to shopping, in place of face-to-face on-premise interactions, leading to a dramatic shift in consumer behavior. E-commerce platforms have become integral parts of daily life, and this development offers an immense wealth of valuable data on user interactions and behaviors within a digital system, since each action, such as product clicks, additions to cart, or purchases, taken by a user can be tracked and logged onto a structured database for further analysis. In return, marketing teams or sales managers can make use of this information as a guide towards strategies and policies the entice customers into spending more, thereby helping to increase profit.



*Figure 1.1: A few of the various e-commerce platforms in existence nowadays*

It goes without saying that analyzing such data presents an opportunity for businesses to gain profound insights into customer preferences and decision-making processes. By leveraging these insights, companies can develop personalized marketing strategies, optimize product recommendations, and improve the overall shopping experience. A core component of these strategies is the use of recommender systems, which help match users with items they are most likely to engage with or purchase.

Among different approaches to recommending, **session-based recommender systems (SBRS)** are particularly valuable in contexts where user identification is not persistent, such as guest sessions or short-term interactions. Unlike traditional user-based recommenders that rely on historical data tied to individual users, SBRS focuses on understanding the intent and preferences expressed within a single session. This makes them ideal for environments with high user churn, where data privacy concerns or regulatory constraints may limit the use of long-term user profiles.

Among the existent datasets available to the public, the **OTTO Recommender Systems Challenge** on Kaggle centers on this critical topic, tasking participants with developing innovative algorithms to predict the next actions (e.g., clicks, add-to-cart, or purchase) within a session. The challenge reflects real-world scenarios faced by online retailers and demonstrates the importance of SBRS in driving customer engagement and boosting sales.

**Motivation**

In the process towards deciding a topic for this Capstone project, the problem of Session-based recommendation systems align closely with topics covered in the course, and the group wants to use this opportunity to bridge the gap between the theoretical teachings and the implementation of an actual system utilizing such principles. In addition, in an era of fierce competition among e-commerce platforms, recommendation systems have become a cornerstone of business strategies. Success stories from companies like Amazon and Netflix highlight the potential of such systems to enhance user experience and drive revenue growth. With this in mind, the group feels that this project can serve as a chance to enhance the industry-wise applicability of what is learned during the lectures. Finally, this topic also requires us to make use of our analytics skill in order to understand this data, in contrast to simpler and more popular datasets. These reasons warrant the choice of this topic.

## **1.2. Problem Description**

The challenge involves predicting user actions within an ongoing session, based on sequential event data. Each session is represented as a sequence of events, where each event includes information such as *product ID*, *timestamp*, and *action type* (e.g., click, add-to-cart, purchase). Given the partial sequence of events, the goal is to **accurately predict the remaining actions in a deliberately truncated session**, a typical setup of a Session-based Recommender System.

Specifically, the Otto dataset is provided in chronological order for each session, simulating the temporal nature of user interactions. In terms of the outcome to be achieved, the challenge involves three prediction tasks: clicks, add-to-cart actions, and purchases. The complexity of the problem is heightened due to the fact that the dataset is extremely large-scale, necessitating efficient algorithms capable of handling millions of sessions and events revolving almost two million products. Details regarding the dataset

# **2. DATA DESCRIPTION**

## **2.1. Overview**

The **OTTO Recommender Systems Dataset** is a comprehensive collection of anonymized user behavior logs from the OTTO e-commerce platform, designed to serve as a benchmark dataset for research in session-based recommendation systems.

The dataset comprises approximately **12 million real-world user sessions**, with any identifiable details of users removed. Each session represents a sequence of user interactions on the OTTO retail site within a specific timeframe - all sessions are from 2022 and are of varying length. The OTTO group collects such information, from which data analytics and machine learning models can be implemented to derive insights into user navigation patterns and decision-making process when they shop online. It is also confirmed that each session is of a unique user, so there are correspondingly 12 million users appearing in the dataset; however, this will not matter much since the data itself is inherently anonymized.

Across sessions, the dataset encompasses around **220 million events**, with loggings of three primary types of actions that users of the site can take. Details of these actions are elaborated on as below.

| **Action** | **Description** |
| --- | --- |
| Clicks | Short for “clicking a product”. These are events where a user clicks on a specific product, from which detailed information are presented to them. |
| Carts | Short for “adding to cart”. These are events where a user adds an item onto their shopping cart, usually indicating a higher level of intent to purchase. Most e-commerce sites offer a virtual cart - users can add multiple items before heading to check-out for payment. |
| Order | Short for “ordering a product”. These are complete purchases or full transactions. Business terminology considers this a successful conversion. |

*Table 2.1. Three action types and what they represent.*

In addition to events within a session, a crucial piece of information is the unique articles (or products/items). The dataset includes approximately **1.8 million articles**, a diverse catalog offered on the inventory of the OTTO platform. It is also notable that aside from article IDs, we are offered no extra information on the inherent nature of the products - no names, descriptions, or categories. All articles are nameless and stateless entities, only known for their unique distinction from one another. The detailed statistics regarding the dataset are as follows.

| **Dataset** | **#sessions** | **#items** | **#events** | **#clicks** | **#carts** | **#orders** |
| --- | --- | --- | --- | --- | --- | --- |
| Train | 12.899.779 | 1.855.603 | 216.716.096 | 194.720.954 | 16.896.191 | 5.098.951 |
| Test | 1.671.803 | 1.019.357 | 13.851.293 | 12.340.303 | 1.155.698 | 355.292 |

*Table 2.2: Specific number of items and action counts across the dataset*

## **2.2. Data Format**

The dataset is structured in JSON Lines (.jsonl) format, where each line represents a single session with the following structure:



*Figure 2.1: Sample of the dataset.*

In this sample, ***session*** is a unique identifier for the user session, while ***events*** is a list of events that occurred during the session, each containing: ***aid***, the article (product) identifier; ***ts***, the timestamp of the event in Unix milliseconds, and ***type***, the type of interaction (one among *clicks*, *carts*, or *orders*)

## **2.3. Exploratory Data Analysis**

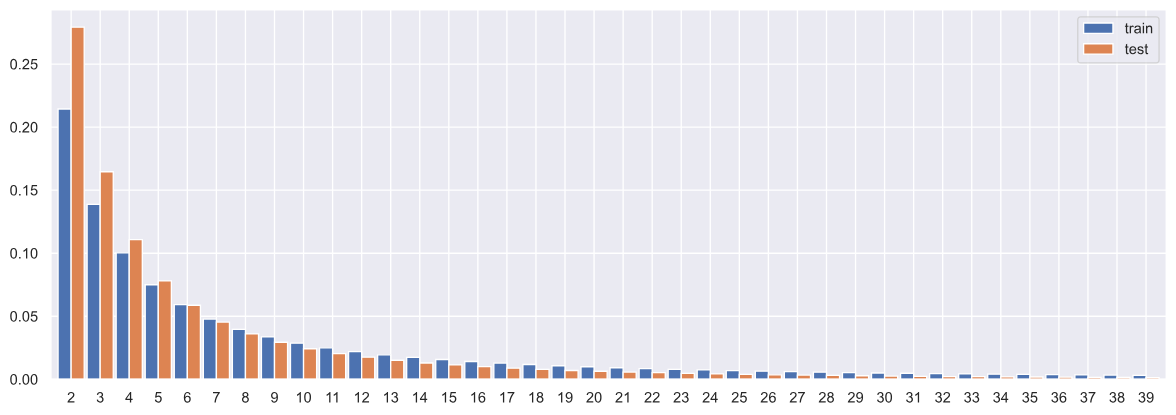
In this section, we delve deeper into the nature of the dataset and offer some insight, via notable statistics, into the size and scale of the dataset.

