# Leveraging NLP to Build Smarter Recommender Systems

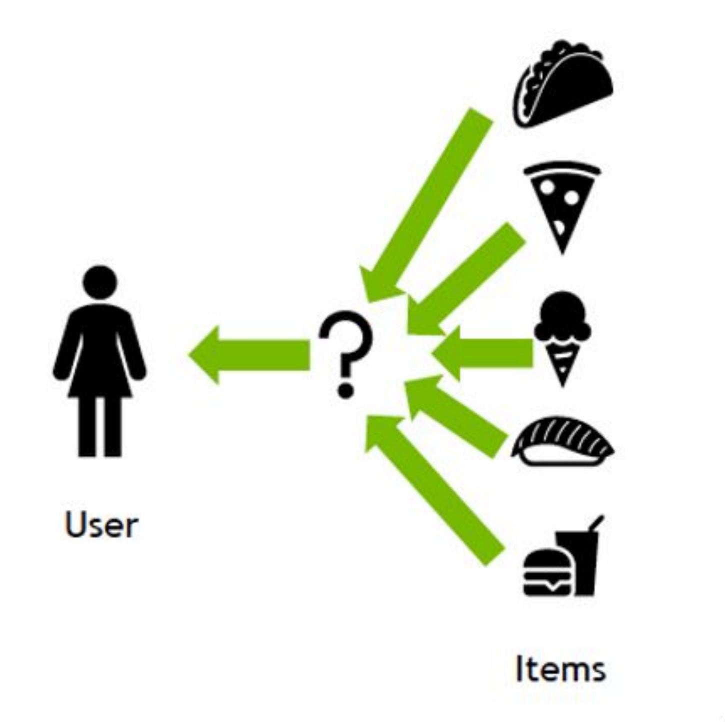
Sook hyun Lee, PhD

Data Science Career Track Capstone Project, Oct 2025

Springboard

# Problem statement

Modern *users* overwhelmed by the vast amount of available *content, products,* and *choices* across digital platforms.




- ➢ How can we effectively handle ***new users*** and ***items*** in the system?
- ➢ Can we teach the recommender to understand ***content***, not just behavior?
- ➢ What strategies can improve the learning ***efficiency*** of the recommender engine?

