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)

In RL Framework

RL Concept	Dataset Source	Explanation
state	Review history	Sequence of previous business_id, stars, date, user profile, business features
action	Business	The next business to recommend
reward	Review	Usually 1 if stars ≥ 4, else 0
next_state	Review	User's updated sequence after action
context_features	Business, Check-in, User	Additional info like categories, check-in frequency, average_stars

Component	Feature Name	Source File	Description
state	business_id	review.json	Previously interacted businesses
	stars	review.json	Rating the user gave to that business
	date	review.json	Timestamp of the review (for sequence)
	time_segment	derived from date	Morning / Afternoon / Evening
	days_since_action	computed from date difference	Used for time-decay weight
	recency_weight	computed	exp(-λ × time_diff)
	user_profile.review_count	user.json	Number of reviews the user wrote
	user_profile.average_stars	user.json	User's mean rating
	user_profile.friend_count	user.json	Proxy for social influence
	business_checkin.morning/afternoon/evening	checkin.json	Popularity at different times
	business_categories	business.json	Category tags (multi-label)
action	business_id	from state or model selection	The business recommended to the user
	action_time_segment	review.json	When the recommendation occurred
	business_checkin	checkin.json	Contextual popularity
	business_categories	business.json	Contextual metadata
reward	reward	review.json	Binary reward: 1 if stars ≥ 4, else 0
next_state	same as state shifted forward	review.json	Resulting state after the user reacts to the action

---

Notes:

- state is a **sequence** of 5 (or more) past interactions
- action is typically a **single business_id**

- reward is derived from the user’s rating
- next_state = state + action