COMP9727: Recommender Systems
Project: Project Pitch and Design

Name: Yang Liu

ID:z5471680

1 Scope

The recommender system is aimed for the internet travel industry, especially reserving hotels on a site like Expedia. People who utilize it include both business and leisure travellers who have input basic search criteria like their location, check-in and check-out dates, and the size of their group, but they haven't yet picked which hotel to book. In the first viewport, we provide a ranked list of five hotel possibilities for each search, along with a "Show More" button that opens up a scrollable grid of twenty. This keeps the cognitive strain minimal while yet providing the algorithm space to show a range of options.

The UI is thought of as a responsive web first, which is also seen in the mobile app. A wire-frame mockup would show a search bar at the top, a horizontal card carousel with five hotel tiles (picture, name, price per night, star rating, distance to city centre), and two light-weight feedback controls on each tile: a heart icon to "save" and a little × to close. When you scroll, you may see a two-column grid of the other fifteen results. Clicking on a tile takes you to the hotel detail page, and making a reservation triggers a "conversion" event. We will utilize static HTML displays to mimic this flow in a classroom user research and record three signals: the first tile clicked, the time it took to click, and if the participant indicates they would book one of the hotels offered.

Real-time logging of both implicit feedback (clicks, dwell time) and explicit feedback (bookings, saves, dismissals). Using the events of the previous day, the production model is updated every night. Feature statistics (such as pricing trends and destination popularity) are sent to an online feature store, where the ranking layer can change in a matter of minutes. Every month, there is a thorough retraining on the two-year past to keep things from drifting. Cold-start hotels utilize a content-based vector made up of the 149-dimensional destination embedding, star rating, price bucket, and neighbourhood. New users go back to popularity within their geographic cohort until they have three interactions, at which point the hybrid model takes over.

Expedia's public dataset groups comparable hotels into 100 "hotel cluster" labels instead of listing each hotel separately. These clusters are like items in our recommendation task and fit perfectly with the Top 5 slate. This keeps sensitive information private while yet allowing for large-scale collaborative filtering (around 37 million previous interactions). Higher booking conversion drives revenue; each extra booking earns commission. Sponsored listings are also selectively added to lower ranks of the slate once the basic algorithm is set up.

2 Dataset

The public Expedia Hotel Recommendations log that was provided on Kaggle will be used in the research. There are around 37 million interaction rows in the training split, which come from logs from 2013 to 2014. There are also 149-dimensional destination embeddings in a separate destinations. csv. Each row shows one action taken by a user against a "hotel cluster," which is a group of hotels that are similar. It includes a hashed user ID, the search and hotel geography (user_location_country, hotel_market), the trip intent (srch_adults_cnt, srch_children_cnt, is_package), the device/channel, the search and stay dates, and two outcome flags: is_booking (1 = booking, 0 = click-through) and cnt (the number of times the same action happened in the same session). There are around 1.2 million different users and exactly 100 clusters. This makes the user–item matrix dense enough for collaborative-filter baselines and keeps the candidate space manageable for deep models.

We consider each booking row to be a positive label and prior clicks to the same cluster to be weak positives. We will take rows with is_booking=0 from other clusters in the same search window as negatives. This will help us get close to the listwise context that the original log doesn't provide. The destination embedding (d1-d149) gives us a learnt semantic vector that may be used to start up new hotels and improve a hybrid model. Numerical context like distance, party size, and lead time will go into feature-crosses or transformer side inputs. The time divides (2013–mid-2014 train, late-2014 validation, 2015 test) follow the natural order of events to avoid leaks.

The statistics are realistic in size and variety, but there are four things to keep in mind. First, hotel clusters are an internal abstraction, so we can't show a picture or location of a single property. Any UI mock-up must use external information to relate clusters back to a representative hotel. Second, the file is a privacy-filtered sample, which means that just a small part of Expedia's overall traffic and only the columns that are safe to share are shown. That implies that long-tail users, people who go on the same vacation over and over again for years, cancellations, and sold-out properties are not included. This might make offline stats seem like they are doing better than they really are. Third, we only see the clicked or booked rows for each search, not the whole impression set. This means that actual exposure bias is still not overcome. We lessen this by using negative sampling in each session, but we know that the evaluation we get is more like CTR than whole slate quality. Fourth, since the dataset was made for a leaderboard task (MAP@5), participants historically focused a lot on that metric. To avoid overfitting to the leaderboard, we will report a wider panel (Recall@K, NDCG, novelty, latency) and set aside a month that wasn't used in model selection.

3 Method

3.1 Matrix-Factorisation Collaborative Filtering

We begin with a classic implicit-feedback matrix factorisation because it gives us a transparent yard-stick and scales linearly with the sixty-five million user–cluster pairs in the filtered train set. Each booking is a positive event; non-clicked clusters that were shown in the same search window are sampled as negatives at a 1:3 ratio to approximate exposure. The interaction matrix R is factored into user and item latent vectors via Bayesian Personalised Ranking (BPR). BPR is a good fit here because it directly optimises pairwise ordering, which aligns with our MAP@K objective, and the 100-cluster item space keeps factor dimension modest so a full grid search over rank (32–128) and regularisation is feasible on a laptop. This model will expose cold-start pain points early—new users have no vector until they interact, and new clusters inherit the global bias only—thereby motivating richer variants.

3.2 Variant 1 | Temporal Weighting and Drift Control

Hotel preference clearly drifts with seasonality (e.g., ski resorts spike in winter). To acknowledge this we extend the factorisation with exponential time decay on the interaction strength and include month-of-year biases. Decay half-life will be tuned on the validation split (likely between 90 and 180 days). Because Expedia's test period sits a year after the training window, controlling drift is critical; in pilot trials on similar logs a +3-5 % absolute Recall@10 gain is common once ageing is introduced.