# Who Finds This Relevant — And Why It Matters?

E-commerce Platforms

Streaming Services

Social Media

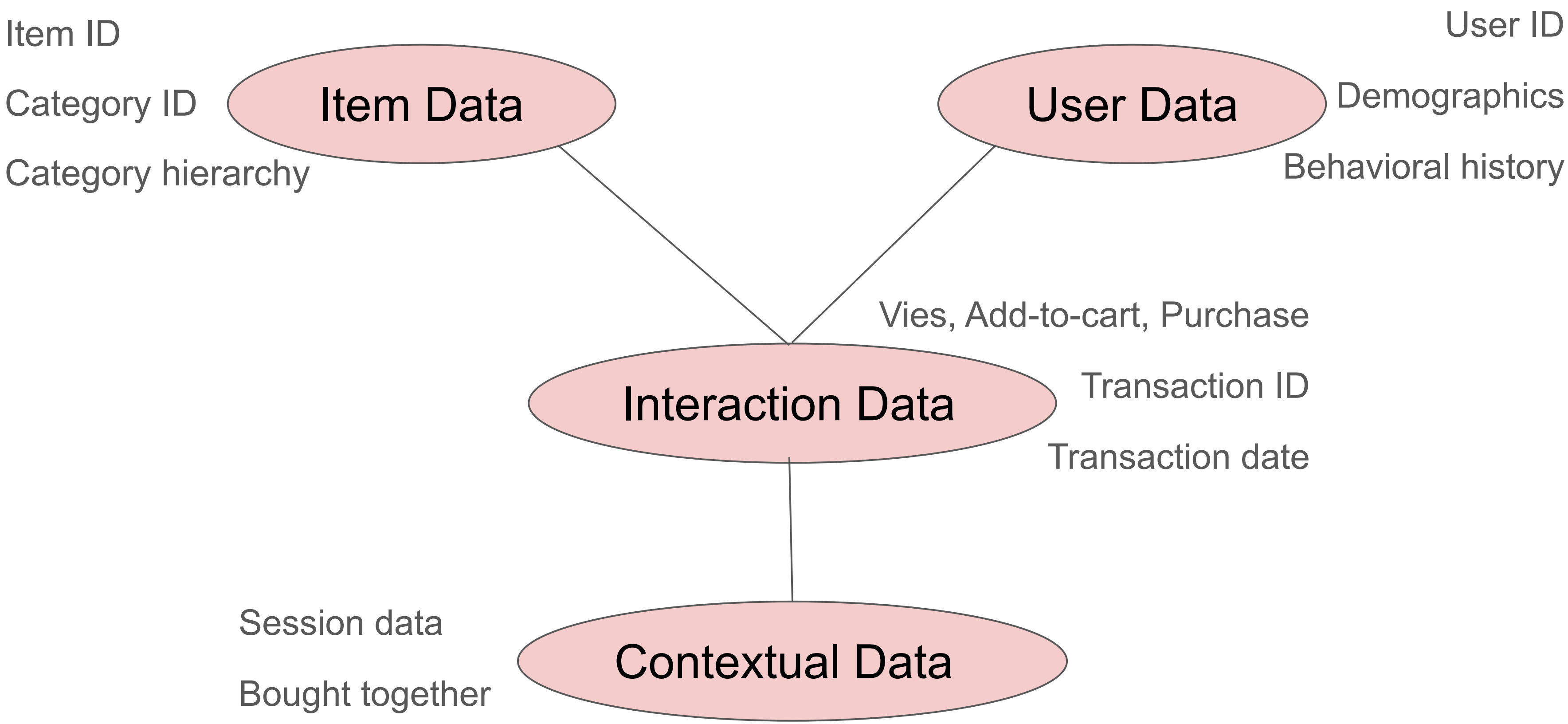
Online Education

**Marketing** — Personalized campaigns, segmented targeting

**Product** — UX improvements, product bundling

**Leadership** — Strategic investment, growth metrics

# Which Data Inputs Are Essential for Personalization?



# Data Source and Summary

Data source: Kaggle
[RetailRocket Recommender System Dataset]

`events.csv`
`item_properties_part1.csv`
`Item_properties_part2.csv`
`category_tree.csv`

## Data acquired for the period:

2015-05-10 to 2015-09-13

## Records contain:

417053 unique items w/ category ID (item properties data)
1670 unique *category IDs* (category data)

## Number of fields:

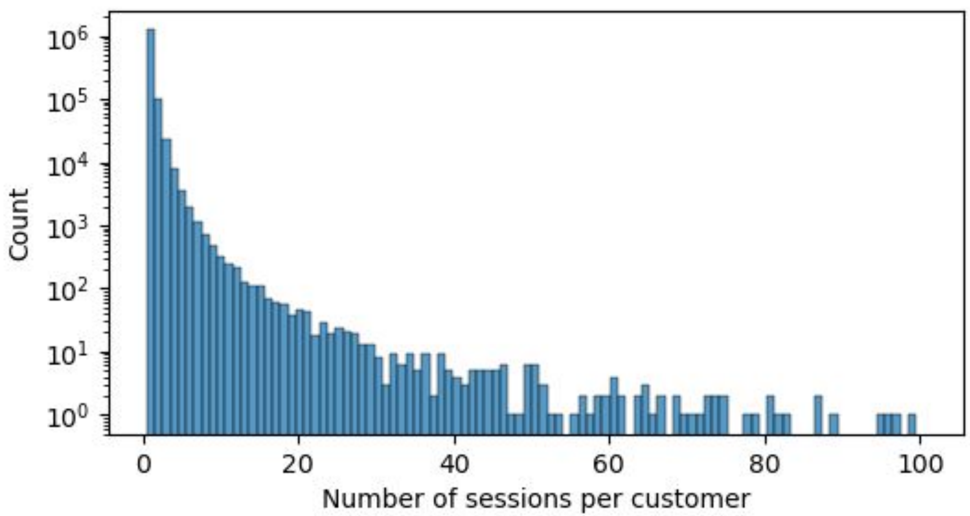
5 (event data), 4 (item properties data),
2 (category data)

# Data Exploration - Event Data

**FIELDS**
Time stamp,
Visitor ID,
Event type,
Item ID,
Transaction ID

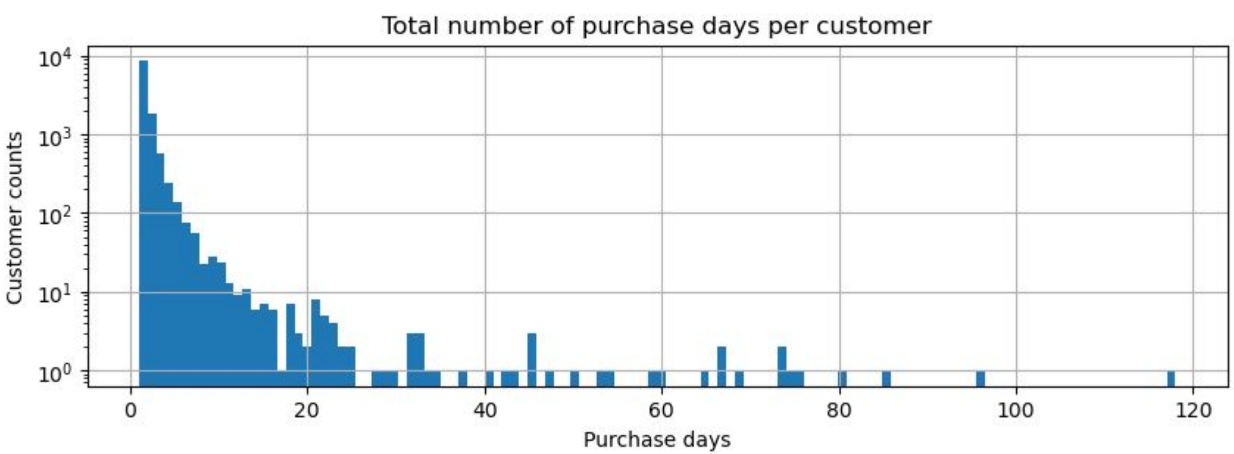
Table 1. An example sequence of activities by a customer (ID 90352)

| | timestamp | visitorid | event | itemid | transactionid |
|---|---|---|---|---|---|
| **265573** | 1434403666570 | 90352 | view | 425758 | NaN |
| **271793** | 1434404197081 | 90352 | transaction | 425758 | 0.0 |
| **277295** | 1434403991902 | 90352 | addtocart | 425758 | NaN |
| **533488** | 1435331816929 | 90352 | view | 425758 | NaN |