**Distribution of Events per session**

The dataset exhibits a wide range of events per session, with a mean of **16.80** and a standard deviation of **33.58** in the training set. The median number of events per session is **6**, indicating a right-skewed distribution where a significant number of sessions have relatively few interactions, while a smaller proportion includes a high volume of events.

|  | **mean** | **std** | **min** | **50%** | **75%** | **90%** | **95%** | **max** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Train #events per session | 16.80 | 33.58 | 2 | 6 | 15 | 39 | 68 | 500 |
| Test #events per session | 8.29 | 13.74 | 2 | 4 | 8 | 18 | 28 | 498 |

*Table 2.3: Number of events per session, at various percentiles, for the train - test sets*



*Figure 2.2: Frequency of distinct number of events per session, for both the train and test sets. As can be seen, a large proportion of the sessions tend to be shorter (under 20 events)*

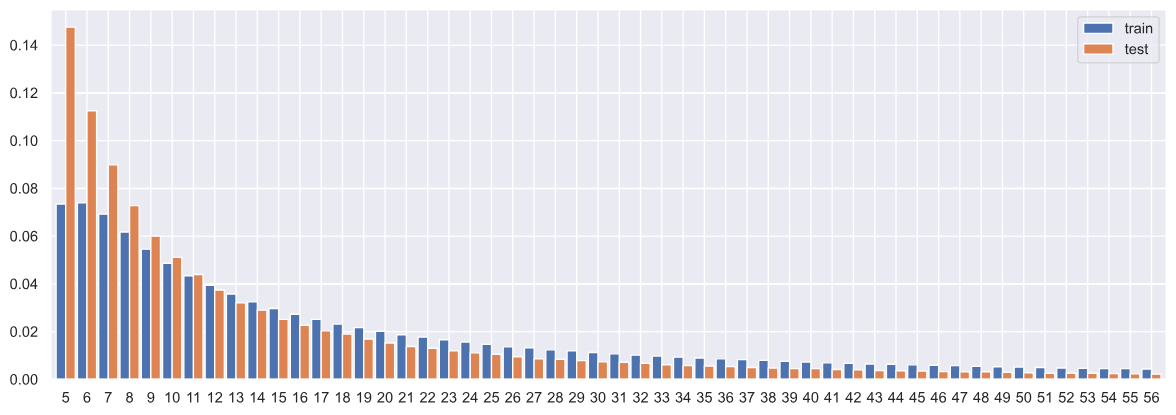
From Table 2.3 and Figure 2.2, it can be observed that the number of events per session tend to fall on the shorter side, with a large proportion having under 20 events. The frequencies seen on the test set are also higher on the left-hand side due to the fact that they are truncated, thereby more likely to be even shorter.

**Distribution of Events per item (article)**

Next, the average number of events per item is approximately **116.79**, with a substantial standard deviation of **728.85**, suggesting that while some items receive consistent attention, others experience sporadic or minimal interactions. The median number of events per item is 20, further highlighting the skewness in item interaction distribution.

|  | **mean** | **std** | **min** | **50%** | **75%** | **90%** | **95%** | **max** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Train #events per item | 116.79 | 728.85 | 3 | 20 | 56 | 183 | 398 | 129,004 |
| Test #events per item | 13.59 | 70.48 | 1 | 3 | 9 | 24 | 46 | 17,068 |

*Table 2.4: Number of events per item, at various percentiles, for the train - test sets*

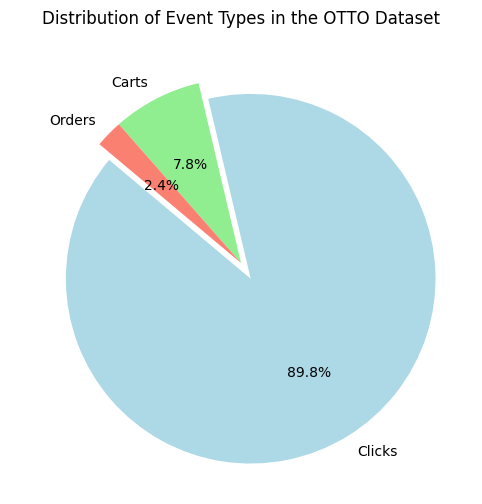


*Figure 2.3: Frequency of distinct number of events per item, for both the train and test sets. Once again, a large proportion of the items are involved in very few events (under 40 events)*

From Table 2.4 and Figure 2.3, it can be observed that the number of events involving a particular item tend to fall on the shorter side, with a large proportion of items being in under420 events. The frequencies seen on the test set is also higher on the left-hand side due to the fact that the test set truncates some events, thus more likely to reduce the presence of some items.

**Distribution of Event types**

Constituting the majority of events, clicks account for about **89.8%** of the total interactions, reflecting users' initial engagement with products. Representing approximately **7.8%** of events, cart additions indicate a higher level of purchase intent. Making up around **2.4%** of events, orders signify completed transactions and are the primary metric for conversion.



*Figure 2.4. The distribution of Event Types in the OTTO Dataset. It is highly clear that product clicks are the most common, followed by cart additions then complete orders, which follows the intuitive understanding that only a small proportion of considered items are purchased by users*

**Density of User-product interactions**

With a density of **0.0005%** (calculated as the ratio of the number of observed interactions between distinct user-product pairs to the total number of possible interactions) , the dataset is highly sparse, indicating that only a small fraction of possible user-item interactions are observed. This sparsity presents challenges for modeling and necessitates robust techniques to effectively capture user preferences. Overall, these findings will serve as natural guidance towards the modeling process, in order to shape practical solutions that can manage the dataset’s unique characteristics and challenges.

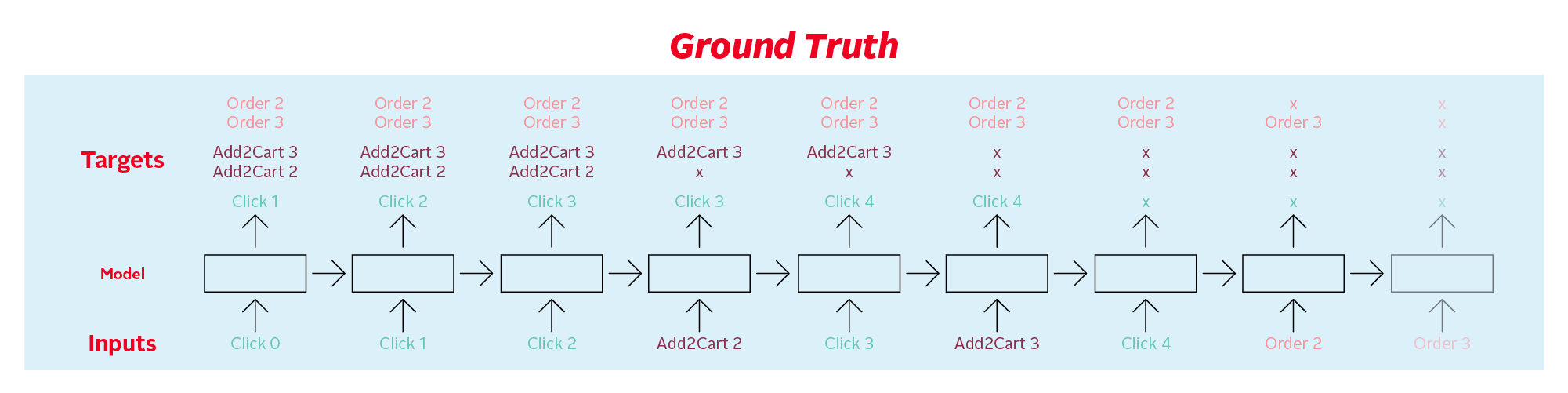
**Other distinctive characteristics**

* Each session is of a unique user, completely disjunct
* All test items are included in the train set, as confirmed by the dataset owner
* A session may start with any action type, even an “order”, because of the period of data extraction or the use of a wishlist
* The article IDs are completely anonymized and cannot be traced back to the characteristics or content of that product

## **2.4. Data Preprocessing**

Due to the dataset’s large scale and size, it is imperative that we find a systematic approach to handling and preprocessing the data for further solutions.

