Global Jobs Scraper v2.7

Location-accurate, faster, and international-aware job search using Google Programmable Search (Custom Search) + schema.org `JobPosting` extraction.

This tool focuses on: - **Strict city/state/country matching** (toggleable) so you don't get random locations. - **Closed/expired job skipping** (HTTP 404/410, expired `validThrough`, and common "closed" phrases). - **Speed** via concurrent fetches + connection pooling. - **Cleaner results**: per-domain caps, spam domain blacklist, relevance scoring.

Features

- **Google CSE** queries with optional freshness (last 7/30/90/180/365 days).
- Extracts rich job data from `application/ld+json` (`JobPosting`) when available; falls back to robust heuristics.
- **International support** (US/EU/UK/NL/IN/AU/CA/SG etc.) + region awareness (US states, CA provinces, AU states).
- **Remote role handling** (allow/deny based on prompt).
- **CSV output** with normalized fields and reason codes (e.g., `location_match_reason`).
- **Good defaults** + adjustable constants at the top of the script.

Requirements

- Python **3.9+** (3.10/3.11 recommended)
- Packages: `requests`, `beautifulsoup4`, `pandas`, `python-dateutil`
- A Google Cloud project with **Custom Search JSON API** enabled
- A **Programmable Search Engine (CSE)** configured to **search the entire web**

Setup (Environment)

1) Create a Python virtual environment

```
python3 -m venv .venv
source .venv/bin/activate   # Windows: .venv\Scripts\activate
python -m pip install --upgrade pip
pip install -r requirements.txt  # or: pip install requests beautifulsoup4
pandas python-dateutil
```

> If you don't have `requirements.txt`, create one with these lines:
>
> ```
> requests
> beautifulsoup4
> ```

```
pandas
python-dateutil
```

2) Get a Google API Key

1. Go to **https://console.cloud.google.com/** and create/select a project.
2. **Enable API**: Search for **“Custom Search API”** and enable it for your project.
3. **Credentials** → **Create credentials** → **API key**.
4. (Recommended) **Restrict** the key to the **Custom Search API** and your IP/domain if relevant.

3) Create a Programmable Search Engine (CSE)

1. Go to **https://programmablesearchengine.google.com/**.
2. **Add** a new search engine.
3. Under **Sites to search**, choose **“Search the entire web”** (or include target job sites).
4. Note your **Search engine ID** (aka **CSE ID** / `cx`).

> ⚠️If you can’t see “Search the entire web”, use the **Control Panel** → **Basics** → set **Search the entire web**. Some accounts may need to first add one site then expand to entire web.

4) Set environment variables

```
export GOOGLE_API_KEY="YOUR_API_KEY"
export GOOGLE_CSE_ID="YOUR_CSE_ID"
```

Windows (PowerShell):

```
$env:GOOGLE_API_KEY="YOUR_API_KEY"
$env:GOOGLE_CSE_ID="YOUR_CSE_ID"
```

> You can also store these in a `.env` file (do **not** commit it) and `source` it:
>
> ```
> echo 'export GOOGLE_API_KEY=YOUR_API_KEY' >> .env
> echo 'export GOOGLE_CSE_ID=YOUR_CSE_ID' >> .env
> source .env
> ```

Running the scraper

Interactive mode (recommended first run):

```
python3 global_jobs_scraper.py
```

You’ll be prompted for roles, cities, job types, experience, country, remote inclusion, CSV path, etc.

Examples

- **Amsterdam principal engineer** (strict city matching, remote allowed):
- Role: `Principal Engineer`
- City: `Amsterdam`
- Country: `Netherlands`
- Allow remote: `Y`

- **US cities** (e.g., Austin, Seattle) for SRE roles:

- Role: `SRE`
- Cities: `Austin, Seattle`
- Country: `US`

- **Australia** (Sydney/Melbourne) for Data Engineer:

- Role: `Data Engineer`
- Cities: `Sydney, Melbourne`
- Country: `Australia`

Output (CSV)

Columns include (subset): - `source_url`, `title`, `company`, `location`, `city`, `date_posted`, `valid_through`, `employment_type`, `salary`, `description`, `apply_url`, `job_id`, `remote` - `experience_level_detected` - `normalized_city`, `normalized_country`, `normalized_region` - `location_match_reason` (e.g., `city:amsterdam`, `country:netherlands`, `remote-allowed`) - `closed_reason` (when a posting was detected as closed/expired and skipped)

If **no results** matched your filters, the tool prints a clear message and still writes an empty CSV for traceability.