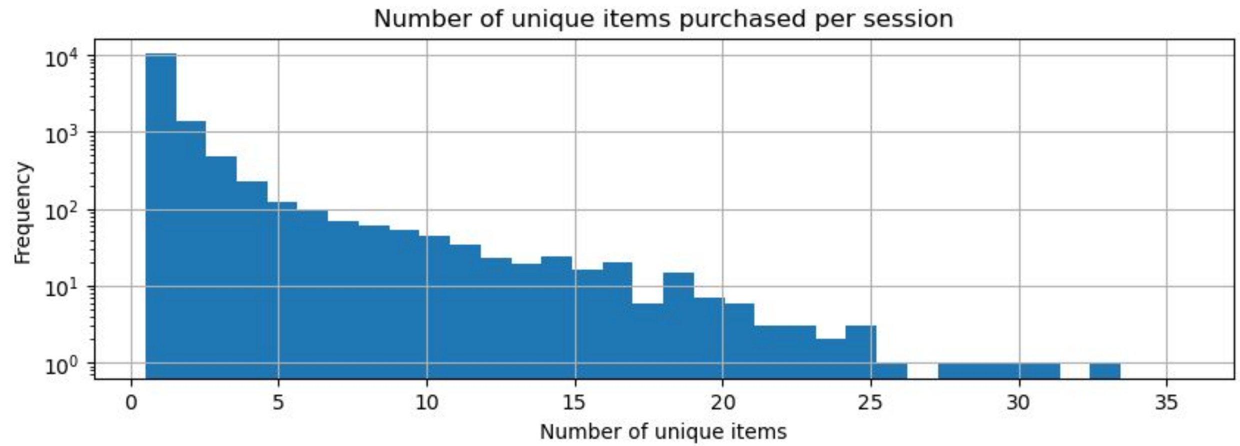


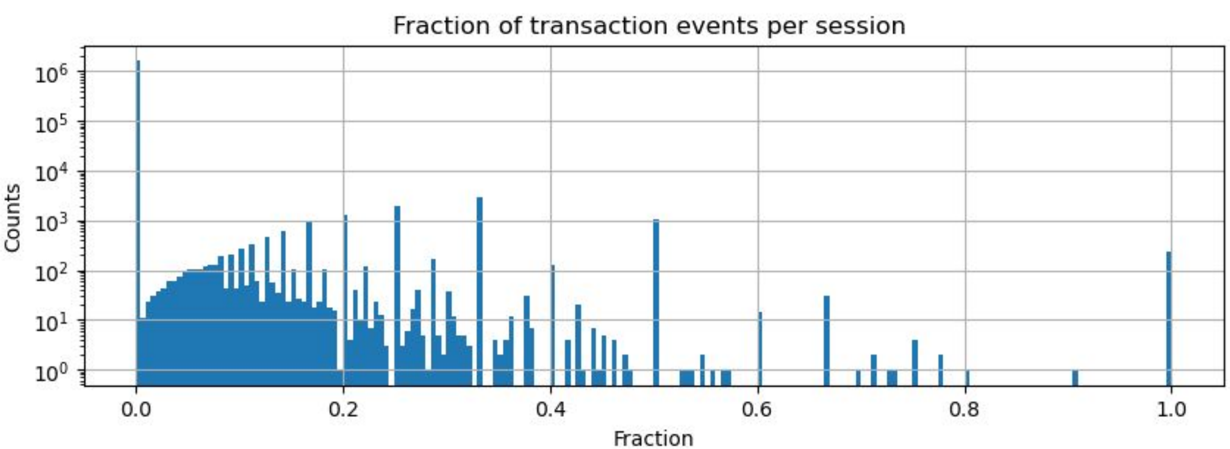

**SESSION:**
Activities occurring on the same day for each customer were grouped together, and a new column was created to enumerate these daily sessions.

