ML System Design Similar Listings on Vacation Rental Platform

Video ID: https://www.youtube.com/watch?v=2fF6aQGLQQc

Business Objective:

Similar Listings on Vacation Rental Platforms

Clarifying the Requirements

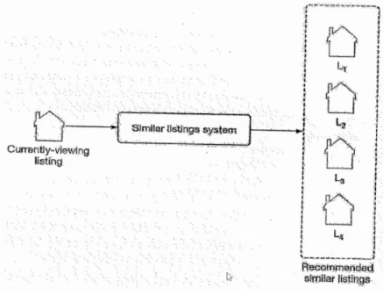
Clarifying the Requirements:

- To design a vacation rental's "similar listings" feature where:
  - Business Objective (B.O.) is to increase the number of bookings.
- Assumptions and constraints:
  - When we say "similar listings" meaning, they are in same neighbourhood, city, price range etc.
  - The model will only utilize user's browsing history during training.
  - Listings for logged in users and anonymous users will be same.
  - There are 5 million listings on the platform where new listings will appear one day after.
  - For training dataset of the model we only have user-listing interaction only.

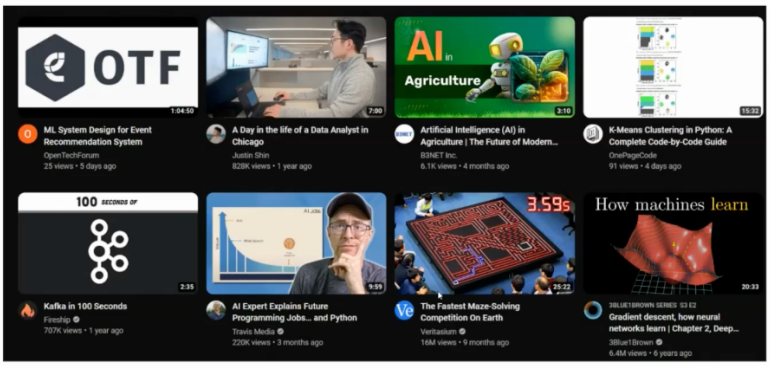
Framing problem as ML task

Framing the Problem as an ML task: Defining the ML objective:

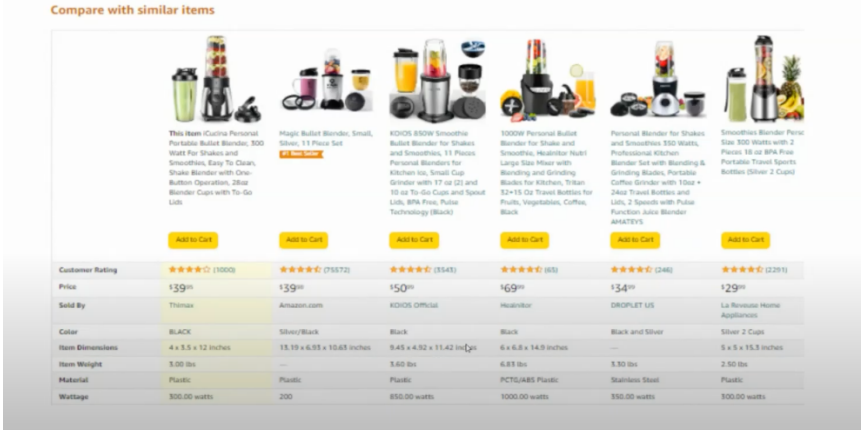
- User clicks have similar characteristics.
- ML objective - to accurately predict which listing the user will click next given the listing the user is currently viewing.



Choosing the right ML category:
Traditional recommendation system vs Session based Recommendations:

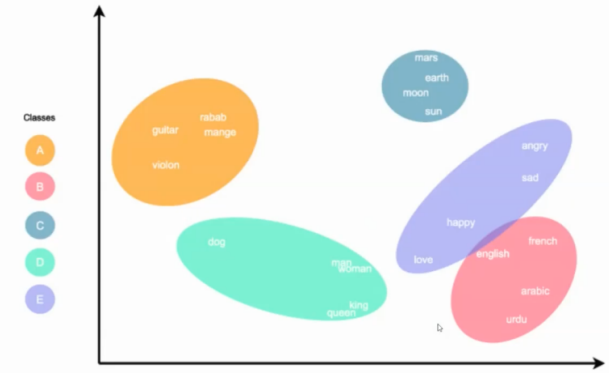


Session based recommendations:



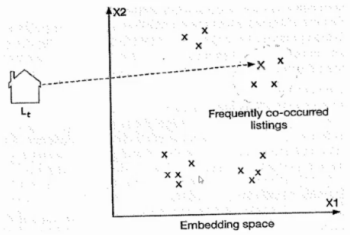
User browsing session:

2D Embedding Space:



Choosing Best ML category: Session based Recommendations

- A widely used technique to build a session based recommendation system is to learn the item embeddings and their co-occurrences in user's browsing histories.
  - Instagram - account embeddings for "Explore" feature.
  - Airbnb - listing embeddings for "similar listing" feature.
  - Word2vec - meaningful word embeddings.



Data

Data preparation : Data Engineering

- User
- Listing
- User - Listing Interaction

- User:

ID	Username	Age	Gender	City	Country	Language	Timezone

- Listing Data: 1.

ID	Host ID	Price	Sq ft	Rate	Type	City	Beds	Max guests
1	135	135	1060	4.97	Entire place	NYC	3	4
2	81	80	830	4.6	Private room	SF	1	2
3	64	65	2540	5.0	Shared room	Boston	4	6

2.

Season	Nightly Rate	Min Nights
Low Season	$ 135.00	4
Mid Season	$ 150.00	4
Peak Season	$ 200.00	5
Holiday Season	$ 250.00	6

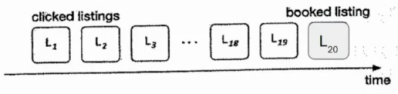
Season	Start	End	Nightly Rate	Min Nights
Low Season 2017	9/3/17	9/30/17	$ 135.00	4
Low Season 2017	10/1/17	10/31/17	$ 135.00	4
Low Season 2017	11/1/17	11/18/17	$ 135.00	4
Mid Season 2017	11/19/17	11/30/17	$ 150.00	4
Low Season 2017	12/1/17	12/18/17	$ 135.00	4
Holiday Season 2017	12/19/17	1/4/18	$ 250.00	6

User-Listing Interaction Data:

ID	User ID	Listing ID	Position of the listing in the displayed list	Interaction type	Source	Timestamp
2	18	26	2	Click	Search feature	1655121925
3	5	8	5	Book	Similar listing feature	1655135257

Data Preparation : Feature Engineering

- Using user's search history. Browsing sessions are "search sessions".
  - Search session - sequence of clicked listing IDs eventually getting booked, without interruption.



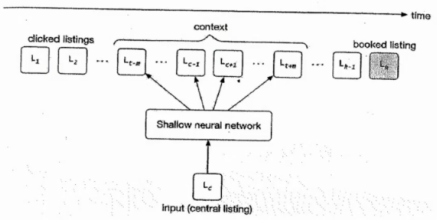
- Extracting Search Sessions from interaction data:

Session ID	Clicked listing IDs	Eventually booked listing ID
1	1, 5, 4, 9	26
2	6, 8, 9, 21, 6, 13, 6	5
3	5, 9	11

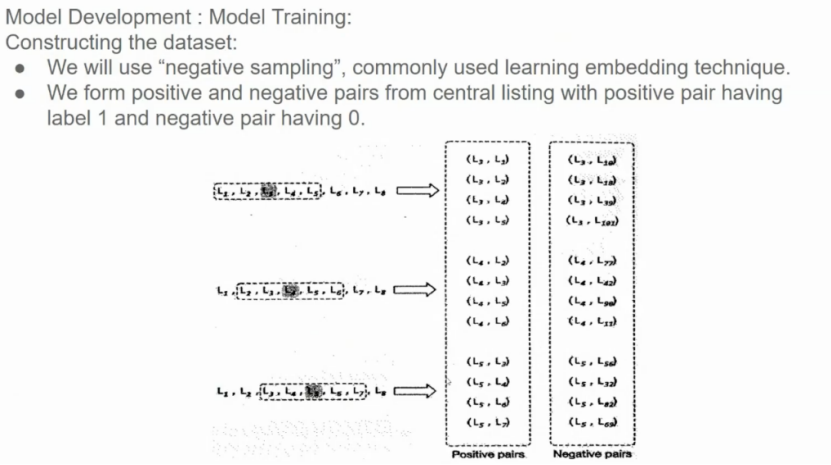
Model Development: Model Selection

- We will use a neural network.
- Common things considered while choosing a neural network architecture:
  - Complexity of tasks, amount of training data etc.
- We will choose a shallow neural network architecture for learning embeddings.

