OpenCódice Technical Report OC-TR-2026-009

orcid-mcp: A Model Context Protocol Server for the ORCID Researcher Registry, with Heuristic Author Disambiguation

Francisco-Javier Rodrigo-Ginés
OpenCódice Research
fran@opencodice.org

Jorge Chamorro-Padial
OpenCódice Research
jorge@opencodice.org

4 May 2026 • **DOI:** 10.5281/zenodo.20023535

Abstract

Author disambiguation is the load-bearing problem under nearly every scholarly workflow that an LLM agent is now expected to perform: citation deduplication, reviewer vetting, CV reconstruction, affiliation verification. Three decades of academic literature on author name disambiguation [Smalheiser and Torvik, 2009, Ferreira et al., 2012, Tang et al., 2012, Sanyal et al., 2021, Färber and Ao, 2022] converge on the same conclusion: any approach that infers identity from co-occurrence patterns, citation graphs, or institutional metadata leaves a non-trivial residual error. ORCID [Haak et al., 2012] is the canonical answer: a self-attested 16-digit identifier covering more than 20 million researchers and integrated by every major publisher and funder. There is no MCP server for ORCID. This report introduces `orcid-mcp`, a small open-source server that closes the gap. We document the architecture, the `/search` versus `/expanded-search` trade-off, the design of the disambiguation tool, and a real-data benchmark on twelve well-known computer-science researchers. The benchmark surfaces a result that matters for any agent built on the tool: ORCID's primary expanded-search response carries a non-null `institution-name` for only two of the twelve queries we ran, and matches the user-supplied affiliation hint for only one. Token-overlap scoring against a sparsely-populated affiliation field is therefore a real limitation; the practical fix is a v0.2 fallback that fetches each candidate's full employment history before scoring. The server is the third member of the OpenCódice academic-MCP family.

Keywords: Model Context Protocol, ORCID, author disambiguation, scholarly metadata, agentic research, reviewer vetting, identity resolution

Highlights

- Open-source MCP server exposing seven tools over the ORCID Public API, installable via `pip install orcid-mcp`
- Signature heuristic disambiguation tool that ranks candidates by token-overlap between an affiliation hint and each candidate's listed affiliations, returning evidence rather than enforcing a fixed taxonomy
- Defaults to ORCID's `/expanded-search` endpoint to surface name and institution per hit; the plain `/search` endpoint returns only ORCID iDs and was a v0.1 bug caught by live smoke testing
- Twelve-name benchmark on senior CS researchers shows that ORCID's primary `institution-name` field is empty for ten of twelve queries, surfacing a real limitation of any affiliation-based disambiguation built on `/expanded-search` alone
- Identifies the v0.2 fix as a per-candidate fallback to `orcid_get_employments`, whose richer affiliation history closes the gap for the same twelve names

1 Introduction

For an LLM agent reasoning over scholarly metadata, author disambiguation is the load-bearing problem under nearly every other task. Citation deduplication (*is the J. Smith on this 2018 paper the same as the J. Smith on this 2024 paper?*), reviewer vetting (*what does this candidate reviewer actually publish on?*), CV reconstruction (*walk this researcher's complete history of employments and works*), and affiliation verification (*does this paper's claimed corresponding-author affiliation match the registry?*) all reduce to the same underlying question: *is this the same person?*

Three decades of bibliometric and information-retrieval research [Smalheiser and Torvik, 2009, Ferreira et al., 2012, Tang et al., 2012, Sanyal et al., 2021, Färber and Ao, 2022] have circled this problem from many angles, classifying methods by feature set (string-based, co-author-graph-based, citation-based, content-based), by supervision regime (fully supervised, semi-supervised, unsupervised, ground-truth-bootstrapped), and by target corpus (DBLP, PubMed, MAG, OpenAlex). The consensus is sober: any approach that *infers* identity from co-occurrence or content patterns leaves a non-trivial residual error, on the order of two to ten percent of attributions, depending on the prolificness of the name and the density of the local citation graph [Sanyal et al., 2021, Färber and Ao, 2022]. The error is not a bug in any specific index. It is a structural property of inferring identity from indirect signals.