# Customer Behavior

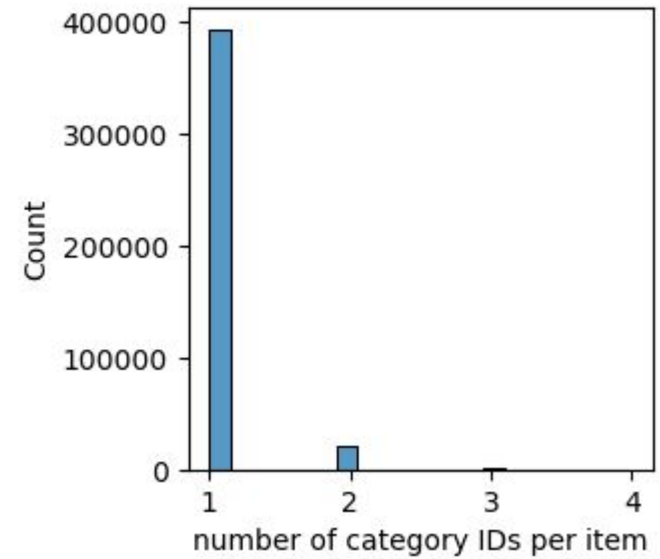
# Customer Behavior



# Item Properties Data

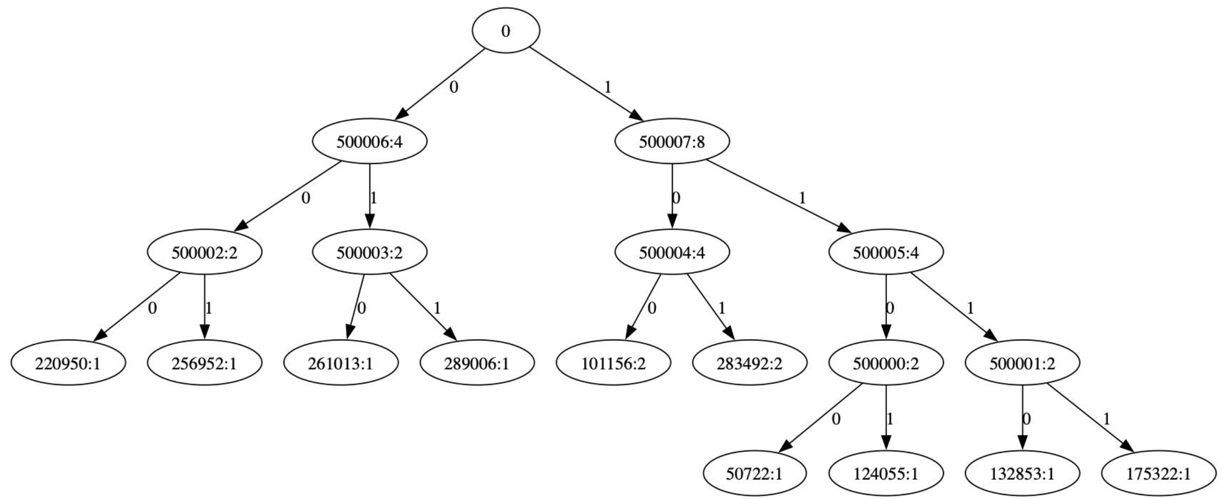
- ❑ **5.6%** of items linked to more than one *category ID* - **Dropped**
- ❑ Merge unique *item ID–category ID* mapping with event data.
- ❑ **21.2%** of all unique items in event data lack associated category ID - **Dropped**.
- ❑ Only **3.16%** of unique items involved in **transactions** are missing category ID - Only **Kept** items in **transactions** with **unique category ID**.




| | timestamp | itemid | property | value |
|---|---|---|---|---|
| **0** | 1435460400000 | 460429 | categoryid | 1338 |
| **1** | 1441508400000 | 206783 | 888 | 1116713 960601 n277.200 |
| **2** | 1439089200000 | 395014 | 400 | n552.000 639502 n720.000 424566 |
| **3** | 1431226800000 | 59481 | 790 | n15360.000 |
| **4** | 1431831600000 | 156781 | 917 | 828513 |

Table 2. Contents of item properties data in `item_properties_part1.csv.`

# Building Huffman Tree for Hierarchical Softmax




## Huffman tree

- is a **binary tree** over items that
- is used for efficient probability estimation in **hierarchical softmax** training technique.
- speeds up training and allows for a scalable output layer.

Node representation:
(Node ID: Frequency)

# Category Data and Building Category Tree

6 levels of categories - items may be attached at any level.

| Level | L0 | L1 | L2 | L3 | L4 | L5 | L6 |
|---|---|---|---|---|---|---|---|
| No. of categories | 1 | 25 | 174 | 702 | 665 | 90 | 13 |


```
item ID <------> root
[132853, 0, 605, 1482, 10000]
```

Figure 10. Full path from item 132853 to the root node 10000.

# Machine Learning Modeling Overview

- Two primary approaches to recommender systems
  - Collaborative filtering
    - User-based filtering
    - Item-based filtering
  - Content-based filtering

| Collaborative Filtering | Content-based Filtering |
|---|---|
| Data sparsity<br>Cold start<br>Scalability<br>Popularity bias<br>Lack of Interpretability<br>Gray sheep problem | No need for other users' data<br>Handles cold start (user side)<br>Interpretable recommendations<br>Less prone to popularity bias<br>Privacy-friendly |

# Deep Learning Based Approaches

- Strengths
  - Better at capturing non-linear and complex patterns
  - Better at cold start and sparse data
  - Effective feature representation (Embeddings)
  - Personalized and context-aware recommendations
- Item2Vec and hierarchical Item2Vec models
  - Baseline Item2Vec: Inspired by Word2Vec technique in NLP.
    - Item-based collaborative filtering approach.
  - Hierarchical Item2Vec: Context-aware extension of Item2Vec.
    - Implement in *Pytorch* with **DNN** architecture to capture similarity and co-occurrence patterns of items and their hierarchical content-based info.

