**ML System Design Similar Listings on Vacation Rental Platform**

Video ID: https://www.youtube.com/watch?v=2fF6aQGLQQc&list=PL\_b\_MRp1BnEj4UuHv1jAGeO1a0VTRXsAk&index=15

Business Objective:

A black text on a white background

Description automatically generated

Clarifying the Requirements

A screenshot of a white page

Description automatically generated

Framing problem as ML task

A diagram of a house

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a website

Description automatically generated

A diagram of a pair of pants and shoes

Description automatically generated

A screen shot of a diagram

Description automatically generated

A diagram of a house

Description automatically generated

Data

A screenshot of a data analysis

Description automatically generated

A screenshot of a white sheet

Description automatically generated

A screenshot of a web page

Description automatically generated

A white background with black text

Description automatically generated

A diagram of a network

Description automatically generated

A diagram of a graph

Description automatically generated

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.

**Why Negative Sampling?**

* **Efficiency:** Instead of comparing the central listing to every other listing in the dataset, we only compare it to a small subset of positive and negative examples. This significantly reduces computational cost.
* **Focus on Hard Examples:** The model is forced to distinguish between very similar listings (positive pairs) and very dissimilar ones (negative pairs). This helps the model learn more robust representations.

**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.

A diagram of a model

Description automatically generated

A math equations and numbers

Description automatically generated with medium confidence

**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:**

The authors identify two main shortcomings with this approach:

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:**

To address these limitations, the authors suggest exploring alternative strategies, such as:

* **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.

A screenshot of a computer

Description automatically generated

A math equations and formulas

Description automatically generated with medium confidence

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.

**Overall, the improved loss function aims to:**

* **Enhance Recommendation Quality:** By considering booked listings and hard negative examples, the model can learn more nuanced representations of listings.
* **Improve User Experience:** By providing more accurate and relevant recommendations, the model can enhance user satisfaction and engagement.
* **Address the Limitations of the Original Loss Function:** The additional terms help to overcome the shortcomings of the original loss function, which focused solely on general similarity and dissimilarity.

A diagram of a model

Description automatically generated

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.

A white background with black text

Description automatically generated

A diagram of a building

Description automatically generated

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.

A white text with black text

Description automatically generated