ORCID [Haak et al., 2012, ORCID, Inc., 2025] is the canonical answer to that structural problem. Researchers self-attest a 16-digit identifier (e.g. `0000-0001-6235-6860`), link it to their employments and educations, claim publications under it, and integrate it across publisher submissions, funder applications, and most major scholarly indexes. The registry now covers more than 20 million researchers [ORCID, Inc., 2025] and is the closest thing the community has to a global ground truth for "who is this person?". Adoption has accelerated since 2018: roughly 60–70% of recent computer-science journal authors carry an ORCID iD [Porter et al., 2025, Schnieders et al., 2022], with substantial geographic variance [Schnieders et al., 2022].

There is no MCP server for ORCID. This report introduces `orcid-mcp`, an open-source server that closes the gap. We make three contributions.

- **A complete MCP surface for the ORCID Public API.** Six tools cover the standard record traversal (profile, works, employments, educations, free-text search, structured search). The package ships as `pip install orcid-mcp`.
- **A signature heuristic disambiguation tool.** `orcid_disambiguate` ranks candidate ORCID iDs for an ambiguous name plus an affiliation hint, scoring each candidate by token-overlap between the hint and the candidate's listed affiliations. The score is returned as a number on every candidate, not as a binary accept/reject decision: the consuming agent decides whether to trust the top hit, query for more candidates, or fall back to other heuristics.
- **A real-data benchmark.** We run the structured-search tool against twelve well-known senior computer-science researchers (the names are stable, the affiliations are public knowledge, and the queries are naturally ambiguous because each name has multiple ORCID candidates). The result is a sharp engineering finding: ORCID's primary `/expanded-search` response carries a non-null institution name for only two of the twelve queries, and the user-supplied affiliation hint matches only one. Token-overlap on the field returned by `/expanded-search` is therefore a weak signal in practice; the right next step is to fall back, per candidate, to `orcid_get_employments` and score against the full employment history.

The server is released under the MIT license at `https://github.com/OpenCodice-Research/orcid-mcp` and is the third of five MCP servers OpenCódice Research is shipping for the academic stack.

The remainder of this report is organised as follows. Section 2 reviews the ORCID registry, the academic-disambiguation literature, and the existing landscape of academic MCP servers. Section 3 documents the system architecture. Section 4 catalogues the seven tools and the schema design. Section 5 discusses the disambiguation tool in detail. Section 6 reports a critical engineering finding from live smoke testing: the plain `/search` endpoint returns only ORCID iDs, and the production server uses `/expanded-search` instead. Section 7 presents the twelve-name benchmark and discusses the headline finding. Section 8 surveys downstream use cases. Sections 9 and 10 discuss limitations and place the server within the OpenCódice academic-MCP family.

2 Background

2.1 ORCID as the canonical identifier

ORCID was launched in 2012 as a non-profit, member-supported organisation tasked with solving the author-identity problem at internet scale [Haak et al., 2012]. The core abstraction is a 16-digit identifier (issued in the form `XXXX-XXXX-XXXX-XXXX`) that the researcher controls, claims publications under, and links to their employments, educations, peer-review activity, and external profiles (Scopus, ResearcherID, LinkedIn, GitHub). Critically, ORCID iDs are *self-asserted*: the researcher decides what the record says, which trades off accuracy (compared to a curated registry like DBLP) for coverage (any researcher with internet access can claim one) and accountability (the researcher is the authority on what they assert).

Adoption has been steady and is now the dominant identifier in scholarly metadata. Most major publishers' submission systems require ORCID iDs for corresponding authors; most funders' application portals link to them; Crossref and DataCite both treat ORCID as a first-class identifier in their metadata graphs. The registry passed 20 million researchers in 2024 and continues to grow at roughly 2 million per year. Coverage in computer science is roughly 65% of recent first authors [Schnieders et al., 2022, Porter et al., 2025], with continuing variance by discipline and country.

For an LLM agent, ORCID is the ground truth that every downstream disambiguation task should fold back to. A name search in Semantic Scholar may return five algorithmic clusters under one researcher's name; the right move is not to guess which is right, but to map each candidate to an ORCID iD when one is registered and reason about identity at the canonical layer.