# Modeling Steps

**Training Data Preparation**

Split Training and Validation Data

Model building

Identify Hyperparameters

Hyperparameter Tuning

Performance Evaluation

Select sessions with two or more items per transaction.

**7547** unique items

**3,055** distinct sessions

**Inputs**

**(item1, item 2)**

⋮

# Identifying Hyperparameters

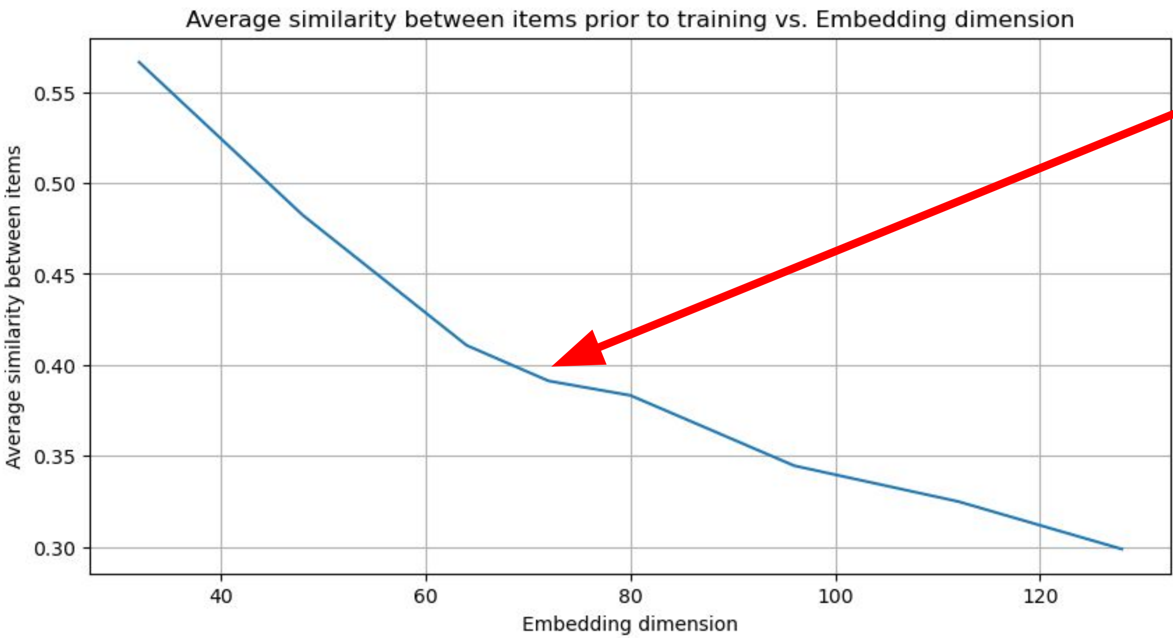
- Regularization Parameter $\lambda_{cat}$
- Embedding Dimension $\mathrm{dim}_{\mathrm{embed}}$
- Threshold for Item Frequency $f_{\mathrm{thresh}}$ = 1
- Learning Rate and Batch Size
- Loss Function = {Negative sampling, Hierarchical softmax}
- Similarity Measure: Cosine similarity, Distance

# Hyperparameter Tuning

- Embedding Dimension

$$\mathrm{dim}_{\mathrm{embed}} = \{32, \ldots, 128\}$$

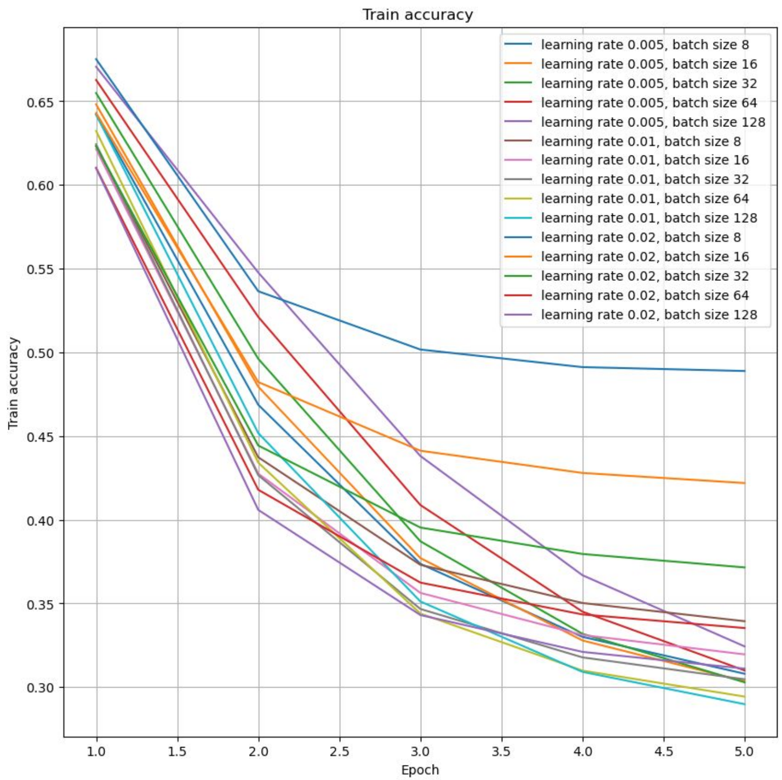
Select the embedding dimension at which the average similarity between the **five most frequent items** and their **five nearest neighbors** first drops below **0.4.**

