

Look-Alike Cities

Methodology & Statistical Approach

How IDiyas identifies cities with similar innovation profiles

IDiyas Patent Intelligence

Internal Technical Documentation

Contents

1. Overview.....	3
2. The Data Foundation	3
2.1 What a “city” is.....	3
2.2 The CPC classification system	3
3. Representing a City as a Vector.....	4
4. Measuring Similarity.....	4
4.1 The intuition.....	4
4.2 Pearson correlation over the CPC profile.....	4
4.3 Key alignment step: the union of codes	5
5. The Computation Pipeline	6
Phase 1 — Build the CPC profiles	6
Phase 2 — Compute pairwise similarity.....	6
5.1 Computational scale	6
6. A Worked Example	6
7. Methodological Notes & Limitations	7
Profile-level correlation, not classical sampling	7
Raw counts favor large, broad cities	7
Linear co-movement only.....	7
Positive matches only	7
8. How Results Are Exposed.....	8

1. Overview

The Look-Alike Cities feature answers a deceptively simple question: which cities innovate in the same way? Two cities can differ wildly in size, geography, and cost of living yet share a near-identical technological fingerprint — the same mix of semiconductors, biotech, software, or materials research. IDiyas surfaces those hidden matches.

Each city page links to its live results, for example [Cities with similar research — Charlotte, NC](#).

In one sentence

We represent every qualifying city as a vector of patent counts across technology categories (CPC codes), then measure how closely any two cities' vectors move together using a Pearson-style correlation.

This document describes the data source, the statistical method, the exact computation pipeline as implemented, known limitations, and how the results are exposed to users and partners.

2. The Data Foundation

2.1 What a “city” is

A city in the IDiyas graph is a geographic place to which patents are attributed through their inventors and assignees. Every patent in the corpus carries one or more place identifiers; aggregating across the full corpus yields, for each city, a complete record of its patenting activity.

2.2 The CPC classification system

Patents are categorized using the Cooperative Patent Classification (CPC) scheme — a hierarchical system jointly maintained by the USPTO and the European Patent Office. Each patent is tagged with one or more CPC codes describing its technical subject matter.

CPC is organized into nine top-level sections:

- **Section A:** Human Necessities (agriculture, food, medical, personal items)
- **Section B:** Performing Operations / Transporting
- **Section C:** Chemistry / Metallurgy
- **Section D:** Textiles / Paper
- **Section E:** Fixed Constructions
- **Section F:** Mechanical Engineering, Lighting, Heating, Weapons
- **Section G:** Physics (computing, instruments, optics)
- **Section H:** Electricity (semiconductors, communications)
- **Section Y:** Emerging / Cross-Sectional Technologies

IDiyas builds city profiles at the CPC subgroup level (e.g. H01L29-78), which is far more granular than the section letter — there are tens of thousands of distinct subgroups. This granularity is what lets the method distinguish, say, a power-electronics city from a memory-chip city even though both sit in section H.

3. Representing a City as a Vector

For each city we construct a CPC profile: a dictionary mapping every CPC code that appears in the city's patents to the number of patents in that code. Conceptually:

```
Charlotte, NC profile (illustrative):
{
  "H04L9-32": 412,    // cryptographic mechanisms
  "G06F16-23": 388,    // database management
  "H04W12-06": 201,    // wireless security
  "G06Q20-38": 174,    // payment architectures
  ... thousands more CPC codes ...
}
```

This dictionary is the city's vector. Each CPC code is one dimension; the count is the magnitude along that dimension. A city active across many technologies has a long, dense vector; a specialized city has a sparse one concentrated in a few codes.

Why counts and not just presence

Using raw patent counts (rather than a yes/no flag per CPC code) preserves intensity. A city with 400 cryptography patents and a city with 4 are treated very differently — which is exactly what we want when comparing innovation focus.

4. Measuring Similarity

4.1 The intuition

Two cities are similar if their CPC profiles rise and fall together: where City A has many patents, City B also tends to have many, and where A is thin, B is thin too. This co-movement is precisely what a correlation coefficient captures.

4.2 Pearson correlation over the CPC profile

We use a Pearson-style correlation computed across the union of CPC codes from the two cities. For two profiles X and Y aligned over the same set of CPC codes, the coefficient is:

$$r = \frac{n \cdot \sum(xy) - \sum x \cdot \sum y}{\sqrt{[n \cdot \sum x^2 - (\sum x)^2] \cdot [n \cdot \sum y^2 - (\sum y)^2]}}$$

where, for each CPC code k in the union of both cities:
x = count of code k in City A (0 if absent)

```
y = count of code k in City B    (0 if absent)
n = number of distinct CPC codes in the union
```

The result r ranges from -1 to $+1$:

- **r near $+1$:** Profiles move together strongly — a strong technological look-alike.
- **r near 0 :** No linear relationship — unrelated innovation profiles.
- **r below 0 :** Profiles move oppositely — where one specializes, the other is absent.