Model Training:



- For training the model, initializing to listing embeddings to random vectors eventually learning through search sessions the embeddings using sliding window method.
- We train the model daily to adapt to new data.



we want to create a system that can accurately recommend similar listings to users. To achieve this, we need a way to represent each listing as a numerical vector, or embedding, that captures its essential characteristics.

The Solution: Negative Sampling

Negative sampling is a technique used to efficiently train models on large datasets. In the context of Airbnb listings, it involves creating pairs of similar and dissimilar listings.

How it works in the image:

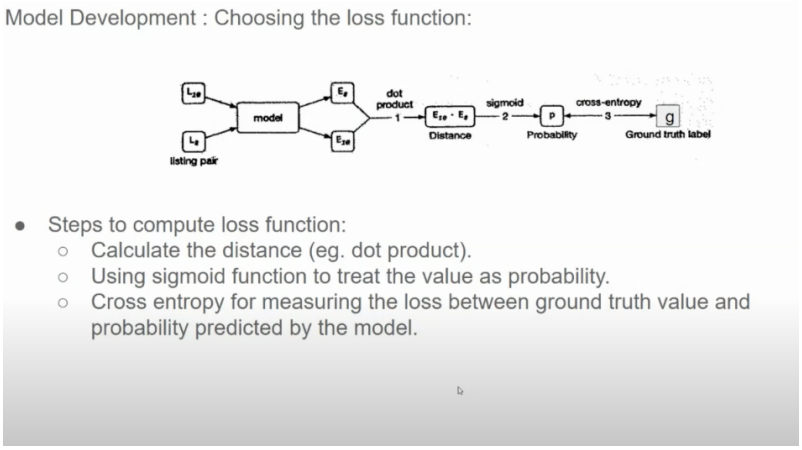
1. **Positive Pairs:**
    - A central listing is chosen.
    - Similar listings are selected as positive pairs. These are listings that share similar features like location, price, amenities, etc.
2. **Negative Pairs:**
    - Dissimilar listings are selected as negative pairs. These are listings that are significantly different from the central listing.

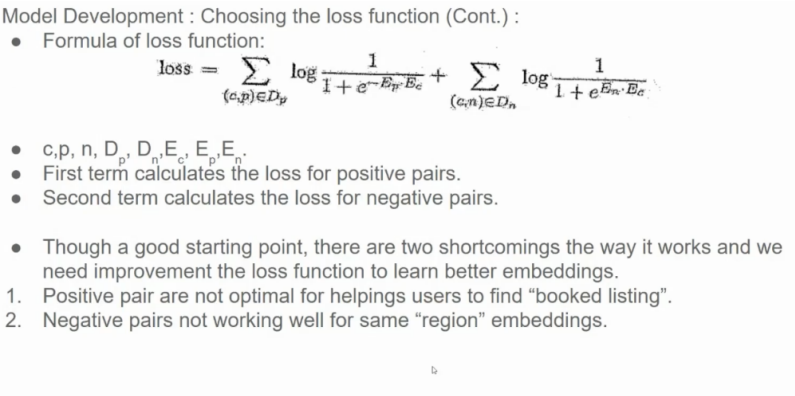
Training the Model:

- The model is trained on these pairs of positive and negative examples.
- The goal is to learn embeddings that maximize the similarity between positive pairs and minimize the similarity between negative pairs.
- This training process helps the model capture the underlying semantic and visual features of the listings.

Benefits of this Approach:

- **Improved Recommendation Accuracy:** By learning meaningful representations, the model can recommend more relevant listings to users.
- **Efficient Training:** Negative sampling reduces the computational cost of training.
- **Robustness:** The model becomes more robust to variations in listing descriptions and images.