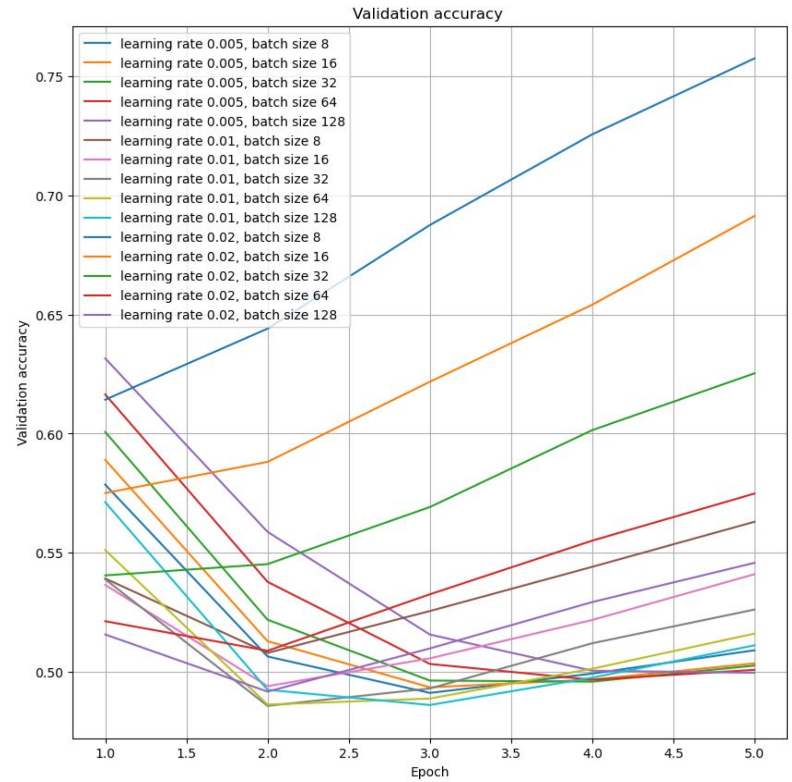


- # Learning Rate and Batch Size

Optimal learning rate = **0.01**,
Batch size = **64**

Optimize the batch size and learning rate while keeping the regularization parameter **$\lambda_{cat}$** set to **zero**.



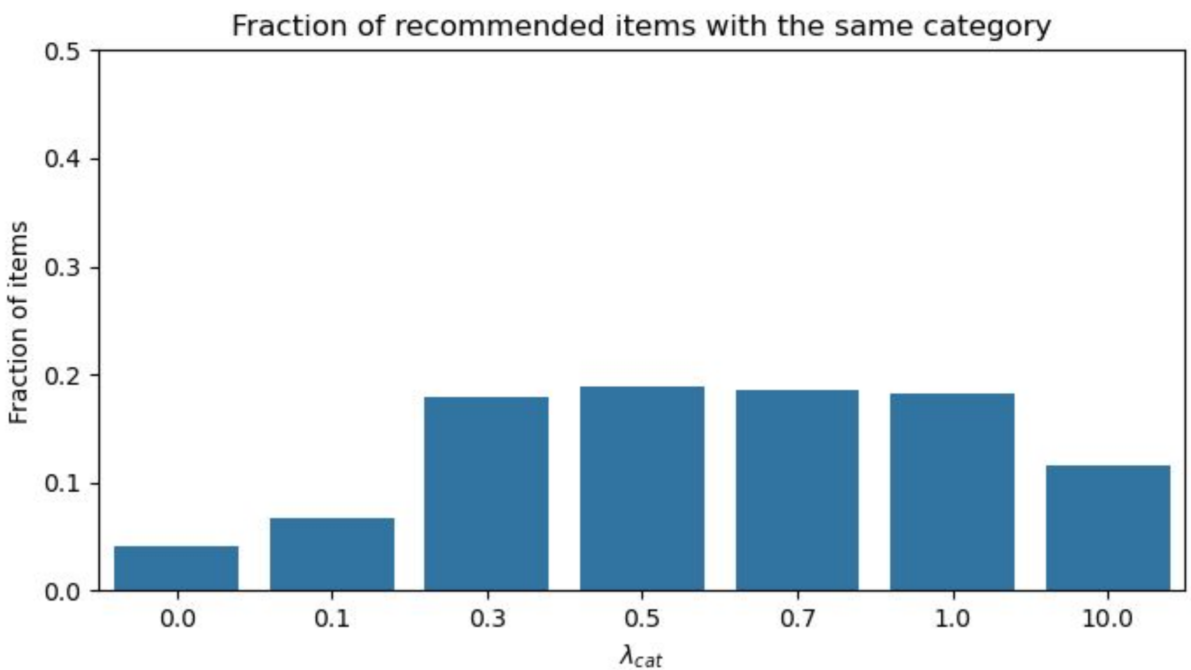



- Regularization Parameter

$\lambda_{cat}$ = {0, 0.1, ... , 1, 10}

- Shared-category fraction grows with regularization strength, up to a point.