Important knobs (top of the script)

- `STRICT_LOCATION = True`
  Enforces strict city/state/country matching. Set to `False` to allow near matches.
- `MAX_WORKERS = 14`
  Increase for more concurrency; be mindful of target sites' rate limits.
- `PER_DOMAIN_CAP = 3`
  Maximum results per registrable domain (pre- and post-fetch).
- `BLACKLIST_DOMAINS = {...}`
  Add noisy/spam sites here.
- `MIN_RELEVANCE_SCORE = 2`
  Quick title/description relevance threshold.

Troubleshooting

- `Missing GOOGLE_API_KEY or GOOGLE_CSE_ID`: Ensure environment variables are set in the *same shell* where you run the script.
- **HTTP 429 / CSE daily limit**: You've hit Google quota. Consider adding billing to raise quotas, wait for reset, or reduce queries.
- **Too few results**:
- Add more roles or nearby cities.
- Expand freshness (the script automatically widens from 7→365 days).
- Set `STRICT_LOCATION=False` or enable remote roles.
- Add major job boards to your CSE or pass custom `domains` when prompted.
- **Many links from one site**: Lower `PER_DOMAIN_CAP` (or raise if you want more).
- **Wrong country bias**: Provide the country prompt (e.g., `US`, `Netherlands`) so the script can set `gl` and filter more accurately.

Legal & Privacy

- Respect each site's **Terms of Service** and **robots.txt**.
- Use reasonable request rates. The script attempts to be polite but high concurrency can still stress sites.
- Do not store or distribute data in violation of site policies.

Project structure (suggested)

```
.
├── global_jobs_scraper.py        # the script
├── requirements.txt              # python deps
├── README.md                     # this file
├── .env                          # (optional) your API keys (untracked)
└── .gitignore
```

.gitignore (example):

```
.venv/
__pycache__/
*.pyc
.env
.DS_Store
*.csv
```

How to publish on GitHub

1. **Create a repo**

```
gh repo create global-jobs-scraper --public --source=. --remote=origin
--push
```

If you don’t use GitHub CLI, do it manually:

2. Create a repository on GitHub named `global-jobs-scraper` (no README).

3. Then run:

```
git init
git add global_jobs_scraper.py README.md requirements.txt .gitignore
git commit -m "feat: initial release of Global Jobs Scraper v2.7"
git branch -M main
git remote add origin https://github.com/<YOUR_USER>/global-jobs-
scraper.git
git push -u origin main
```

4. **(Optional) Add a license**
   Create `LICENSE` with MIT:

```
MIT License

Copyright (c) 2025 <Your Name>

Permission is hereby granted, free of charge, to any person obtaining a
copy
of this software and associated documentation files (the "Software"), to
deal
in the Software without restriction, including without limitation the
rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or
sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
...
```

   Then:

```
git add LICENSE
git commit -m "chore: add MIT license"
git push
```

5. **Tag a release** (optional)

```
git tag v2.7.0
git push origin v2.7.0
```

6. **Open-source hygiene**

7. Keep API keys **out of git** (`.env` in `.gitignore`).
8. Document any breaking changes in the README.

FAQ

Q: Does this work for Amsterdam / US / AU / EU cities?
Yes—supply the **city list** and optional **country**. The script normalizes city aliases (e.g., `NYC → New York`, `Bengaluru → Bangalore`) and is aware of regions/states.

Q: How does it avoid closed jobs?
We check HTTP status, `validThrough` dates, and typical "closed/expired" phrases; such results are skipped and tracked in `closed_reason`.

Q: I'm still seeing some off-location links.
Leave `STRICT_LOCATION=True`. If the page hides location, JSON-LD/heuristics may lack signals. Adding the **country prompt** improves recall; you can also pass custom allowed domains.

Q: Can I turn off interactive prompts?
Current version is interactive; CLI flags can be added. Open an issue or PR and we'll extend it.

Happy scraping! If you run into bumps, open an issue with your prompt, console log, and (if possible) a small CSV sample.