4.3 Key alignment step: the union of codes

Before computing r , the two profiles are aligned over the union of their CPC codes. Any code present in one city but missing in the other is filled with zero. This ensures both vectors have the same dimensions and that a technology unique to one city correctly counts as “present here, absent there.”

5. The Computation Pipeline

The end-to-end process, as implemented in the production batch job, runs in two phases.

Phase 1 — Build the CPC profiles

- Step 1. For every city, retrieve all of its patents and read their CPC codes.
- Step 2. Tally the codes into a {code → count} dictionary, keeping only the nine valid sections (A–H and Y) and discarding malformed codes.
- Step 3. Store the resulting profile in a dedicated collection so it can be reused without re-scanning the entire patent corpus on every run.

Phase 2 — Compute pairwise similarity

- Step 4. Restrict the working set to cities with at least 50 patents (see threshold rationale below).
- Step 5. For each qualifying city, load its CPC profile as the reference vector.
- Step 6. Compare it against every other city's profile using the Pearson-style correlation, aligning on the union of CPC codes.
- Step 7. Keep only positive correlations, sort descending, and retain the top 10 matches.
- Step 8. Write the ranked matches back to the city record as its list of similar cities.

The 50-patent threshold

Cities with very few patents produce noisy, unstable profiles — a handful of patents can make two tiny cities look identical or wildly different by chance. Requiring at least 50 patents ensures each profile carries enough signal for the correlation to be meaningful. It also keeps the all-pairs computation tractable.

5.1 Computational scale

Similarity is an all-pairs comparison: every qualifying city is compared against every other. With C qualifying cities this is on the order of C^2 correlations, each spanning the union of two large CPC dictionaries. To keep this feasible the profiles are precomputed once (Phase 1), held in memory during the run, and the job is processed in batches with controlled memory limits.

6. A Worked Example

Suppose we compare three small illustrative cities across four CPC codes:

CPC code	City A	City B	City C
H04L9-32 (crypto)	400	380	10
G06F16-23 (databases)	350	330	20

CPC code	City A	City B	City C
H01L29-78 (transistors)	30	25	390
C07D (organic chem)	5	8	360

City A and City B both concentrate in cryptography and databases with little chip or chemistry work. Their profiles rise and fall together, so $r(A,B)$ is high — they are look-alikes. City C inverts the pattern: heavy in transistors and chemistry, light in crypto. So $r(A,C)$ is low or negative — not a match — even if City C is geographically next door.

The core insight

Look-alike status is about what a city builds, not where it sits or how big it is. That is why the feature can pair an expensive coastal tech hub with an affordable inland city an ocean away.

7. Methodological Notes & Limitations

In the spirit of transparent documentation, the following properties of the current approach are worth stating explicitly.

Profile-level correlation, not classical sampling

The correlation is computed over the set of CPC codes shared and unshared between two cities, with n being the number of distinct codes in the union. This is a profile-similarity measure in the Pearson family rather than a correlation over independent paired observations in the textbook sense. In practice it behaves like a centered, count-weighted similarity score and ranks look-alikes effectively.

Raw counts favor large, broad cities

Because profiles use absolute patent counts, very large cities have large magnitudes across many codes. The Pearson centering mitigates pure scale effects, but a future refinement could normalize each profile (e.g. by share of total patents, or via TF-IDF-style weighting that down-weights ubiquitous CPC codes) to further isolate specialization from sheer volume.

Linear co-movement only

Pearson captures linear association. Two cities related in a more complex, non-linear way could be under-scored. Alternative metrics (cosine similarity, rank correlation, or Jensen-Shannon divergence over normalized distributions) are natural candidates for comparison in future iterations.

Positive matches only

Only positive correlations are surfaced as look-alikes, which is the correct user-facing choice: negatively correlated cities are explicitly dissimilar and not useful as “similar research” suggestions.

8. How Results Are Exposed

For each qualifying city, the top 10 look-alike matches — each with its similarity score — are stored directly on the city record. This makes retrieval instant: the user-facing page simply reads the precomputed list rather than running any live computation.

Results are available in three places:

- **On-site city pages:** Each city profile shows its “Cities with similar research” list, linking through to the matched cities.
- **Refresh cadence:** The batch job is re-run as the patent corpus grows, so matches stay current as cities accumulate new patents.
- **Partner / API distribution:** The same ranked, scored matches are available programmatically — the basis of the predictive Look-Alike Cities dataset distributed through partners such as the Spartera Marketplace.

Summary

Look-Alike Cities turns the full IDiyas patent corpus into a per-city technological fingerprint (CPC profile), then ranks every city against every other by Pearson-style correlation of those fingerprints, filtered to cities with ≥ 50 patents and to positive matches. The output is a fast, precomputed, top-10 similar-cities list per city — powering both the website and partner data feeds.