- High parameter values (e.g., 10) suppress meaningful user–item interaction patterns.



# Evaluation Metrics

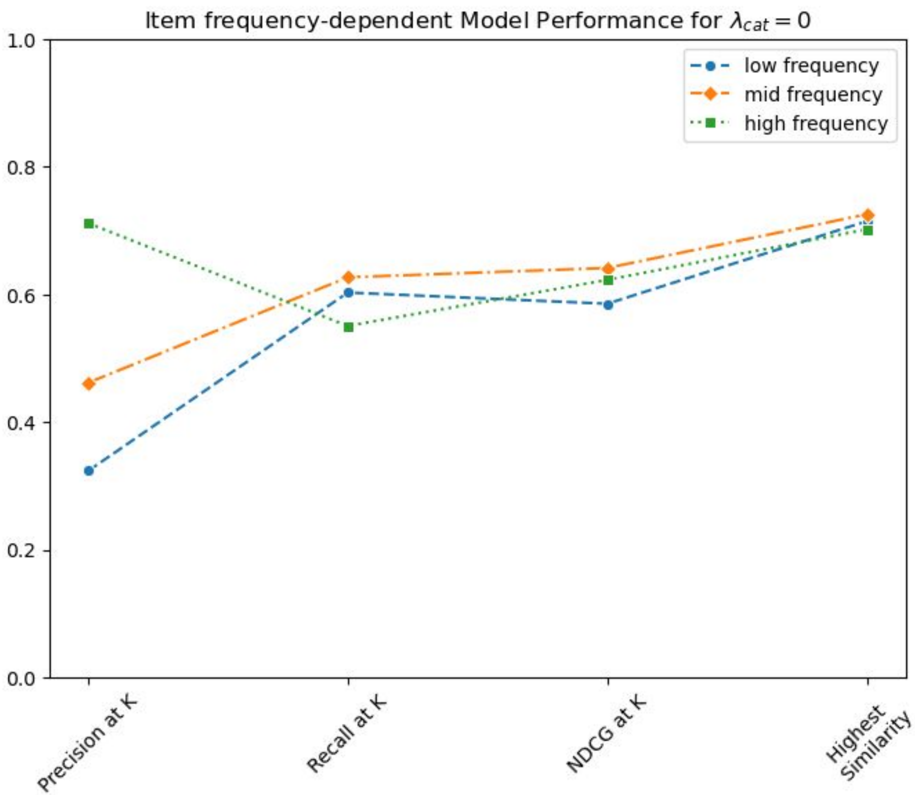
- ☐ Precision at K = $\dfrac{Number\ of\ relevant\ items\ in\ top\ K}{K}$

- ☐ Recall at K $=\dfrac{Number\ of\ relevant\ items\ in\ top\ K}{Total\ number\ of\ relevant\ items}$

- ☐ NDCG at K = $\dfrac{Discounted\ Cumulated\ Gain\ (DCG)\ at\ K}{Ideal\ Discounted\ Cumulated\ Gain}$

  , where DCG at K = $\displaystyle\sum_{i=1}^{K} \frac{Number\ of\ relevant\ items\ in\ top\ K}{log_2(i+1)}$

# Performance By Frequency Group




Baseline Item2Vec

| Frequency | Group | Counts |
|---|---|---|
| **0** | Cold items | |
| **1** | Low frequency | 4025 |
| **2-4** | Moderate frequency | 2884 |
| **5+** | High frequency | 638 |

**Recall@K** and **Precision@K**

: Fixed value of K imposes artificial constraints on metrics for high/low frequency items.

