Search Engineer - Re-Ranking Candidate Facing

Overview: We want you to improve upon a baseline search pipeline over a subset of our off platform search data. We'll provide you access to an indexed snapshot of around 200k rows. Then, we want to build a retrieval engine on that collection. To validate your results, you can use our private evaluation endpoint, which will give you a measure of whether your search results are good or not.

Your goals:

1. **Query** a provided **Turbopuffer** (TPUF) collection of ~100K–200K user profiles.
2. Optionally **re-rank** the initial hits using any approach you prefer (cross-encoder, LTR features, rules, etc.).
3. **Evaluate** via our live API for each provided query config.

Relevant keys:

```
TURBOPUFFER_API_KEY=tpuf_wTbagsVtzNmVfzDm48lNeszzJdTaCOUF VOYAGE_
API_KEY=pa-vNEmoJfc5evP_SSvpxIAj3uFzs9dfppEZkpx-3kOFZy OAI_KEY=sk
-proj-RBYjP4yoZsKpvAwY0L6xWPYwxXOEZxHl7wfnx4RMriDN9GLuQtV3-F_6GJo
kPskn-k0zqkNPlcT3BlbkFJx80LWXg_FlIuoDGbMlB36NcxhQK7bmfgYgUR0dGo40
znghKQ2OefrWcuMSt2CvznNZRwXYfaAA
```

1. Resources

- **Read-only TPUF collection** with embedded vectors and key metadata.
  - Schema
    - `_id` (Mongo ObjectId string)
    - `updated_at` (ISO-8601 timestamp)
    - `linkedin_id` (LinkedIn identifier)
    - `email` (string)
    - `name` (string)
    - `country` (string)
    - `vector` (embedding - 1024 dims based on re-rank summary and embedded using **voyage-3** by Voyager AI)
    - `rerankSummary` (consolidated bio/summary)
    - `education`, `experience` (lists of strings)
    - `deg_start_years` (string array, e.g., ["2012","2019"])
    - `deg_end_years` (string array, e.g., ["2015","2023"])
    - `deg_years` (bucketed duration strings)
    - `deg_degrees` (string array, e.g., ["Master's","Doctorate"])
    - `deg_schools` (normalized school names)
    - `deg_fos` (fields of study)
    - `exp_titles` (title list)
    - `exp_companies` (company list)
    - `exp_start_years` (per-experience start years)
    - `exp_end_years` (per-experience end years, "" if current)
    - `exp_years` (bucketed duration strings)

- **Common Field Components:**
  - `yrs_[bucket]`: One of "1", "3", "5", "10", indicating duration in + format (e.g., "3" means 3+ years)
  - `start_[year]` and `end_[year]`: Start and end years (calendar year only). Fields can be empty (e.g., `end_`)
  - **Experience** (array)
    - Each experience string has the format:

```
yrs_[bucket]::title_[title]::company_[company]::start_[year]::end_[year]
```

    - `title_[title]`: Job title
    - `company_[company]`: Company name
  - **Education** (array)
    - Each education string has the format:

```
yrs_[bucket]::school_[school]::degree_[degree]::fos_[field_of_study]::start_[year]::end_[year]
```

    - `school_[school]`: School name
    - `degree_[degree]`: Degree obtained (standardized to valid degree types: "bachelor's", "master's", "mba", "jd", "doctorate", "certificate", "high school", "associate")
    - `fos_[field_of_study]`: Field of study
  - `rerank_summary`: A full text summary of their profile. The associated embedding of a profile is made by embedding this summary.

Example Turbopuffer Row

2. How to query turbopuffer

Turbopuffer Quickstart (1)

3. Test queries

When developing, we recommend testing your search against the following queries. You'll find that each of these has a natural language description, hard criteria, and soft criteria. You're welcome to pass any combination of these inputs into the search system. In reality, we often have a query rewriting layer in our prod systems, which will rewrite all of them into an ideal query for our search layer. You can think of the hard criteria as must have requirements in the candidates that are surfaced. If these criteria aren't met, then this person cannot be hired, which is really unideal. Soft criteria, I would think of as nice to haves. You should build your system to first optimize for the hard criteria and only then switch to thinking about soft criteria.

1BggMRCZ0BBRhrhOJWAQqE_3FPO85E7fMVKF3I1d_S0Q

There are two kinds of queries:

- **Public set queries**: these you'll have access to immediately and you can use these to inform the design of your system. While you can technically overfit to these, please note that we will also have a private query set which you can't see, which we'll use to evaluate your performance.
- **Private Queries:** These queries you will not have access to until a second-stage live interview presentation if you make it past this round. The idea behind these queries is to see how your system does on queries that you haven't seen before. 3 of these queries will mirror the public set. The other 7 are designed to test robustness to distribution shifts.

---

4. Evaluation Endpoint

You can evaluate your search system using this evaluation endpoint. Simply use your retrieval system to generate a ranked list of candidates for a given query. Then pass in the object ids of your best candidates (in sorted order) into the function below. For each of the queries, you will find that they also have an associated .yml linked. If you pass through this .yml into the endpoint below, it will evaluate the quality of your ranked candidates. Make sure you include your full name and the right config path to ensure we can grade your submission accurately.

Use the same email address for the authorization header that you received the take home assignment to.

```
POST https://mercor-dev--search-eng-interview.modal.run/evaluate
```

Headers

```
Content-Type: application/json Authorization: <YOUR_EMAIL>
```

Request Body

Field	Type	Description
config_path	string	Name of the `.yml` config file for the evaluation setup.
object_ids	array of string	Array of object IDs for the users that you are submitting that correspond to the `_id` field in the MongoDB `linkedin_data_subset` database. **Min of 1 and max of 10. Your private score submissions must include 10.**

Example Body

```
{ "config_path": "tax_lawyer.yml", "object_ids": [ "67970d138a14699f161
4c6b6", "679508a7a1a09a48feaadf0c" ] }
```