### 2.4.1. Labels Generation



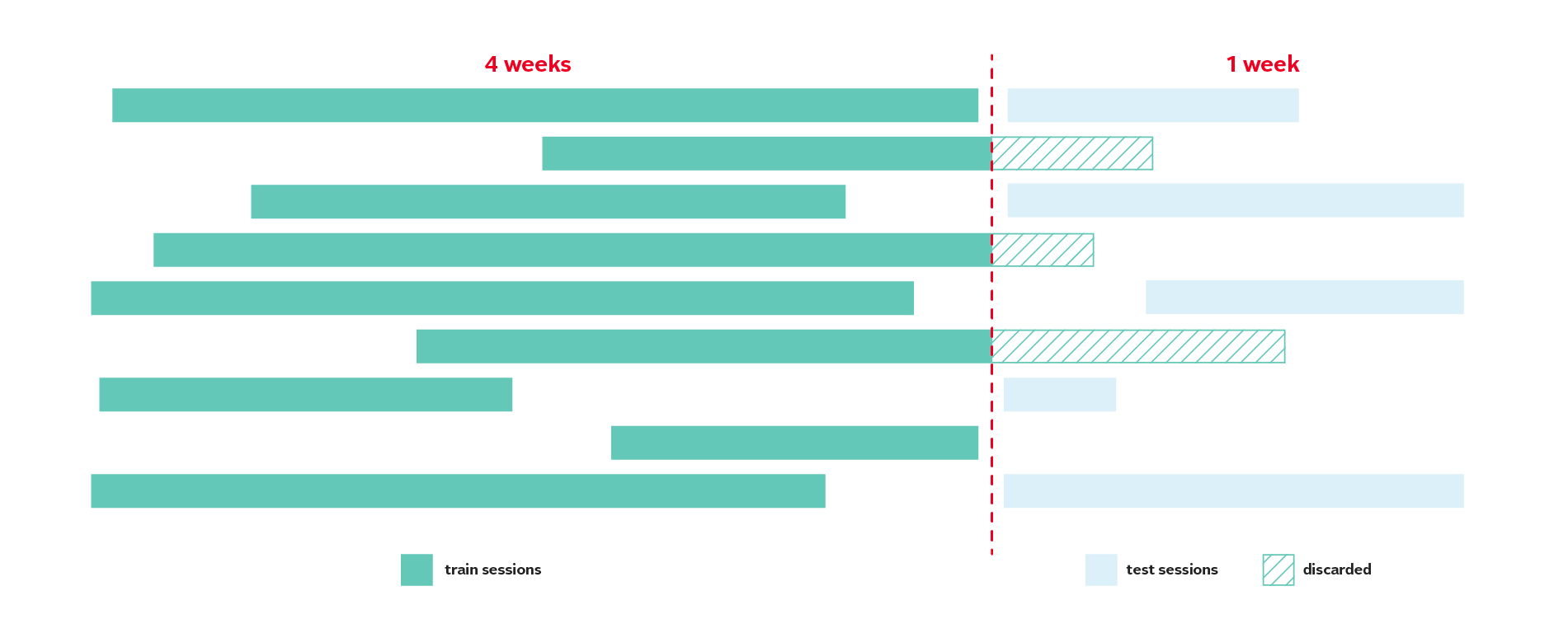
*Figure 2.5: Each input event sequence (e.g., clicks and add-to-cart actions) is paired with the corresponding targets (immediate next click & future add-to-cart or order actions).*

The corresponding labeled session in json is as follows. Each step has labels having all three action types: clicks, carts, and orders. Note that only the immediate next click is included, because in evaluating, only the immediate next click is considered as ground truth.



Label generation for training and validation involves extracting meaningful targets from session events. Each session is split into two parts: the earlier events form the input sequence, while subsequent events are used to generate the ground truth labels. For example, consider a session where the events are a sequence of "click\_0", "click\_1", "cart\_3", "order\_2", and "order\_3" For each input event, the model generates predictions that are compared to the corresponding targets. Initially, "click\_0" and "click\_1" may predict subsequent events like "cart\_3", "order\_2", or “order\_3”. As the sequence progresses, the targets dynamically change to represent the most recent events (e.g., “order\_2” will no longer have any “click” or “cart” to predict, plus “order\_3”). This approach ensures that the model learns to predict progressively more complex actions based on earlier behavior within a session.

### 2.4.2. Data Splitting



*Figure 2.6: Sessions occurring in the last week are used for testing (light blue), while sessions from the preceding four weeks are reserved for training (teal). Events spanning the split boundary are trimmed*

To create training and validation sets from the dataset, we prepared a script. This script splits the original training data based on a specified time frame. Specifically, a splitting timestamp is calculated by subtracting the duration of the test period from the maximum timestamp in the dataset. Sessions with events occurring after this timestamp are assigned to the test set, while those entirely before it are included in the training set. This method ensures temporal consistency between training and testing data.

During the splitting process, training sessions are trimmed to exclude any events occurring after the splitting timestamp to avoid any data leakage. Also, sessions with fewer than two events are discarded to ensure sufficient context for training. For the test set, events involving unknown items (AIDs not seen in the training set) are removed, ensuring the test evaluation remains realistic by focusing only on items encountered during training.

### 

### 2.4.3. Transformation to Parquet

The dataset is provided in .jsonl format, where each line represents a session with its associated events. Converting this data into a more efficient format, such as Parquet, can significantly enhance data loading and processing performance.

Specifically, we prepared a script that reads the input .jsonl file in chunks, allowing it to handle large datasets without exhausting system memory. For each session, it extracts individual events (clicks, carts, orders) and organizes them into a tabular format with columns such as session, aid (article ID), ts (timestamp), and type (event type).

The script maps event types to integer identifiers and converts timestamps from milliseconds to seconds, ensuring efficient storage and processing. The processed data is then written to Parquet files, which offer efficient columnar storage, improving both read and write performance.

Converting data to Parquet format is advantageous because Parquet's columnar storage is optimized for analytical queries, reduces storage space through efficient compression, and accelerates data loading times, which is particularly beneficial when working with large datasets.

### 

### 2.4.4. Use of Polars

Polars, a high-performance DataFrame library, is employed in the preprocessing pipeline to handle data efficiently. Polars is designed for speed and can outperform traditional libraries like pandas, especially with large datasets. Its benefits include:

* **Performance**: Polars is optimized for parallel execution and efficient memory usage, enabling faster data processing.
* **Lazy Evaluation**: It supports lazy evaluation, allowing for query optimization and reduced computational overhead.
* **Expressive Syntax**: Polars provides a concise and expressive syntax for data manipulation, enhancing code readability and maintainability.