2.2 Author name disambiguation as a research problem

Author name disambiguation has been studied for at least two decades. Smalheiser and Torvik [2009] provide an early systematic survey grounded in the PubMed corpus, formalising the task as a clustering problem over publication-author pairs. Ferreira et al. [2012] extend the survey to general bibliographic databases and partition the literature into author-grouping methods (cluster all papers by an author) and author-assignment methods (assign each paper to an author cluster), each with a wide range of feature engineering. Tang et al. [2012] present a unified probabilistic framework that integrates supervised, semi-supervised, and unsupervised learning over Markov random fields. More recent surveys [Sanyal et al., 2021] catalogue dozens of techniques specific to PubMed and observe that the per-paper error rate is remarkably stable: most automated systems sit between 95% and 98% accuracy on standard benchmarks, with the residual concentrated on prolific common names. Färber and Ao [2022] document the same residual on the Microsoft Academic Knowledge Graph and argue, like us, for grounding identity in author-controlled identifiers wherever possible.

The implication for an LLM-agent stack is that algorithmic disambiguation cannot be made arbitrarily accurate. Folding back to a self-attested registry is the right architectural move when the cost of confusing two researchers is high, as it is for reviewer vetting, integrity auditing, and

citation crediting. ORCID is that registry. `orcid-mcp` is the last mile.

2.3 Existing academic MCP servers

The MCP catalogue contains a growing number of academic wrappers. Academia MCP [Gusev, 2025] covers arXiv, ACL Anthology, Semantic Scholar, and Hugging Face. Smaller servers handle specific subsystems: arXiv-only wrappers, PubMed bridges, Crossref DOI resolution. None of them expose ORCID. Author profiles surfaced by Semantic Scholar are clusters produced by an algorithm, not canonical identities; OpenReview profiles include ORCID iDs but only when the user has linked them, and surface only the iD, not the full record.

2.4 Why an MCP wrapper

A natural objection to a dedicated wrapper is that an agent could call ORCID's REST API directly. The objection has the standard answer: a generic HTTP tool pushes three concerns onto the agent that should not live there. The agent has to know which endpoints to call (`/v3.0/{orcid}/person` for the basic profile, `/works` for the publication list, `/employments` for the affiliation history), normalise heterogeneous JSON shapes (ORCID's response schemas are deeply nested and full of `{value: ...}` wrappers and namespace-aliased keys), and decide on a search endpoint (`/search` or `/expanded-search`, see Section 6). A small wrapper that returns a Pydantic-validated flat dictionary per entity is dramatically easier for an agent to reason over than the raw ORCID JSON.

3 System architecture

`orcid-mcp` is a small Python package built around four concerns: a thin client wrapper over the ORCID Public API, a disk-backed cache, a set of pure-function tool implementations, and a FastMCP server that registers them as MCP tools. The structure mirrors the rest of the OpenCódice academic-MCP family.

3.1 Layered design

1. **Client wrapper** (`orcid_mcp.client`). Lazy-instantiates an `httpx.Client` against `pub.orcid.org/v3.0`. Public reads work without credentials but are rate-limited; with `ORCID_CLIENT_ID` and `ORCID_CLIENT_SECRET` set, the client transparently mints a 2-legged client-credentials token and reuses it for the life of the process.

2. **Cache** (`orcid_mcp.cache`). Diskcache-backed key-value store with per-tool TTLs (six hours for searches, twelve hours for works, twenty-four hours for profiles and affiliations, since these change less frequently). Cache is bypassed when `ORCID_MCP_NO_CACHE=1`.

3. **Schemas** (`orcid_mcp.schemas`). Pydantic v2 models: `ORCIDName`, `Affiliation`, `Work`, `Profile`, `SearchResult`, `DisambiguationCandidate`.

4. **Tools** (`orcid_mcp.tools`). Pure functions grouped by purpose: `profiles.py` (record), `works.py` (publications), `affiliations.py` (employments and educations), `search.py` (search and disambiguation).

5. **Server** (`orcid_mcp.server`). FastMCP application registering each tool with an `@mcp.tool()` decorator and an LLM-suitable docstring.

6. **Transport** (`orcid_mcp.cli`). Argparse-based CLI selects between `stdio` and `streamable-http`.

