Files in the Dataset

1. yelp_academic_dataset_business.json

- **1.2M+ businesses**
- **Key fields**:
    - business_id: Unique ID
    - name: Business name
    - address, city, state, postal_code, latitude, longitude
    - stars: Average star rating (float)
    - review_count: Number of reviews
    - is_open: Open status (1 = open)
    - categories: Comma-separated categories (e.g., "Japanese, Sushi Bars, Restaurants")
    - attributes: Nested dict with metadata (e.g., “WiFi”, “Parking”, “Good for kids”)

Use in RL:

- To enrich item context (e.g., categories, popularity)
- To support action selection features

---

2. yelp_academic_dataset_review.json

- **8.6M+ reviews**
- **Key fields**:
    - review_id
    - user_id, business_id: Relational links
    - stars: Actual rating (1–5 stars)
    - date: Timestamp of the review
    - text: Freeform review text (not always used in structured models)
    - useful, funny, cool: Count of feedback from other users

Use in RL:

- Forms the core of (state, action, reward) triplets

- The rating becomes reward
- Enables sorting by time to form sequences

---

3. yelp_academic_dataset_user.json

- **2.1M+ users**
- **Key fields**:
    - user_id, name
    - review_count
    - yelping_since: Registration date
    - friends: Comma-separated user_ids
    - useful, funny, cool: Aggregate votes
    - fans, elite: Popularity/social influence indicators
    - average_stars: Avg. rating given

Use in RL:

- Used to enrich the **state** with user profile (average_stars, review_count, friend count)
- Can personalize action-value function

---

4. yelp_academic_dataset_checkin.json

- **Check-in records per business**
- **Key fields**:
    - business_id
    - date: Comma-separated timestamps (e.g., "2018-01-01 09:00:00, 2018-01-02 18:00:00")

Use in RL:

- Extract time-based popularity (morning / afternoon / evening visits)
- Serves as context feature for action (business popularity signal)

---

5. yelp_academic_dataset_tip.json

- **Short tips (mini reviews)**
- **Key fields**:
    - user_id, business_id, text, date
    - compliment_count

Use in RL:

- Optional text-based feedback signal
- Can complement reviews for sentiment analysis or explanation

---

Schema Relationships

User (user_id)

↕

Review (user_id, business_id, stars, date)

↕

Business (business_id, categories, attributes)

↕

Check-in (business_id, date)

Key Fields Mapped into RL Format

state: 5 recent interactions before action

Each item in state includes:

- business_id: ID of the visited business
- stars: Rating given (used for behavior modeling)
- date + time_segment: Timestamp + time of day classification (morning/afternoon/evening)
- user_profile: Dict with review_count, average_stars, friend_count
- business_checkin: Dict of check-in counts in each time segment
- business_categories: Up to 5 most relevant business categories
- days_since_action + recency_weight: Time-aware decay weighting (computed dynamically)

action: The current business under recommendation

Includes:

- business_id
- action_time_segment: When the user acted on the recommendation
- user_profile, business_checkin, business_categories: same format as state

reward:

- Set to 1 if stars ≥ 4, else 0, as per DRL reward heuristics (Chen et al., 2023)

next_state:

- Sliding window of the new state after action, formatted like state

Output Files

The processed data is saved to:

- rl_dataset_train.json – interactions before the 80% timestamp split per user
- rl_dataset_test.json – interactions after the split

Each file is saved in chunked JSON format, supporting scalability and future sampling or batch loading.

Example of data:


```
{
 "user_id": "abc123",
 "state": [
  {
   "business_id": "biz001",
   "stars": 5.0,
   "date": "2022-01-10 09:30:00",
   "time_segment": "morning",
   "days_since_action": 3,
   "recency_weight": 0.740818,
```

```
    "user_profile": {
      "review_count": 45,
      "average_stars": 3.9,
      "friend_count": 12
    },
    "business_checkin": {
      "morning": 22,
      "afternoon": 18,
      "evening": 34
    },
    "business_categories": ["Cafes", "Bakeries"]
  },
  {
    "business_id": "biz002",
    "stars": 3.0,
    "date": "2022-01-11 14:10:00",
    "time_segment": "afternoon",
    "days_since_action": 2,
    "recency_weight": 0.818731,
    "user_profile": {
      "review_count": 45,
      "average_stars": 3.9,
      "friend_count": 12
    },
    "business_checkin": {
      "morning": 5,
      "afternoon": 40,
      "evening": 25
    },
    "business_categories": ["Japanese", "Sushi Bars"]
  },
  ...
],
"action": "biz006",
```

```
"action_time_segment": "evening",
"user_profile": {
 "review_count": 45,
 "average_stars": 3.9,
 "friend_count": 12
},
"business_checkin": {
 "morning": 10,
 "afternoon": 12,
 "evening": 65
},
"business_categories": ["Bars", "American (New)"],
"reward": 1,
"next_state": [
 {
  "business_id": "biz002",
  ...
 },
 {
  "business_id": "biz006",
  "stars": 4.0,
  "date": "2022-01-13 20:15:00",
  "time_segment": "evening",
  "user_profile": {
   "review_count": 45,
   "average_stars": 3.9,
   "friend_count": 12
  },
  "business_checkin": {
   "morning": 10,
   "afternoon": 12,
   "evening": 65
  },
  "business_categories": ["Bars", "American (New)"]
```

```
  }
 ]
}
```

Missing value:

stars / reward: 0, No rating means unsatisfactory
user_profile: {'review_count': 0, 'average_stars': 0.0, 'friend_count': 0}, Simplifying vector space
business_checkin: {'morning': 0, 'afternoon': 0, 'evening': 0}, Ensure that each business has complete features
date / time_segment: Skip invalid records, cannot sort or create sequence without time