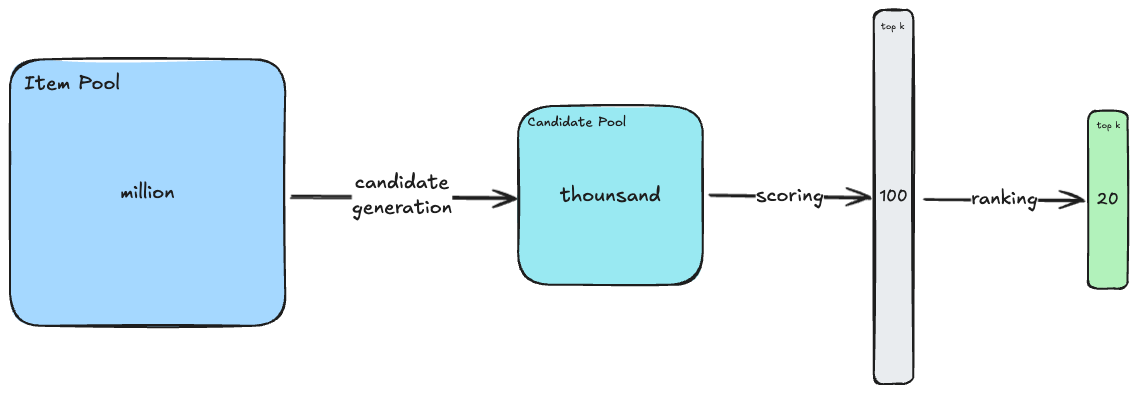
By leveraging Polars, the preprocessing scripts can efficiently handle large-scale data transformations, ensuring that the dataset is prepared for model training and evaluation in a performant manner.

In summary, the preprocessing pipeline involves converting the dataset into an efficient format, splitting it into training and validation sets based on temporal criteria, and utilizing Polars for high-performance data manipulation. These steps are crucial for preparing the dataset for developing and evaluating recommender system models.

### 

# **3. METHOD DETAILS**

## **3.1. Retrieval - Reranker Approach**



*Figure 3.1: Illustration of the general workflow*

The method follows the classical Retrieve (Candidate Generation) then Rerank architectural pattern to get the top 20 items potential for customers to click on, add to cart, or complete order during the session. Approximately 100 candidates are retrieved per session using a combination of session-based heuristics, co-visitation statistics, followed by ranking and selecting the top 20 items for clicks, carts, and orders.

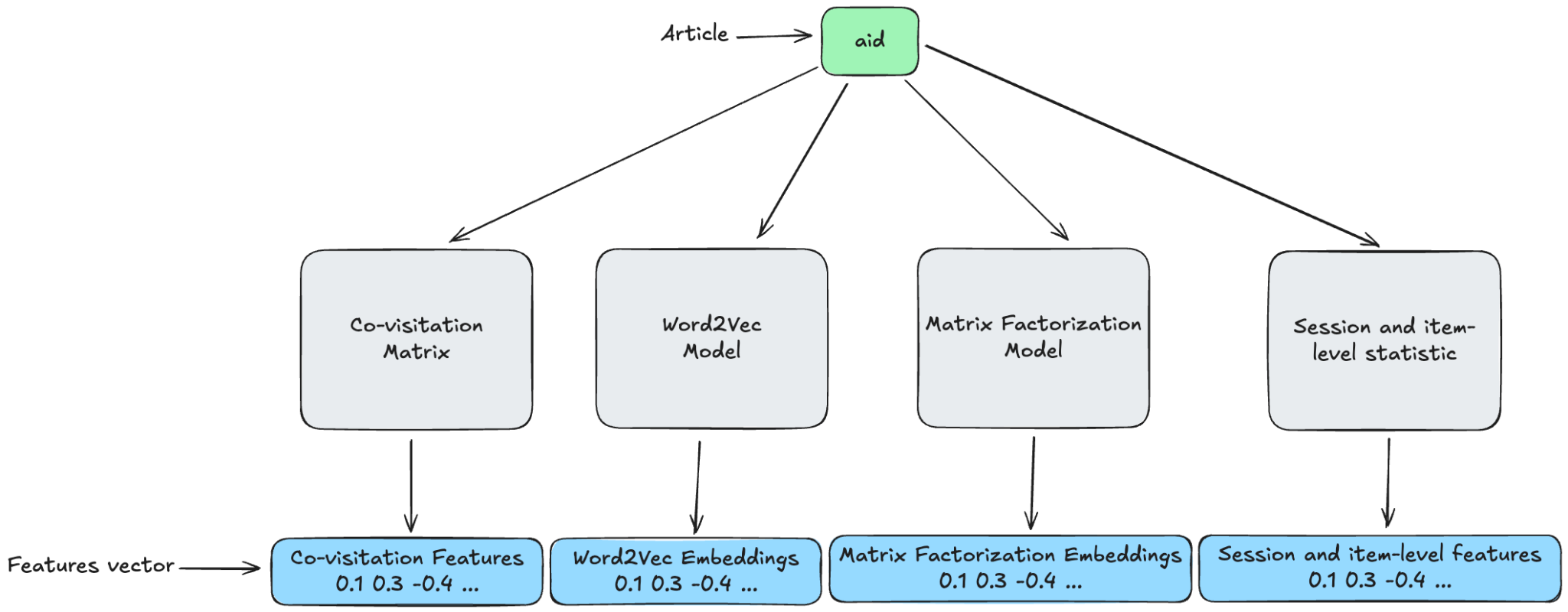
### 3.1.1. Candidate generation

The retrieval process incorporates multiple sources to maximize diversity and relevance:

* **Co-visitation matrix**: Add items frequently visited together. Then we add the 20 most frequent items per each previous item. We utilized several interactions between items listed below to create the co-visitation matrix.
  + Click-to-click: items are clicked together in the span of 12 hours.
  + Click-to-cart-or-buy: one item is clicked and then the other is added to cart or ordered in the span of 24 hours.
  + Cart-to-cart: items are added to cart together in 24 hours.
  + Cart-to-buy: one is added to cart and then the other is ordered in 24 hours.
  + Buy-to-buy: items are bought together in a 24-hour interval.
* **Weighted Co-visitation Matrix:** In addition to the previously mentioned matrices, we can enhance the item pair scoring by incorporating weights based on temporal factors or action types, resulting in the generation of additional co-visitation matrices.This approach recognizes that time can influence item recommendations within each session, as more recent interactions may carry greater relevance. Furthermore, different actions have varying levels of importance, which can be reflected using a weighting scheme These considerations allow for the generation of more nuanced and effective co-visitation matrices.

### 3.1.2. Feature Engineering

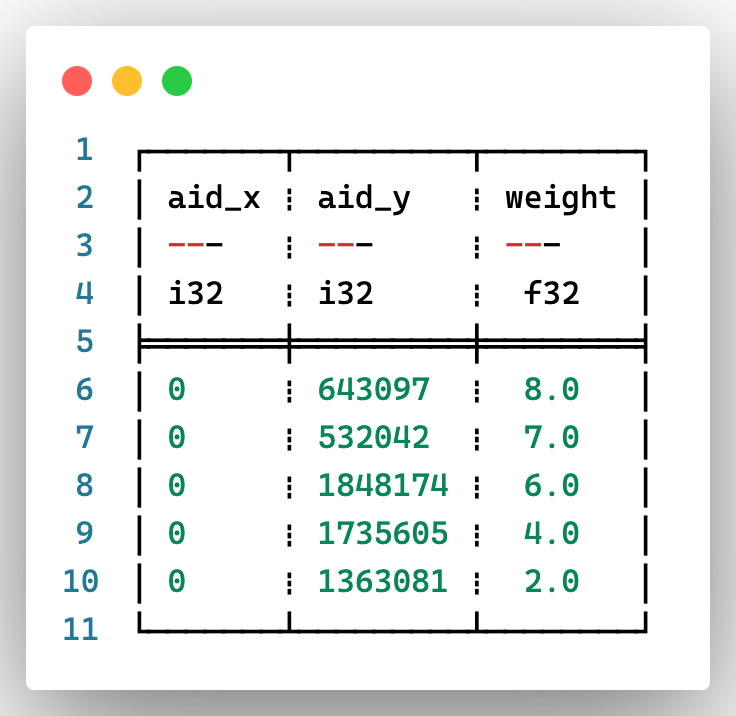
We created approximately 200 features to capture diverse session, item, and interaction-level characteristics.