3.2 Authentication: the 2-legged token flow

ORCID's Public API supports two access modes. Anonymous access works for read endpoints but is rate-limited at roughly 24 requests per second across the public pool. Authenticated access via a 2-legged client-credentials token (the so-called Public API client) raises the budget substantially and grants access to the same data without further restrictions. Registering a Public API client is free at `orcid.org/developer-tools`. `orcid-mcp` reads `ORCID_CLIENT_ID` and `ORCID_CLIENT_SECRET` from the environment, exchanges them for a token at first call, and caches the token for the life of the `httpx.Client`.

The fallback path matters: if no credentials are set, the server runs against the anonymous pool. This makes the install path `pip install orcid-mcp` and a single CLI invocation, with no credential setup required for casual use. Production workloads should configure credentials.

3.3 Caching strategy

Most ORCID record state changes slowly. A researcher claims a paper, updates an employment, or links a new external identifier on the order of weeks; biographical fields change on the order of years. We cache aggressively (six to twenty-four hours per tool) and key on a stable hash of the function name and arguments, normalised through `json.dumps(..., sort_keys=True)`.

3.4 Failure semantics

Tools fail loudly on credential or transport errors but soft-fail on missing records: `orcid_get_record` on an unknown ORCID iD raises an exception the agent can catch and report. Search tools always return a list, possibly empty, never a null.

4 Tools

The server exposes seven tools grouped into four families. Table 1 summarises the surface.

Family	Tool	Purpose
Profile	`orcid_get_record`	Full public profile (name, biography, keywords, country, websites)
Works	`orcid_get_works`	All registered publications, datasets, software
Affiliations	`orcid_get_employments`	Past and current employments
	`orcid_get_educations`	Degrees and qualifications
Search	`orcid_search`	Free-text Solr query against the registry
	`orcid_search_by_name`	Structured search by given/family name and optional affiliation
	`orcid_disambiguate`	Rank ORCID candidates for an ambiguous name + affiliation hint

Table 1: The seven tools exposed by `orcid-mcp`, grouped by family. All tools are prefixed with `orcid_` when registered as MCP tools.

4.1 Schema design

Three choices are worth highlighting.

ORCID iD normalisation on the way in. The helper accepts `0000-0001-6235-6860`, `https://orcid.org/0000-0001-6235-6860`, or `orcid.org/0000-0001-6235-6860` interchangeably and returns the canonical 16-digit form with hyphens. This avoids cache-key fragmentation across input variants.

Year extraction from nested date wrappers. ORCID dates are wrapped as `{"year": {"value": "2024"}, "month": {"value": "06"}, ...}`. The `parse_year` helper walks the structure and returns an integer or `None`, so consumers compare years with a single integer comparison.

Flat output, no nested objects. Tools return shallow dictionaries; cross-entity links are by ORCID iD or DOI, not by reference. This keeps the LLM-facing surface predictable and easy to reason over.

5 The disambiguation tool

`orcid_disambiguate` is the differentiator. It wraps the ORCID name search and adds, for each hit, a score: the fraction of tokens in the affiliation hint that also appear in the candidate's listed affiliation string returned by the registry.

5.1 Design rationale

We considered three alternatives and rejected each.

1. Embed-and-cosine. Embed the affiliation hint and each candidate's affiliation through a sentence-transformer model and rank by cosine similarity. *Pros:* captures semantic proximity (*University of Madrid* matches *UAM* or *UCM*). *Cons:* requires a sentence-transformer dependency, model-version drift makes the score non-reproducible, and the model adds 200–500 ms of cold-start latency per call. The simpler heuristic is good enough in the common case (the candidate's known affiliation also appears literally on the ORCID record).

2. Fixed taxonomy of institution acronyms. Maintain a hand-curated table mapping common university acronyms to their canonical names (*UNED* → *Universidad Nacional de Educación a Distancia*, etc.) and use it to normalise both sides of the comparison. *Pros:* robust to common abbreviations. *Cons:* infinite long tail (joint institutes, affiliated hospitals, recently-renamed universities, country-coded variants); maintenance burden is high; institutions reorganise faster than the table can be updated.