The Proposed Loss Function:

The loss function presented in the image is designed to optimize the model's ability to differentiate between positive and negative pairs. It consists of two terms:

1. **Positive Pair Loss:**
    - Encourages the model to bring embeddings of similar listings closer together.
    - The loss is minimized when the distance between the embeddings of a positive pair (p, p') is small.
2. **Negative Pair Loss:**

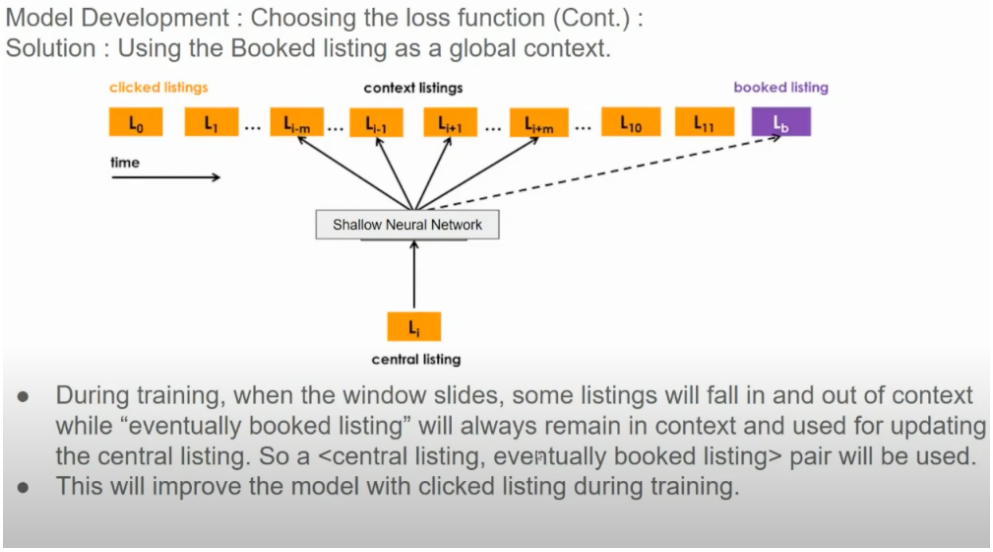
- Encourages the model to push embeddings of dissimilar listings further apart.
- The loss is minimized when the distance between the embeddings of a negative pair (c, n) is large.

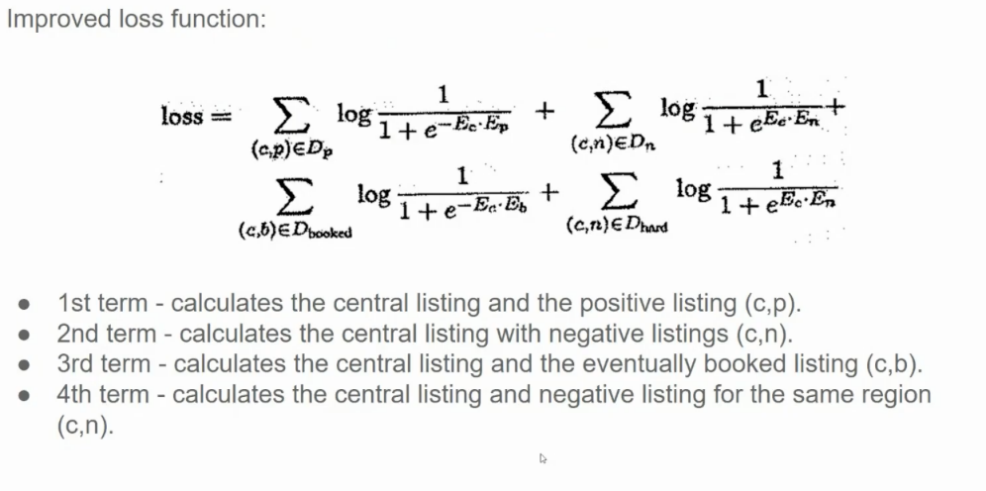
The Shortcomings:

1. **Positive Pair Limitation:**
   - The current approach of using similar listings as positive pairs might not be optimal for recommending "booked listings."
   - The model might learn to associate similar listings, but it might not capture the essence of popular or highly-rated listings.
2. **Negative Pair Limitation:**
   - Using completely dissimilar listings as negative pairs might not be the best strategy.
   - For instance, two listings in the same region might be considered similar, even if they have different features. Using them as negative pairs could hinder the model's ability to capture regional preferences.