*Figure 3.2: Overall feature engineering process.*

**Covisitation Features**

Each co-visitation matrix can be used as both for candidate generation and feature engineering, using its weights as features.



*Figure 3.3: Sample of Co-visitation Matrix*

**Embedding Features**

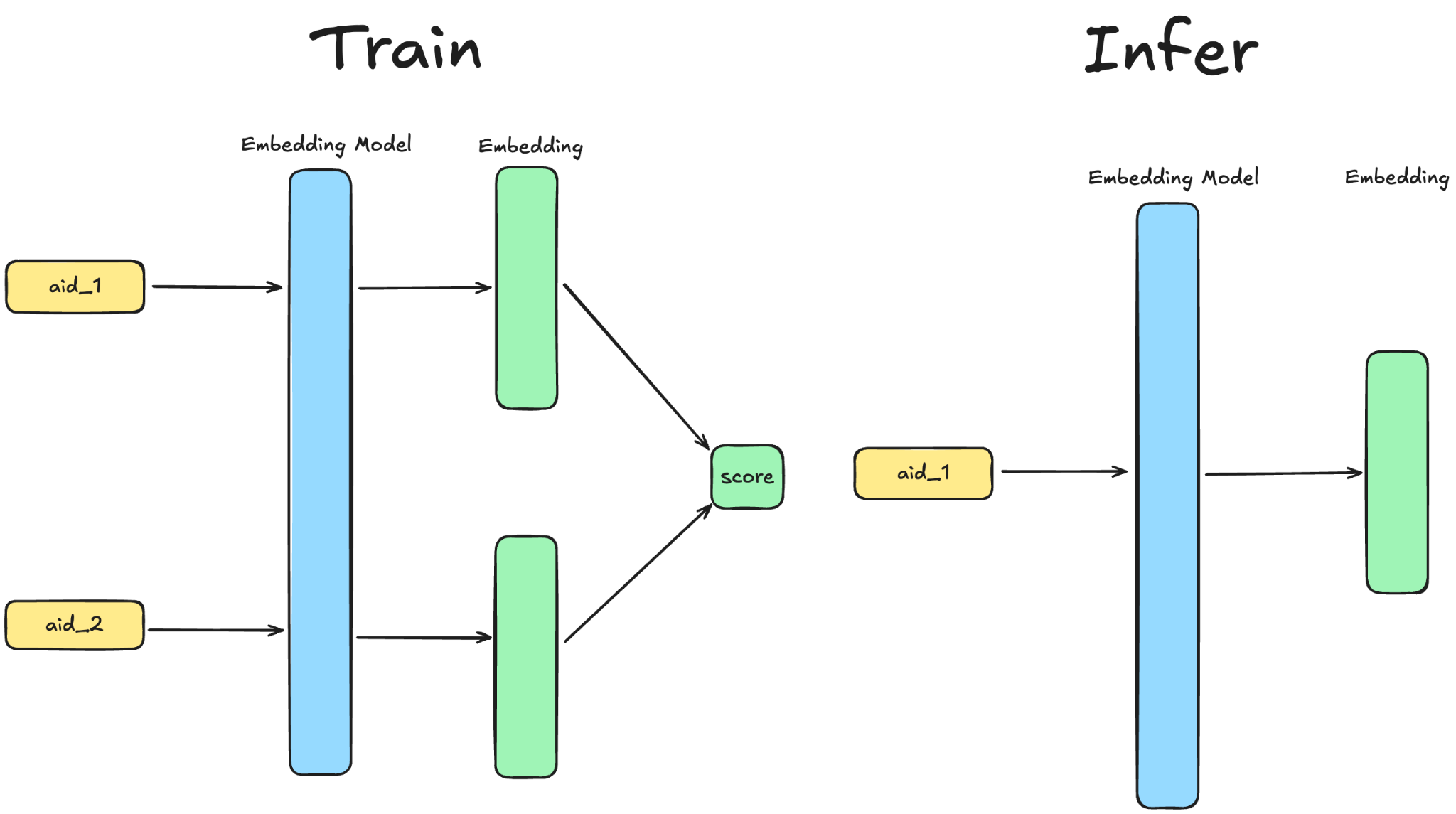
In addition, a number of vector features can be derived as follows.

* **Word2Vec-based Embeddings:** A Word2Vec model was trained using article IDs (aids) from each session to generate embeddings. These embeddings were concatenated and integrated as features for the re-ranking model.



*Figure 3.4: Setup and training Word2Vec model*

* **Item-Item Matrix Factorization:** We can train a simple Matrix Factorization model that inputs a couple of *aid* and *aid\_next*(the next article in the same session) and get the embedding as features.



*Figure 3.5: Basic Matrix Factorization model.*

**Session-Level Features**  
Features were aggregated over sessions to capture temporal and behavioral patterns.

* **Timestamp Statistics**: Aggregated metrics such as mean, min, max, and standard deviation of interaction timestamps.
* **Interaction Frequency**: Count and rank of candidate selection events within sessions.
* **Item-Level Features**: Features describing individual item interactions and properties, such as:
* **Candidate Characteristics**: Metrics like candidate score, rank, and whether the item was selected.
* **Interaction Time Distribution**: Mean hour of interactions (e.g., clicks, carts, orders) aggregated over items.

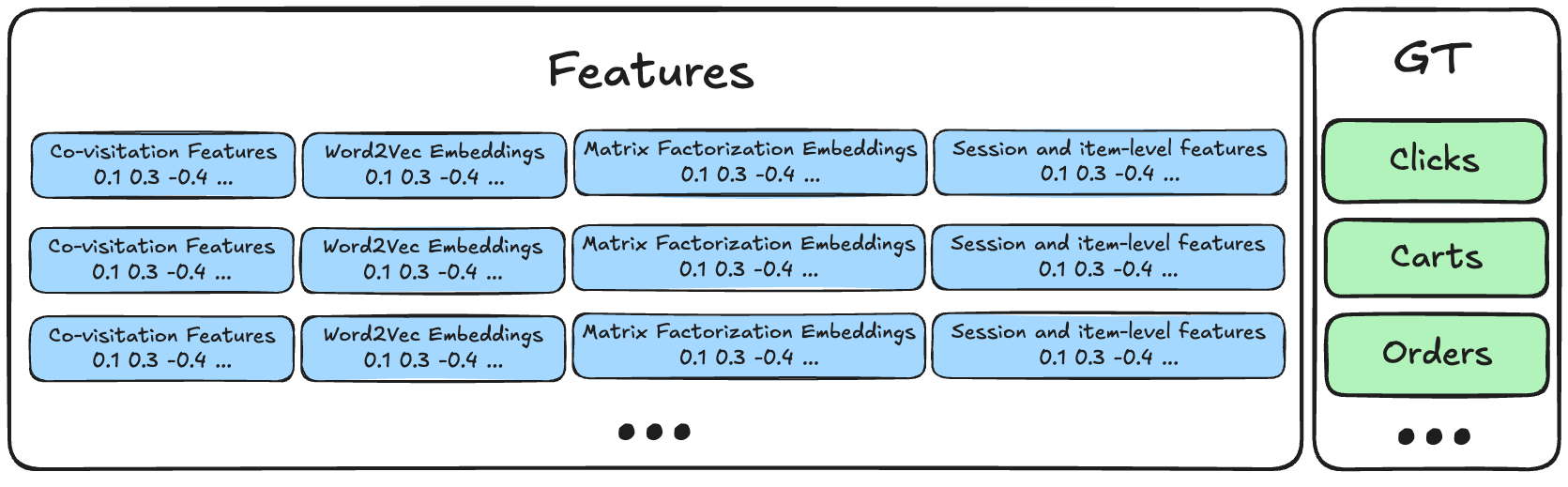
**Interaction Statistics**

* Aggregated counts of interaction types (e.g., clicks, carts, orders) over various time windows.
* Multi-action probabilities for each aid, capturing the likelihood of specific actions .