3. Token overlap. (Selected.) Tokenise both strings on word boundaries, lowercase, take the intersection, divide by the size of the hint. The score is the fraction of hint tokens present in the candidate's listed affiliation. *Pros:* reproducible (no model dependency, deterministic), fast (microseconds per pair), good enough for the common case when the registry returns a non-empty affiliation. *Cons:* fails when the candidate's primary affiliation field is empty (an extremely common failure mode, see Section 7), or when the registry stores a name that does not lexically overlap the hint (*CIFAR* vs. *Montreal*).

The selected design is deliberately weak. It is a *ranker*, not a *classifier*: the score is a signal to the consuming agent, not a final answer. An agent that finds the top hit has score 0.0 (because the candidate stores the affiliation under a different name, or under no name at all) can decide to fall back to a wider search, fetch the candidate's employments and try matching against the longer institution names there, or consult a human. Baking a fixed accept/reject threshold into the server would mask the failure case; surfacing the score as a number lets the agent reason about it.

5.2 Multi-affiliation researchers

Many researchers have multi-affiliated records: a primary employment, a visiting position, an honorary appointment, a foundation membership. The Bengio case illustrates this clearly:

the most common affiliation listed on his ORCID record is *CIFAR* (his Turing-Award-related membership), not the Université de Montréal where he holds his primary appointment. A name search returns the right ORCID iD; a token-overlap score against the hint *Montreal* returns 0.0; the agent receives a signal that the hit is name-correct but affiliation-mismatched and can decide accordingly. This is the right behaviour: the disambiguation tool surfaces evidence, the agent applies policy.

5.3 Cold-start latency

The disambiguation pipeline is one search call (`/expanded-search` with the structured name query) plus client-side scoring. Cold-start latency from a fresh client without credentials is roughly 500–800 ms wall-clock (dominated by the round-trip and ORCID's anonymous-pool throttle). With credentials, it drops to 150–300 ms. The cache absorbs subsequent calls within the TTL window.

6 The `/search` versus `/expanded-search` story

A non-trivial v0.1 finding deserves explicit documentation. The first iteration of `orcid-mcp` used the plain `/search` endpoint at `pub.orcid.org/v3.0/search`. Offline tests passed; live smoke testing returned hits without given names, family names, or affiliations.

The cause was structural: the plain `/search` endpoint returns only *ORCID iD pointers*. The full hit shape is

```
{
  "result": [
    {
      "orcid-identifier": {
        "uri": "https://orcid.org/0000-0001-9084-8782",
        "path": "0000-0001-9084-8782",
        "host": "orcid.org"
      }
    }
  ],
  "num-found": 1771
}
```

To resolve each iD to a name and affiliation, the consumer would have to issue a second `/{orcid}/person` call per hit. For a 100-hit search, this is a hundred round-trips.

ORCID exposes a richer endpoint, `/expanded-search`, that returns the same hits with given/family names and the primary listed institution inlined:

```
{
  "expanded-result": [
    {
      "orcid-id": "0000-0002-9322-3515",
      "given-names": "Yoshua",
      "family-names": "Bengio",
      "institution-name": ["CIFAR"]
    }
  ]
}
```

The released version of `orcid-mcp` uses `/expanded-search` as the primary path and falls back to `/search` when the expanded endpoint returns nothing (a defensive fallback in case ORCID changes the endpoint behaviour).

We document the failure here for the same reason we documented Crossref's `sort=posted` rejection in the predecessor report: live smoke testing is mandatory before any v0.1 release

of an MCP wrapper. Offline tests with mocked HTTP responses cannot catch endpoint-level shape mismatches; only a real call against the real upstream surfaces them. The corollary for downstream wrapper authors is straightforward: do not skip the smoke-test step. (As an aside, the meta-validation that drove us toward this benchmark was that the disambiguation tool resolved this report's own co-author from his name and the affiliation hint *OpenCódice*, in two API calls and well under one second; a working anecdote, but not the deeper question of how the tool behaves at population scale.)

7 Benchmark: twelve senior CS researchers

The interesting question is not whether the tool resolves a known co-author. The interesting question is what the tool returns, and what it *fails* to return, when it is pointed at queries that an agent will actually run in the wild. To answer it we constructed a small but pointed benchmark on twelve senior computer-science researchers whose names, affiliations, and ORCID iDs are matters of public record.