3.3 Variant 2 | Session-based Transformer

Matrix factorisation ignores within-session intent signals such as a sudden switch from city-centre hotels to beach resorts. We model that micro-intent using SASRec, a transformer encoder over the chronological click sequence inside each user session. Each cluster id is embedded, positional encodings are added, and the final hidden token is passed to a dot-product scorer against candidate clusters. The long sequences (median length ≈ 4 events, max ≈ 40) are well within transformer limits, so one encoder block with eight heads suffices. Unlike MF, SASRec can recommend sensibly for anonymous users who generate a short sequence in a single visit. It couples naturally with our nightly retraining cadence because updating only the user-history cache is enough to refresh recommendations without touching item parameters.

3.4 Two-Stage Hybrid Pipeline

In production we do not serve MF or SASRec directly; instead we adopt a candidate–ranker architecture that mirrors industrial practice and allows each method to contribute what it does best. Stage 1 returns up to 300 candidates per search from a union of three recall sources: (i) top 40 clusters from temporal MF, (ii) top 40 from SASRec if the session length ≥ 2, and (iii) popularity within the destination when orig_destination_distance < 50 km, filling the remainder with global popularity. Stage 2 reranks those candidates with a gradient-boosted decision tree model (LightGBM-LTR) that consumes twelve handcrafted features: MF score,

SASRec score, destination embedding cosine similarity, price rank within slate, lead-time bucket, device type, package flag, and seasonality dummies. This hybrid captures collaborative, sequential, content, and context signals in one listwise objective (LambdaRank). Because LightGBM handles categorical splits efficiently it can be re-trained nightly on fresh logs without GPU.

3.5 Content-Side Cold-Start Back-off

Hotel clusters that appear for the first time in the test window would receive minimal score from MF and SASRec. We address this by pre-training a 256-dimensional auto-encoder on the d1–d149 destination vectors plus one-hot hotel geography, star rating bucket, and relative price bucket. The decoder side is discarded, leaving a content embedding that populates the item vector for unseen clusters. During inference, if a cluster has fewer than ten historical interactions we linearly interpolate its MF weight with the cosine similarity between its content embedding and the user's average embedding. A similar trick applies to new users: until three clicks are logged we blend MF/SASRec scores with cohort popularity determined by user_location_country. These heuristics are lightweight yet remove most of the harsh zero-score cases observable in pure CF.

3.6 Knowledge-Based Layer for Rule Compliance

Although not the project's core, we reserve a small rule-based overlay to guarantee business constraints—for instance excluding sold-out clusters or blacklisted partners from the Top 5 slate and boosting in-network sponsored clusters when the bid CPM surpasses a threshold. The overlay is implemented as post-processing penalties/bonuses applied to the final LightGBM score, ensuring explainability to business stakeholders without having to retrain the model each time a contract changes.

4 Evaluation

Offline evaluation starts with Expedia's own leaderboard statistic, Mean Average Precision at 5, because the interface presents a "Top 5" slate. MAP@5 gives points for proper rank order and takes points away for duplicate suggestions per user. To go beyond that one metric, we report Recall@10 and NDCG@10, averaged across all users. These are more sensitive to lower-rank relevance and let us compare fairly with models that yield larger candidate lists. Coverage (the percentage of the 100 clusters that were ever recommended) and Expected Popularity Bias measure how many long-tail hotels the model finds. These balance-checks are important since a simple popularity model can have a high MAP score but demonstrate little newness.

A two-stage gate chooses the best offline model. First, we need MAP@5 to improve statistically significantly (paired t-test, $p < 0.05$) over the baseline MF. We pick the model with the greatest harmonic mean of NDCG@10 and Coverage from among those that pass. This formula provides relevance and variety about the same amount of weight and stops people from tuning too much to one metric.

Live interaction logs or a controlled simulation based on those logs are used to get system-level metrics. The main KPIs are the click-through rate on the slate and the booking conversion rate. These are calculated for each session and then averaged to reduce the effect of heavy users. Time-to-first-click (engagement) and revenue-per-thousand-impressions are two secondary KPIs. The latter is calculated by multiplying each booking by the historical commission of its cluster. The 95th percentile is used to quantify latency. Candidate retrieval must finish in less than 200 ms, and ranker inference must finish in less than 50 ms on an 8-core CPU. If this SLO is not met, the model is no longer viable, no matter how accurate it is. We keep an eye on nightly retraining time (less than 90 minutes) and incremental feature store updates (less than 10 minutes) to make sure the pipeline can respond to demand shocks within a business day.

Since we can't distribute to Expedia users, we'll do a modest lab research with twenty people. In a web mock-up, each participant does five hotel searches. For each search, we randomly assign either the baseline MF slate or the hybrid ranker slate (within-subject Latin square). We keep track of implicit signals like the first item clicked, how long someone stays on the detail page, and when they leave a slate. After each slate, the user fills out a four-question Likert scale about how relevant, varied, easy to choose, and confident they are about booking. The last open-ended question asks what they liked or didn't like. We use Wilcoxon signed-rank tests to look at the questionnaire and click metrics to triangulate the results. This mixed-method approach captures subjective quality that the offline logs can't.

There are clear records of trade-offs. More variety may reduce MAP but make things seem newer; if the user survey reveals a strong preference for the diverse slate, we will accept a minor MAP loss. Adding deep sequential features also increases relevance but slows down inference time. We limit model complexity to 95-p latency below 250 ms since we think that travellers are more likely to leave sluggish pages than notice a little increase in accuracy. We make sure that the chosen model is not only academically solid but also able to be used in a real-world OTA context by integrating ranked-list measurements, real-time limitations, and feedback from people.