**Temporal Dynamics**  
Temporal features ensured the model could account for recency effects and patterns over different periods.

* **Session-specific Features**:
  + Interaction counts by type (click, cart, order) and their ranks.
  + Time-based features, such as time since the last interaction of a specific type.
  + Relative position and rank of items within the session.
* **Co-occurrence and Similarity Features**
  + Co-occurrence counts and ranks (e.g., frequency of items being clicked or carted together).
  + Word2Vec similarity metrics (Euclidean distance and ranks).
  + Similarity between session embeddings and item embeddings (cosine similarity, Euclidean distance).
* **Popularity Features**: Item popularity rank within the same session cluster.

### 3.1.3. Ranking



*Figure 3.6: Example of data used to train Ranking models.*

**Target Definition**

Each retrieved item is assigned a binary label.

* **1**: If the item was clicked, added to the cart, or ordered.
* **0**: If none of these actions occurred.

**Data Filtering**

Sessions without any positive samples were removed, resulting in a reduced dataset.

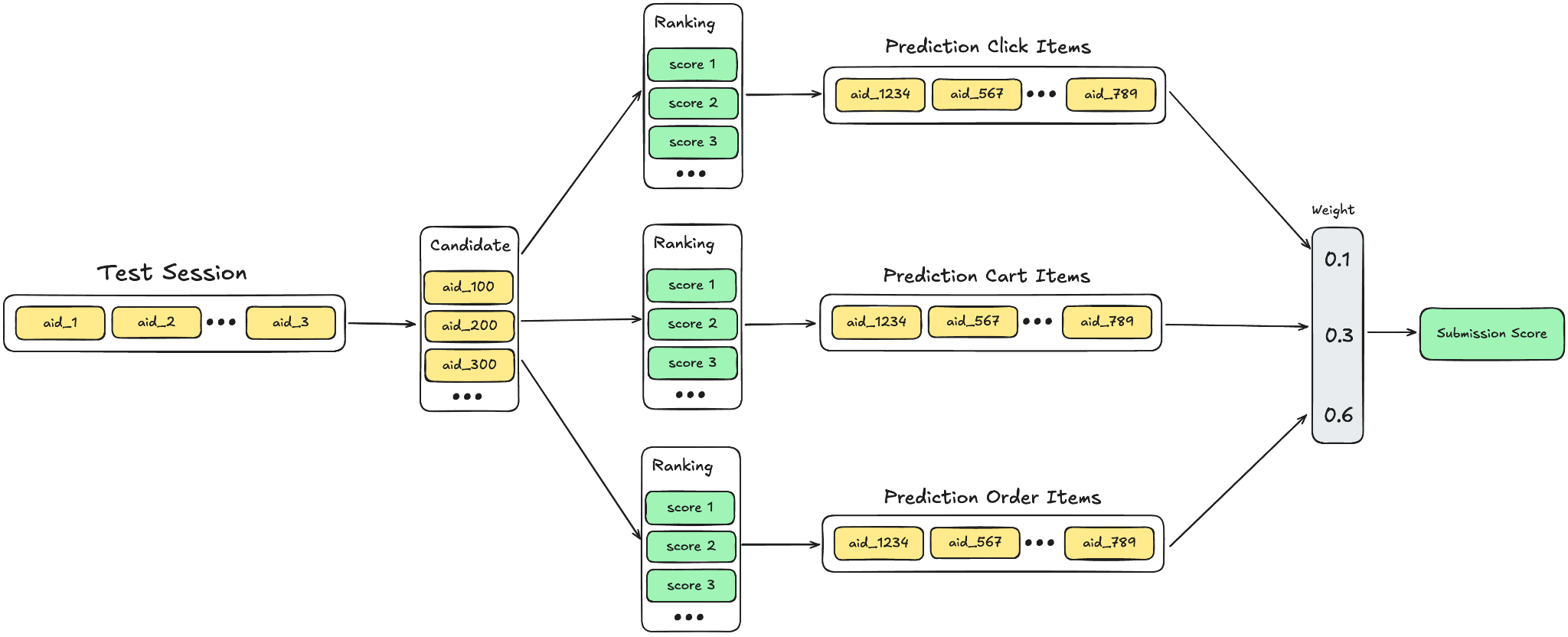
* **Clicks**: ~13% of the original dataset retained.
* **Carts**: ~3.5% of the original dataset retained.
* **Orders**: ~2.5% of the original dataset retained.



*Figure 3.7: Hyperparameter for XGBoost Model*

**Model Training**: Separate models will be trained for the **three targets**—clicks, carts, and orders—using **XGBoost** to optimize the ranking performance for each target.

### 3.1.4. Prediction



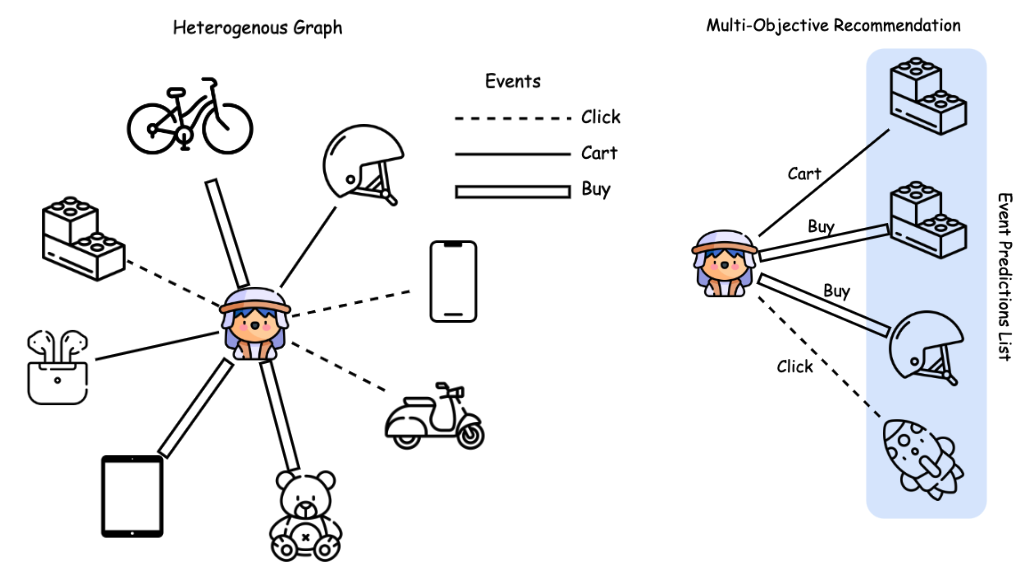
*Figure 3.8: Prediction Pipeline*

**Prediction:**

* A pool of candidate items is generated for each test session. These candidates are based on the methods in the training data.
* Each candidate item is ranked separately for the three targets—clicks, carts, and orders using the trained Xgboost model. The ranking process assigns a score to each candidate based on the likelihood of the target action occurring.
* The weighted scores are aggregated to produce a final ranked list of recommended items for each session. This final list maximizes the submission score by prioritizing the most relevant recommendations based on the combined target-specific predictions.