7.1 Setup

We selected twelve well-known senior CS researchers whose primary appointment is widely known and easy to specify in a single short token: Yoshua Bengio (Montreal), Geoffrey Hinton (Toronto), Yann LeCun (New York), Andrew Ng (Stanford), Fei-Fei Li (Stanford), Christopher Manning (Stanford), Percy Liang (Stanford), Leslie Lamport (Microsoft), Donald Knuth (Stanford), Jeffrey Ullman (Stanford), Sara Hooker (Cohere), and Margaret Mitchell (Hugging Face). For each researcher we ran

```
candidates = search.search_by_name(
    client,
    family_name=<surname>,
    given_names=<given names>,
    rows=5,
)
```

and recorded four quantities for the top hit: the returned ORCID iD, the value of the `institution-name` field on the `/expanded-search` response, the total number of hits returned (a proxy for ambiguity), and a boolean flag indicating whether the user-supplied affiliation hint appeared as a substring (case-insensitive) of the returned institution name. The benchmark therefore measures exactly what the disambiguation tool can see when it is asked to score candidates by token overlap against `/expanded-search` alone.

7.2 Results

Table 2 presents the per-researcher outcomes.

The headline number is the third column of the right-hand side. *Of twelve queries, ten return a null `institution-name`; one returns a non-matching name (Bengio: CIFAR vs. hint Montreal); one matches (Knuth: Stanford University vs. hint Stanford).* The match rate of token-overlap scoring against the institution-name field returned by `/expanded-search` is therefore $1/12 \approx 8\%$ on this set, and would be $0/12$ if the benchmark used only currently-active researchers (Knuth is emeritus and is the only one who maintains a verbose primary affiliation on his record).

The result is not a defect of `orcid-mcp`. It reflects a structural property of the upstream registry: the `/expanded-search` response surfaces the institution-name field that ORCID has chosen to inline into the search index, which is sparsely populated for many active researchers [Schnieders et al., 2022]. Active researchers often have multiple, structured employments behind their ORCID iD; ORCID does not inline them all into the search response.

Name	Hint	#Hits	Top hit institution-name	Match
Yoshua Bengio	Montreal	1	CIFAR	no
Geoffrey Hinton	Toronto	1	(null)	no
Yann LeCun	New York	2	Association for Computing Machinery	no
Andrew Ng	Stanford	5	(null)	no
Fei-Fei Li	Stanford	5	(null)	no
Christopher Manning	Stanford	5	(null)	no
Percy Liang	Stanford	1	(null)	no
Leslie Lamport	Microsoft	1	(null)	no
Donald Knuth	Stanford	1	Stanford University	yes
Jeffrey Ullman	Stanford	3	(null)	no
Sara Hooker	Cohere	1	(null)	no
Margaret Mitchell	Hugging Face	5	(null)	no

Table 2: `search_by_name` top-hit results for twelve senior CS researchers. The *Match* column is true iff the affiliation hint (case-insensitive) appears as a substring of the institution name returned by ORCID's `/expanded-search` endpoint. Ten of twelve top hits return a null `institution-name`; one (Bengio) returns a non-null name (CIFAR) that does not match the user's hint (*Montreal*); one (Knuth) returns a non-null name (Stanford University) that matches.

7.3 Implications for the v0.2 disambiguation pipeline

The benchmark identifies the right next step concretely. For each candidate returned by `/expanded-search` the v0.2 disambiguation pipeline should fall back to `orcid_get_employments(orcid_id)`, which returns the candidate's structured employment history (institution name, role, start year, end year). Token-overlap scoring against the union of all institution names in that history is dramatically more informative. As a quick check, we ran `orcid_get_employments` on the eleven candidates whose primary `institution-name` was either null or non-matching: the user-supplied hint matches at least one institution-name in the employment history for all eleven candidates whose top hit was the "correct" senior researcher in question. The two-call fallback (search, then employments) closes the gap.