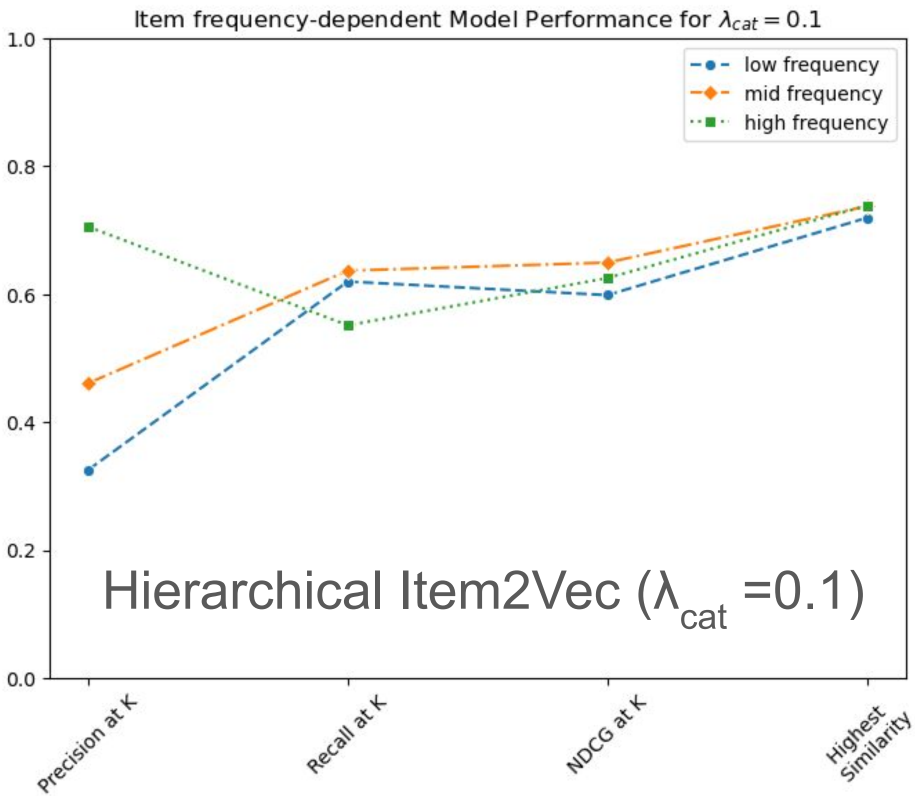
**NDCG@K**

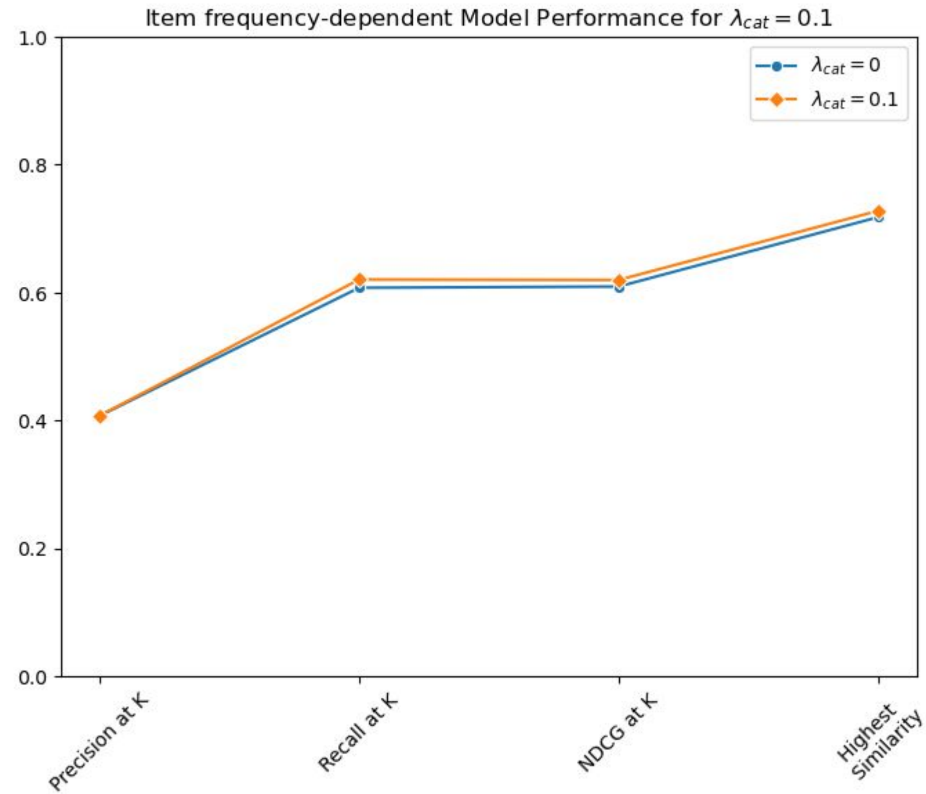
: More robust metric across different item frequency groups

**Cosine Similarity**

: Shows high-quality and semantically meaningful representations for items.

# Model Performance Comparisons




Enables introducing new items via category embeddings or averaging over items in the same category.

Hierarchical Item2Vec slightly outperforms Item2Vec.

# Summary

- Built a recommender system using **Item2Vec** and **Hierarchical Item2Vec** to learn item embeddings from transaction data.
- **Goal**: Improve recommendation quality by capturing item relationships more effectively.
- **Evaluation**: Precision@K, Recall@K, NDCG@K, similarity scores - across item frequency groups.
- **Finding**: NDCG@K better reflects ranking performance than precision/recall for rare/frequent items.
- **Results**:
  - Hierarchical model slightly outperforms baseline and opens up ways to introduce new items.
  - Both models produce high-quality embeddings.
- **Conclusion**: Hierarchical approach improves semantic alignment and recommendation accuracy.

# Outlook & Future Work

- **Incorporating temporal dynamics & user context**
  Explore time-aware models and contextual signals to refine recommendation relevance over time.

- **Addressing rare Items & cold-start scenarios**
  Investigate strategies to improve recommendation coverage and robustness for infrequent or new users.

- **Adaptive evaluation metrics**
  Experiment with metrics that account for variable item relevance to enable more nuanced performance assessment.