

Linking a local record to a Wikidata QID looks simple right up until the point where it is not. A museum catalog entry with a tidy label, a person record from an authority file, a place name in a legacy spreadsheet, a company in a vendor database, they all seem easy when the name is distinctive and the metadata is clean. Then the real work starts. Names collide. Transliteration varies. Dates are partial. Labels drift across languages. Two entities share a profession, a location, and a near-identical title, but only one is the right match.

That is where a disciplined resolution workflow matters more than raw search. The value is not just finding a candidate. The value is deciding, with evidence you can inspect later, whether a local record should be linked at all.

A useful development in that space is the open source project often described as Wikidata + Google Knowledge Graph MCP. It is an MCP server and CLI designed to help agents search Wikidata, read selected facts, and link local records to Wikidata QIDs with explicit outcomes and visible uncertainty when the evidence does not support an automatic match. If you have spent time cleaning identifiers at scale, that design choice stands out immediately. Most mistakes in entity linking happen when systems are too eager to be helpful.

What this MCP server actually solves

The project sits in a practical middle ground. It is not a general purpose graph browser, and it is not trying to replace human review for every ambiguous record. It gives an agent a constrained set of tools to search, inspect facts, and resolve identity in a way that remains understandable to the operator.

That matters because local record linking tends to fail in two opposite ways. Some workflows produce huge, noisy candidate sets and leave the user to sift through them manually. Others over-automate and hide the evidence behind a confidence score that nobody can audit later. This MCP for Wikidata takes a more conservative path. It emphasizes bounded search, selected facts, deterministic outcomes, and optional cross-checks rather than unlimited expansion.

The result is a workflow that feels closer to professional cataloging practice than to broad web search. You ask for a candidate set, examine a small number of plausible entities, compare known facts, and either assign a QID or hold the record for review. That rhythm is familiar to anyone who has **read more** done authority control or master data stewardship in production.

Wikidata's own broader MCP context is also relevant here. Wikidata documents an MCP that gives language models standardized tools to explore and query Wikidata programmatically through the Wikidata API and the Query Service. This project narrows the focus to identity resolution and evidence handling. That narrower scope is a strength.

Why bounded search is more important than it sounds

One detail from the project documentation deserves more attention than it usually gets: by default, search returns three candidates, with a maximum of five. On paper that can sound limiting. In practice, it is often the right choice.

Large candidate sets create a false sense of completeness. They encourage lazy ranking logic and push the burden of decision-making onto whichever person or agent comes next. If you have ever watched a team

export fifty search results per name into a spreadsheet, you know how quickly that becomes a graveyard of unresolved records.

Bounded search forces discrimination early. If a query cannot produce a short list of plausible entities, that is useful information. It often means the source record needs normalization, that the entity is poorly represented, or that the query is too broad to support safe matching. Any of those cases is better surfaced immediately than buried under twenty marginally relevant results.

For example, suppose you are linking a local author record labeled "A. Rahman" with no birth year and a note that the person wrote on education policy in Dhaka. An unconstrained search can flood you with plausible people. A bounded search that favors the best few matches gives you something more actionable. You can inspect whether the candidates have occupation statements, language variants, associated places, and references that support the local description. If not, the correct outcome may be to hold the record, not to guess.

This is one reason the project's explicit uncertainty is important. In real data work, "I do not know yet" is often the most accurate answer available.

The tools that make the workflow usable

The documented MCP tools are straightforward: `kg_search`, `kg_entity`, `kg_related`, `kg_resolve`, and `kg_status`. There is also a CLI with batch and evidence export commands. That combination is practical because entity linking is rarely just an interactive task. Teams usually need both modes. They want a record-by-record workflow while tuning rules, then batch handling once they trust the process enough to scale.

`kg_search` is the obvious entry point. It finds candidate entities. On its own, that is only mildly interesting. The more useful follow-up is `kg_entity`, which supports retrieval of selected facts, including ranks, qualifiers, and references when requested. That phrase, "selected facts," is easy to skim past, but it points to a sensible discipline. Good matching depends less on collecting every statement than on inspecting the few statements that disambiguate identity.

A person record may hinge on birth date, occupation, country of citizenship, field of work, or a notable affiliation. A place record may hinge on instance of, administrative parent, coordinates, and language variants. A work record might turn on publication date and creator relationships. Pulling those facts with ranks and references available lets an operator judge not only whether two records look similar, but whether the knowledge base is stating the right thing with enough structure to support a link.

`kg_related` can help in the situations where direct labels are not enough. If two organizations share a name, related entities may expose a parent body, region, or associated person that distinguishes them. Used carefully, relation context can turn a weak lexical match into a strong identity claim. Used carelessly, it can also create confirmation bias, so the fact that this project keeps the workflow inspectable is reassuring.

The core tool, though, is `kg_resolve`. Deterministic outcomes such as `AUTO_MATCH`, `HOLD`, `AMBIGUOUS`, and `NO_CANDIDATE` are exactly the kinds of states a production pipeline needs. They are not flashy, but they travel well. A data steward can understand them. A logging system can report on them. A downstream review queue can act on them. Most important, they do not pretend that every record deserves a single score on a fuzzy numeric scale.