The cost is one additional API call per candidate. With `rows=5` this is up to five extra calls per disambiguation; with the diskcache layer in `orcid-mcp` the calls are cached for twenty-four hours and re-issuing them is free. The latency on a cold cache rises from 500–800 ms to roughly 1.5–3 s with anonymous credentials, or 600 ms–1.2 s with a registered Public API client. The trade-off is favourable: a roughly 3× wall-clock penalty on the first disambiguation per candidate, in exchange for moving the institution-match rate from 8% to roughly the population fraction of researchers whose ORCID employment history is non-empty (around 65–70% in current CS coverage [Porter et al., 2025]).

7.4 The negative control

For completeness, we also confirm the negative behaviour. Searching for a deliberately implausible name (*Researcher McResearcherFace*) returns an empty list, not an error. Searching for a common name with a misleading affiliation hint (*John Smith* hinted as *Mars*) returns the same set of candidates as the unhinted search but with all scores at 0.0. The tool is well-behaved on these edge cases. The right interpretation of an all-zero score column is the same as an all-null `institution-name` column: the agent should fall back to the employment history before deciding, not pick the top hit by default.

8 Use cases

The disambiguation backbone enables several downstream workflows. We sketch four; each has been exercised at least once during the development of the OpenCódice academic-MCP family.

8.1 Citation deduplication

A `.bib` file with 200 entries authored by 50 unique researchers becomes a `.bib` file with 200 entries authored by 50 unique ORCID iDs. The agent walks the `author` fields, calls `orcid_search_by_name` for each, applies the v0.2 employment-fallback for the disambiguation step, and dedupes on the canonical iD. Citation count per researcher becomes well-defined; collaboration-network analysis becomes well-defined; co-author-graph queries become resolvable across heterogeneous sources.

8.2 Reviewer vetting

Workshop and journal chairs assign reviewers under time pressure. `orcid_search_by_name` plus `orcid_get_works` pulls the candidate's full publication history in two calls; the chair gets context their default tooling does not provide. Combined with research-integrity tooling, the chair can filter out candidates with a flagged history. Combined with `orcid_get_employments`, the chair can verify that a proposed reviewer is not a current colleague of any author of the manuscript under review.

8.3 Affiliation verification

A submitted manuscript claims that all corresponding authors are affiliated with a specific consortium. The agent walks the author list, resolves each to an ORCID iD, fetches `orcid_get_employments`, and checks that each researcher's current employment matches the claimed affiliation. A mismatch surfaces a question for the human editor, not a hard rejection.

8.4 The disambiguation backbone for the rest of the family

A multi-MCP agent that pulls author handles from one server (an OpenReview profile, a DBLP author page, a Zenodo creator field) needs a final ORCID resolution step to reason about identity across them. The pattern is: pull the entity-specific identifier from the relevant server, then resolve to an ORCID iD through `orcid-mcp` for cross-server identity. The disambiguation tool, once the v0.2 employment-fallback is in, is the right primitive for this last step.

9 Limitations

Four limitations of the v0.1 release deserve explicit mention. The first is the one the benchmark surfaces; the others are inherent to the registry.

Sparse `institution-name` in the search response. The benchmark in Section 7 shows the cost: token-overlap scoring on `/expanded-search` alone matches only 1/12 of well-known queries. The v0.2 fallback to `orcid_get_employments` closes the gap and is the highest-priority follow-up.

Self-attestation lag. ORCID records are self-attested. A researcher's ORCID record may lag their actual employments by months or years; the registry is canonical for *identity* but not necessarily *currency* [Schnieders et al., 2022]. Workflows that depend on up-to-the-minute affiliation data should pair ORCID with another source (the publisher's submission record, the institution's own staff directory).

Coverage gaps. ORCID coverage varies by discipline and geography [Porter et al., 2025, Schnieders et al., 2022]. Computer science is well covered (most CS conference submission systems require an ORCID iD); some humanities disciplines are not. Workflows that need a

specific researcher whose ORCID is not registered will fail; `orcid-mcp` cannot conjure an iD that does not exist.

Anonymous-pool rate limits. The public API rate-limits aggressively for unauthenticated clients. v0.1 of `orcid-mcp` ships with diskcache TTLs and a polite User-Agent string but no request-throttling layer; high-volume callers should configure `ORCID_CLIENT_ID` / `ORCID_CLIENT_SECRET`. A request-throttling layer is on the v0.1.1 runway.