## **3.2. Graph-based Approach**

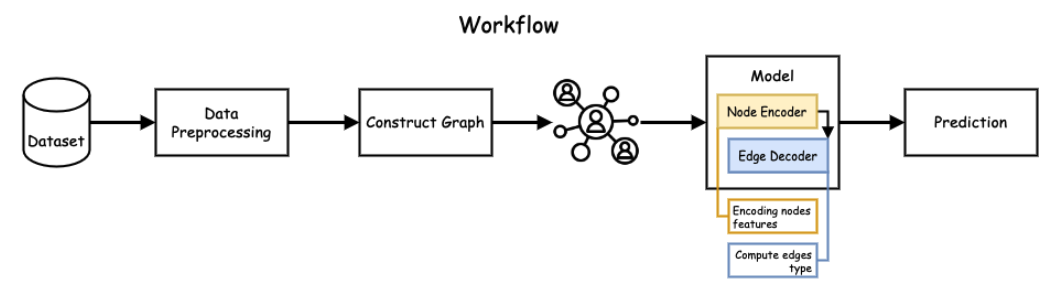
Another way of approaching the session based recommendation system problem beside the classical architecture is using Graph Neural Network, that is we formulate the problem into a link prediction problem. Heterogeneous graph (also called a **heterogeneous information network**) is a type of graph where nodes belong to different types and edges represent different relationships. This is particularly useful for our problem as sessions (or users) can click, add to cart or order an item that can be of many types (different aid). The objective is to build a multi-objective link prediction on a large-scale e-commerce heterogeneous graph. An illustration of this method is shown below.



*Figure 3.9: An illustration of the heterogeneous graph (left) which consists of multiple node and edge types and multi-objective recommendations (right) which takes prediction on multiple event types and items.*

First, we will preprocess the data into nodes and edges. For the nodes, we will have 2 types of nodes: the first represents the users (sessions) and the other represents the items. The latter will also encapsulate its embeddings (randomly initialized) along with its id. We connect these 2 nodes with the ‘event’ edge. The edges will contain their source and destination id along with its label (click, cart or order). The edges are undirected, allowing bidirectional flow of information during GNN computation.. Normally, building a graph will consist of positive sampling, constructed using the existing edges in the graph, where each edge represents a known interaction between a user (session) node and an item node, and negative sampling, generated by introducing edges between nodes that do not have a known interaction. These are assigned a negative label. For our project, due to large data size and hardware constraints, we do not do negative sampling.

We build a model that contains a node encoder and edge encoder. The node encoder learns feature representations for both user (session) and item nodes. It uses the node embeddings and aggregates information from neighboring nodes via graph convolutional layers (e.g., GCN, GAT or SAGE in our model). The edge encoder learns to encode the interaction type (click, cart, or order) as an additional feature influencing the relationship between connected nodes. The model is trained with weighted mean squared error. The overall workflow can be seen below:



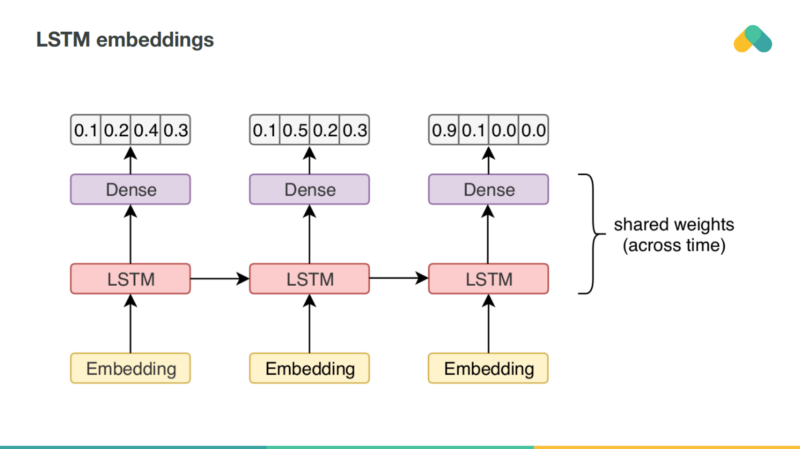
*Figure 3.10: An illustration of the overall workflow of the third method.*

## **3.3. RNN-based Approach**

During the development of the recommendation system, Recurrent Neural Networks (RNNs) were also explored as a potential solution for modeling sequential user interactions. An RNN model was implemented using PyTorch, featuring an LSTM architecture with embedding layers for products and actions. The setup included embedding dimensions of 16 for products and 6 for actions, two LSTM layers with 16 units each, and a dropout rate of 0.3 for regularization. The input consisted of sequences of product and action IDs, with separate embeddings concatenated before being fed into the LSTM layers. The model outputs two separate predictions: one for the next product and one for the next action in the sequence. The dataset comprised 1.8 million unique products (NUM\_PRODUCTS = 1\_855\_603) and three action types (NUM\_ACTIONS = 3), making the size of the embedding matrix and the number of model parameters substantial.

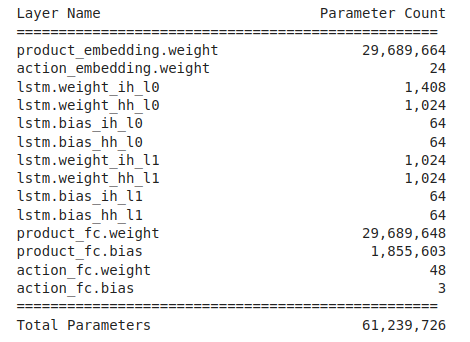
| **Hyperparameter** | **Value** |
| --- | --- |
| NUM\_PRODUCTS | 1,855,603 |
| NUM\_ACTIONS | 3 |
| PRODUCT\_EMBEDDING\_DIM | 16 |
| ACTION\_EMBEDDING\_DIM | 6 |
| RNN\_UNITS | 16 |
| NUM\_LSTM\_LAYERS | 2 |
| DROPOUT | 0.3 |

*Table 3.11: Hyperparameters setup used for an attempt to train an LSTM-based model*



*Figure 3.12: Prototype for an LSTM-based architecture for session-based recommender system*

Despite the theoretical feasibility of using RNNs for this task, training the model proved to be infeasible due to hardware limitations. Specifically, the enormous size of the product embedding matrix required a significant amount of RAM and GPU memory to store and process during training. This led to frequent out-of-memory errors, as the embedding layer alone demanded over 28 GB of memory, which exceeded the available resources. Sparse embeddings were considered in the implementation, but this optimization was insufficient to address the resource constraints given the scale of the dataset. As a result, the RNN approach had to be abandoned in favor of lighter and more scalable alternatives, such as Word2Vec embeddings combined with candidate retrieval and ranking models. This experience underscores the practical challenges of scaling deep learning models to extremely large item vocabularies.



*Figure 3.13: A summary of the number of parameters involved in the model architecture.*

From Table 4.1 and Figure 3.13, it can be seen that even a simplistic architecture has tens of millions of tunable parameters, the majority of which comes from the fact that the linear connection at the last step is burdening due to the large number of products. This fact renders the training process nearly impossible in practice, and complete impossible within the scope of the group’s computing resource availability.

One can argue that a number of actions, such as clustering articles beforehand, in order to reduce the number of products lying at the end layer of the architecture can help reduce the number of parameters needed. Yet, this approach can be considered as converging towards the Retrieval-Reranker approach mentioned in Section 3.1, which uses industry-standard re-ranker models instead of RNN-based ones already.

# **4. EVALUATION**

## **4.1. Metrics**

For the evaluation metrics, we will use Recall@20 provided by the competition on each action type. The three recall values are weight-averaged:

where is defined as

and is the total number of sessions in the test set, and **predicted aids** are the predictions for each session-type (e.g., each row in the submission file) truncated after the first 20 predictions.



*Figure 4.1: Example of the result submission*

For each session in the test data, the task is to predict the **aid** values for each type that occur after the **last** timestamp is the test session. In other words, the test data contains sessions truncated by timestamp, and weare to predict what occurs after the point of truncation.

For **clicks** there is only a **single ground truth** value for each session, which is the next aid clicked during the session. The **ground truth** for **carts** and **orders** contains **all aid values** that were added to a cart and ordered respectively during the session.