Potential Improvements:

- **Hard Negative Mining:** Instead of randomly sampling negative pairs, the model could focus on the hardest negative examples, i.e., those that are most similar to the positive pairs.
- **Hierarchical Loss:** A hierarchical loss function could be used to capture relationships between different levels of listing categories (e.g., city, neighborhood, property type).
- **Contextual Embeddings:** Incorporating contextual information, such as user preferences and search history, into the embedding process can improve recommendation accuracy.





$$\text{loss} = \sum_{(c,p)\in D_p} \log \frac{1}{1+e^{-E_c \cdot E_p}} + \sum_{(c,n)\in D_n} \log \frac{1}{1+e^{E_c \cdot E_n}} + \sum_{(c,b)\in D_{\text{booked}}} \log \frac{1}{1+e^{-E_c \cdot E_b}} + \sum_{(c,n)\in D_{\text{hard}}} \log \frac{1}{1+e^{E_c \cdot E_n}}$$

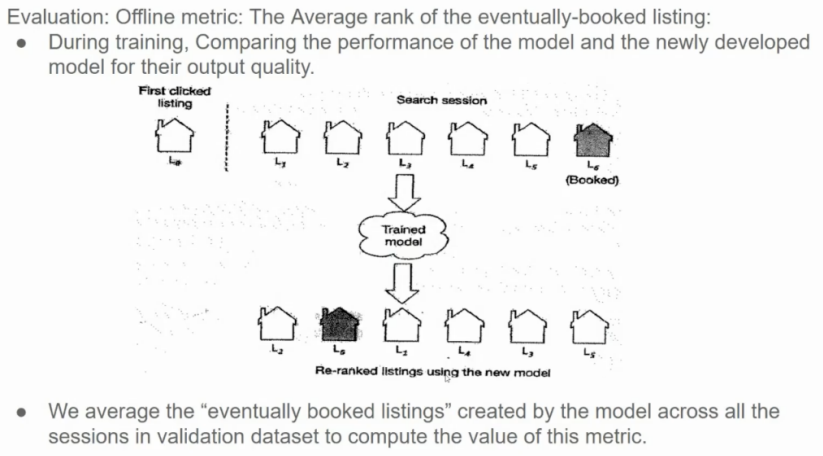
The improved loss function aims to address the limitations of the original loss function by incorporating additional terms to better capture the nuances of Airbnb listing recommendations. Let's break down each term:

1. Original Terms:

- **Positive Pair Loss:** This term encourages the model to bring embeddings of similar listings (positive pairs) closer together. It's calculated as the sum of the log-loss over all positive pairs (c, p) in the dataset D_p.
- **Negative Pair Loss:** This term pushes the embeddings of dissimilar listings (negative pairs) further apart. It's calculated as the sum of the log-loss over all negative pairs (c, n) in the dataset D_n.

2. Additional Terms:

- **Booked Listing Loss:** This term explicitly models the concept of "booked listings." It encourages the model to bring the embeddings of a central listing and the listing that was eventually booked closer together. This is calculated over the set of pairs (c, b) where b is a listing that was booked after viewing c.
- **Hard Negative Mining Loss:** This term focuses on hard negative examples, which are listings that are similar to the central listing but were not booked. It encourages the model to distinguish between these hard negative examples and the positive examples. This is calculated over the set of hard negative pairs (c, n) in the dataset D_hard.



This metric is used to evaluate the performance of a recommendation system, specifically in the context of Airbnb-like platforms. It measures how well the system can rank the listing that a user eventually books.

How it works:

1. **Data Collection:**
    - Collect a dataset of user search sessions. Each session includes:
        - The first listing clicked by the user.
        - A list of listings presented to the user.
        - The listing that the user eventually booked.
2. **Model Training:**
    - Train a recommendation model on this dataset. This model will learn to rank listings based on various factors, such as user preferences, listing features, and contextual information.