10 Discussion

`orcid-mcp` is the third member of the OpenCódice academic-MCP family. Its strategic role is the disambiguation backbone for the rest of the family: cross-server identity resolution between OpenReview profiles, DBLP author pages, and Zenodo creator searches all benefit from a final ORCID resolution step. The pattern (start with an entity-specific identifier from a domain server, resolve to a canonical ORCID iD for cross-domain identity) is the right composition pattern for a multi-MCP agent.

A more general lesson runs through this report. ORCID solves identity by self-attestation rather than by inference. Self-attestation is weaker than curation (DBLP's hand-curated PIDs are more accurate within their domain) but stronger than algorithmic clustering (Semantic Scholar's author clusters routinely merge or split between releases) [Färber and Ao, 2022, Sanyal et al., 2021]. For an LLM agent that needs to reason about *the same person across many systems*, self-attestation is the right primitive, because it is the only one with a stable identifier under the researcher's own control. ORCID is the substrate; `orcid-mcp` is the last mile.

The benchmark in Section 7 sharpens that lesson. Self-attestation is the right primitive for *identity*, but the inlined `institution-name` field that ORCID surfaces in its primary search response is not the right primitive for *affiliation matching*. The richer per-researcher employment history is. Building disambiguation on the wrong field appears to work in development (single-co-author tests pass) and fails on the population (10 of 12 well-known senior CS researchers have a null `institution-name` in the search response). The remedy is a one-call-deeper pipeline. The remedy is on the v0.2 runway.

Availability

Source code, tests, and full API documentation: https://github.com/OpenCodice-Research/orcid-mcp. License: MIT. Data source: ORCID Public API.

References

Michael Färber and Lin Ao. The Microsoft Academic Knowledge Graph enhanced: Author name disambiguation, publication classification, and embeddings. *Quantitative Science Studies*, 3(1): 51–98, 2022. doi:10.1162/qss_a_00183.

Anderson A. Ferreira, Marcos André Gonçalves, and Alberto H. F. Laender. A brief survey of automatic methods for author name disambiguation. *ACM SIGMOD Record*, 41(2):15–26, 2012. doi:10.1145/2350036.2350040.

Ilya Gusev. Academia MCP: A multi-source academic search server for the model context protocol, 2025. URL https://github.com/IlyaGusev/academia_mcp.

Laurel L. Haak, Martin Fenner, Laura A. D. Paglione, Ed Pentz, and Howard Ratner. ORCID: A system to uniquely identify researchers. *Learned Publishing*, 25(4):259–264, 2012. doi:10.1087/20120404.

ORCID, Inc. ORCID: Connecting research and researchers, 2025. URL https://orcid.org/.

Stephen R. Porter, Paul D. Umbach, and Jonathan Willis. Understanding ORCID adoption among academic researchers. *Scientometrics*, 130(5):2783–2797, 2025. doi:10.1007/s11192-025-05300-7.

Debarshi Kumar Sanyal, Plaban Kumar Bhowmick, and Partha Pratim Das. A review of author name disambiguation techniques for the PubMed bibliographic database. *Journal of Information Science*, 47(2):227–254, 2021. doi:10.1177/0165551519888605.

Henning Schnieders, Sandra Mierz, Sara Boccalini, Hendrik Meyer zu Westerhausen, Christian Hauschke, Stephanie Hagemann-Wilholt, and Francesca Schulze. ORCID coverage in research institutions: Readiness for partially automated research reporting. *Frontiers in Research Metrics and Analytics*, 7:1010504, 2022. doi:10.3389/frma.2022.1010504.

Neil R. Smalheiser and Vetle I. Torvik. Author name disambiguation. *Annual Review of Information Science and Technology*, 43(1):1–43, 2009. doi:10.1002/aris.2009.1440430113.

Jie Tang, Alvis C. M. Fong, Bo Wang, and Jing Zhang. A unified probabilistic framework for name disambiguation in digital library. *IEEE Transactions on Knowledge and Data Engineering*, 24 (6):975–987, 2012. doi:10.1109/TKDE.2011.13.

OpenCódice Technical Report OC-TR-2026-009 • 2026 • OpenCódice Research
opencodice.org • DOI: 10.5281/zenodo.20023535