## **4.2. Results**

| **Method** | **Recall@20** |
| --- | --- |
| Covisitation Matrix (Our) | 0.161 |
| GNN (Our) | 0.486 |
| Word2Vec (Kaggle) | 0.533 |
| Co-visitation + LGBM Ranker (Kaggle) | 0.551 |
| Co-visitation + Word2Vec + Factorization + XGBoost (Our) | 0.573 |

*\* Some results are from kaggle public implementation results.*

From the table, we can see that the initial model started with a covisitation matrix, which was quite simplistic, while later models used more complex techniques to model session-level relationships and product-level relationships. The GNN model made a massive advancement in the use of graph structures to represent user-product interactions, but was still unable to beat the simpler Word2Vec approaches because of the lack of scalability.

Another step up came with Word2Vec models from Kaggle, which greatly emphasized the effectiveness of sequence models in encoding product relationships. Still, our last approach which combined Covisitation, Word2Vec embeddings, Factorization, and XGBoost for ranking did better than the rest. In this multi-stage pipeline, the weaknesses of one method were strengthened by the strengths of each of the other two methods through powerful candidate generation and effective ranking. The steady increase of Recall@20 values also substantiates the need for combining various strategies and more importantly shows how performance can be increased without compromising on computational times through layering of techniques. This trend also goes along the trend of enhancing the quality of the candidate generation and ranking for better robustness.

# 

# **5. DISCUSSION**

## **5.1 Key Insights**

The project on session-based recommender systems for retail products has provided valuable insights into the design and implementation of recommendation algorithms tailored for dynamic and large-scale e-commerce datasets. Through the exploration of various methods—including traditional retrieval and ranking pipelines, feature engineering, RNN-based attempts and graph-based neural network approaches—we have observations as follows.

One key realization was the necessity of moving beyond resource-intensive methods, such as RNNs, to scalable retrieval-reranker pipelines. By reducing the number of candidates at the retrieval stage, computational efficiency improved without sacrificing prediction quality. Moreover, temporal relationships emerged as a critical component in feature engineering, with timestamp-based features and interaction frequency metrics enhancing the understanding of user behavior and session dynamics. Overall, the inclusion of session-level, item-level, and temporal features significantly improved the predictive performance of ranking models

Another key insight was the importance of interpreting user intentions behind actions, motivating the weighting of different interaction types (e.g., clicks, carts, and orders) to align with their varying significance in real-world scenarios. The multi-stage pipeline approach, combining candidate generation, feature extraction, and ranking, demonstrated its effectiveness in balancing computational efficiency and prediction accuracy. Lastly, achieving computational efficiencies while handling 1.8 million products required optimizations such as heuristic-driven retrieval methods and streamlined feature engineering processes, ensuring scalability for real-world applications. These insights highlight the importance of prioritizing scalability, interpretability, and efficiency in session-based recommender systems.

## **5.2 Limitations**

Despite the successes achieved in building the recommendation system, several limitations were identified, highlighting areas for improvement.

First, scalability, while somewhat handled in a retrieval-reranker setting, posed a significant challenge, particularly in graph-based approaches, where computational bottlenecks hindered their application to the full dataset of 1.8 million products. Secondly, we observe the dominance of clicks over carts and orders, creating an interaction imbalance, complicating the analysis process needed in extracting relationships between items for the sake of co-visitation matrices or Word2Vec embeddings.

Furthermore, the anonymized nature of the data, lacking descriptive attributes like product categories, names, or images, limited the ability to incorporate content-based features that could enrich the models. Memory constraints and the large scale of the dataset also restricted experiments with resource-intensive methods like RNNs and GNNs, as these approaches exceeded GPU and RAM capacity during training.

Additionally, hyperparameter tuning, particularly for ranking models such as LightGBM and CatBoost, was constrained by time, leaving potentially impactful optimizations unexplored. Lastly, the retrieval and candidate generation phases, while effective, were reliant on heuristic-driven co-visitation counts and Word2Vec embeddings, which may miss nuanced contextual signals that more advanced architectures could capture. These limitations underline the need for future work to enhance scalability, data enrichment, and optimization techniques.

## **5.3 Future Directions**

Building upon the insights gained during this project, several promising avenues for future research and development are identified to overcome current limitations and enhance the system's capabilities.

One key direction involves exploring advanced graph-based techniques, such as scalable Graph Neural Network (GNN) architectures like GraphSAGE or Graph Attention Networks (GAT), combined with mini-batch sampling strategies to manage the scale of 1.8 million products efficiently. Additionally, integrating temporal graph features could improve the model's ability to capture dynamic user-product interactions over time.

Another priority is to augment the dataset with further avenues of extracting product relationships, to enable richer feature engineering for candidate generation and retrieval. Re-examining the feasibility of RNNs, possibly using techniques like mixed precision training or model parallelism to address memory constraints, could also offer new insights for sequence modeling.

Moreover, systematic hyperparameter optimization, leveraging tools such as Optuna or Ray Tune, could unlock the full potential of ranking models. Finally, investigating novel approaches, such as hierarchical candidate generation pipelines or pre-trained language models tailored for recommendation, could further improve the accuracy and scalability of the system. These directions aim to address current challenges while paving the way for a more robust and adaptable recommendation system.

# 

# **6. CONCLUSION**

The development of session-based recommender systems for retail products represents a critical step in enhancing user experience and driving revenue in the e-commerce industry. This project has demonstrated the effectiveness of combining traditional recommendation pipelines with advanced graph-based techniques to address challenges such as data sparsity, scalability, and diverse user behavior.

Key achievements include successfully implementing multi-step pipelines, leveraging Graph Neural Networks for complex interaction modeling, and incorporating comprehensive feature engineering to enhance ranking accuracy. Evaluation metrics such as Recall@20 highlighted the practical relevance of the models, particularly in predicting user actions for clicks, carts, and orders.

However, challenges such as computational scalability, imbalanced interaction types, and the limitations of anonymized datasets emphasize the need for continued innovation. Future work should focus on integrating richer features, optimizing model parameters, and exploring hybrid approaches that combine the strengths of various methodologies.

In conclusion, the insights and methodologies developed through this project provide a strong foundation for advancing session-based recommendation systems. These systems have the potential to transform online retail experiences by delivering personalized, timely, and relevant recommendations that meet both user expectations and business objectives.

# 

# **7. CONTRIBUTION**

Our work are at: https://github.com/datvodinh/otto-recommendation-system.git

| **Member** | **Task** | **Workload** |
| --- | --- | --- |
| Võ Đình Đạt | * Basic Co-visitation matrix generation. * Generate candidates based on all the given models, embeddings and matrices. * Implement the XGBoost Ranking model. | 20% |
| Đoàn Ngọc Cường | * Converting the dataset to a more optimal format (Parquet). * Implement Matrix Factorization model and training code for the training session. * Implement prediction pipeline. | 20% |
| Lê Trung Kiên | * Implement Graph Neural Network. * Start combining predictions from different models using ensembling methods. * Validate and hyperparameter tuning for all methods. | 20% |
| Phạm Quang Trung | * Clean and preprocess the dataset (handle missing values, data transformation). * Weighted Co-visitation matrix generation. * Feature engineering (session-level features, interaction types, time-based features). | 20% |
| Đoàn Thế Vinh | * Explore the OTTO dataset (understand data structure, distribution, event types). * Implement the Word2Vec model and training code for the training session. * Explore stacking or blending techniques for ensembling the models. | 20% |

# **8. REFERENCES**

1. Hamilton, W. L., Ying, Z., & Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. *Advances in Neural Information Processing Systems*, 30.
2. Kaggle. (2024). OTTO Recommender Systems Challenge. [Online]. Available: https://www.kaggle.com/competitions/otto-recommender-system
3. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8), 30-37.
4. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.
5. Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. *J Big Data* 9, 59 (2022).