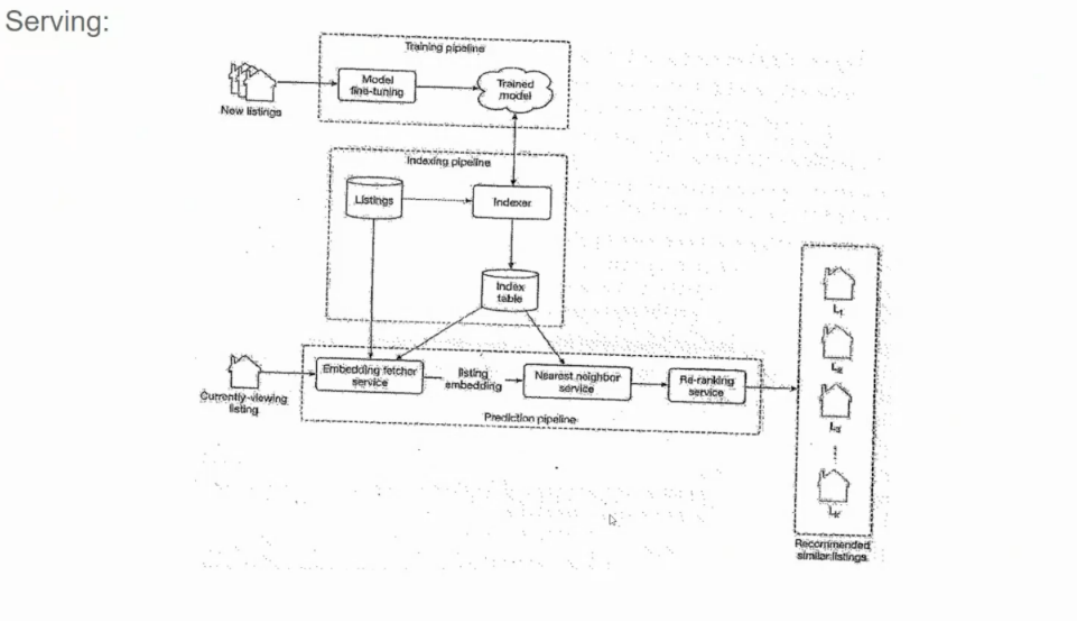
3. **Offline Evaluation:**
    - For each search session in the validation dataset:
        - Use the trained model to re-rank the listings presented to the user.
        - Calculate the rank of the eventually-booked listing in the re-ranked list.
    - Average the ranks of the eventually-booked listings across all sessions in the validation dataset.

Interpretation:

- **Lower Average Rank:** A lower average rank indicates that the model is better at ranking the eventually-booked listing higher in the search results. This suggests that the model is more effective in helping users find the listings they are interested in.
- **Higher Average Rank:** A higher average rank indicates that the model is less effective in ranking the eventually-booked listing highly. This suggests that the model needs improvement in its ability to predict user preferences.

Evaluation : Online metric.
B.O. - increasing the number of bookings.

1. Click-through Rate (CTR) = Number of clicked listing / number of recommended listing
2. Session book Rate = number of sessions turned into booking / total number of sessions.



The diagram illustrates a simplified architecture of an Airbnb-like recommendation system. It involves three primary pipelines:

1. **Training Pipeline:** Responsible for training the machine learning model.
2. **Indexing Pipeline:** Responsible for indexing listings and their embeddings.
3. **Prediction Pipeline:** Responsible for generating recommendations for a given listing.

Training Pipeline:

- **New Listings:** New listings are continuously added to the system.
- **Model Fine-Tuning:** The existing model is fine-tuned on these new listings to improve its accuracy.
- **Trained Model:** The fine-tuned model is saved for future use.

Indexing Pipeline:

- **Listings:** Existing listings are stored in a database.
- **Indexer:** The indexer processes listings and generates embeddings for each listing. These embeddings capture the semantic and visual features of the listing.
- **Index Table:** The embeddings are stored in an index table, which allows for efficient retrieval of similar listings.

Prediction Pipeline:

- **Currently Viewing Listing:** The user is currently viewing a specific listing.
- **Embedding Fetchor Service:** Fetches the embedding of the currently viewing listing.
- **Nearest Neighbor Service:** Uses the embedding to find similar listings from the index table.
- **Re-ranking Service:** Re-ranks the similar listings based on additional factors like user preferences, booking history, and real-time signals.
- **Recommended Similar Listings:** The re-ranked list of similar listings is presented to the user as recommendations.

Key Components and Their Roles:

- **Model Fine-tuning:** Ensures that the recommendation system stays up-to-date with the latest trends and user preferences.
- **Embedding Generation:** Creates numerical representations of listings, enabling efficient similarity calculations.
- **Indexing:** Enables fast retrieval of similar listings based on their embeddings.
- **Nearest Neighbor Search:** Finds the most similar listings to the current listing.
- **Re-ranking:** Improves the quality of recommendations by considering additional factors.

Additional improvements:

- What is a positional bias and how to address it?
- How does a session-based approach compare to random walk, random walk to random walk with restart (RWR) can be used to recommend similar listings.
- How to use user's long term interest in session personalization for session-based recommendation systems?
- How to incorporate the seasonality into our similar listing system?