Deterministic outcomes beat vague confidence scores

Confidence scores have their place, but they are often abused in record linkage. A score of 0.84 may look scientific while saying very little about why a record matched or why it should be trusted. Scores also invite endless threshold tuning, which can become a substitute for understanding the edge cases.

Explicit categories are often better operational tools. A record marked `AUTO_MATCH` can be reviewed under one policy. A record marked `AMBIGUOUS` can be sent to a specialist queue. A record marked `NO_CANDIDATE` may trigger source cleanup or a later retry. `HOLD` is especially valuable because it recognizes a common reality: the candidate space may exist, but the available evidence is not enough to decide safely.

I have seen teams waste months trying to squeeze every record into matched or unmatched, because their systems had no stable state for uncertainty. Once uncertainty becomes a first-class outcome, the quality of the entire workflow improves. Reviewers stop fighting the machine. Analysts can track why records are stalling. Product owners can quantify how much ambiguity comes from poor local metadata versus sparse external representation.

That same logic is part of why MCP for google knowledge graph and wikidata is an interesting combination here. The system does not claim that adding another provider magically proves identity. Instead, it treats cross-provider agreement as concordance, useful but not definitive. That is the right call.

What the Google cross-check is, and what it is not

The project supports an optional Google cross-check through exact identifier joins. Specifically, it documents joins using `/m/` for Wikidata property P646 and `/g/` for P2671. This is a very different proposition from scraping broad search results or treating a search engine snippet as evidence. It is a structured concordance check based on known identifier mappings.

That distinction matters. There is a temptation in entity resolution to treat agreement between two systems as proof of truth. It is not. Two providers can agree because they share a common source, because one imported from the other, or because the same historical mistake propagated across the web. Provider agreement can strengthen a case, but it does not close it by itself.

The project is explicit about that. The Google and Wikidata agreement is treated as concordance rather than proof of identity. If you have worked with authority files, this is exactly the kind of modesty you want from a linking tool.

There is also a practical advantage. The Google Knowledge Graph Search API is optional. Wikidata does not require an account or an API key for this workflow, which lowers the barrier for testing and for controlled internal pilots. Teams can begin with a pure Wikidata flow and add the cross-check only where it helps. That staged adoption is sensible, especially in organizations where procurement or credential management slows down experimentation.

This is where the phrasing “MCP for google knowledge graph” can mislead people if they are not careful. The project is not an export of the Google Knowledge Graph, and it is not official software from Google or Wikimedia. It is a read-only server that helps an agent use available services responsibly. That modest scope is worth preserving, because overclaiming provenance is one of the fastest ways to erode trust in linked data work.

A realistic linking scenario

Imagine a local cultural heritage database with 40,000 creator records accumulated over twenty years. Labels come from many sources. Some entries contain full dates, some only decades, some just a city. A few include internal notes like “worked with municipal theater, 1980s.” The team wants to add Wikidata QIDs to improve interoperability, but they cannot afford a fully manual review of every record.

A workflow using this MCP for Wikidata would begin by normalizing labels and pulling a short candidate list per record. For records with strong metadata, `kg_resolve` could likely identify clear `AUTO_MATCH` cases. A person named “Marina Volkova,” born in 1964, occupation sculptor, active in Riga, is much easier to distinguish than a record named “J. Kim” with no dates. Selected-fact retrieval would surface the disambiguating statements, and evidence export would preserve the reasoning.

The difficult cases are where the design earns its keep. Suppose a local record says “St. Michael’s School, founded 1912, Bristol,” but multiple institutions share that name and some changed governance structures over time. A broad search may find several plausible candidates. If none aligns clearly on location and institutional type, `AMBIGUOUS` or `HOLD` is the correct outcome. That is not failure. It is quality control.

Batch processing is especially useful in a collection like this. Rather than asking staff to click around record by record from day one, they can run batches, inspect evidence exports, and study the distribution of outcomes. If seventy percent of records are stable `AUTO_MATCH` cases, the project has already paid for itself. If only fifteen percent are automatic and half the remainder are `NO_CANDIDATE`, that is also valuable, because it shows where local metadata or source coverage needs work.

Where evidence makes or breaks trust

In linking projects, people usually talk about precision and recall. Those matter, but the day-to-day trust of a system often comes down to evidence. A match that is correct but opaque tends to be rejected by careful users. A match that is slightly uncertain but well-explained may still be accepted for review or conditional use.

This project’s support for selected facts, plus ranks, qualifiers, and references on request, is one of its most mature aspects. Wikidata statements often need that context. A raw property-value pair can look decisive until you discover it is deprecated, qualified to a narrow timeframe, or lightly sourced compared with a competing statement. Any serious matching workflow should be able to see those distinctions.

Take a city entity with multiple names across languages and periods. The right local match may depend on whether a label is historical, official, colloquial, or tied to a specific administrative era. Qualifiers and ranks help keep those cases from collapsing into simplistic string equality. The same is true for people with contested dates or organizations that merged and split.

Evidence export from the CLI is important for another reason: institutional memory. Teams change. Contractors leave. Six months after a batch run, someone will ask why record 18,742 was linked to a particular QID. If the answer is buried in transient prompts or a hidden scoring model, the audit trail is gone. If the answer can be reconstructed from exported evidence and explicit outcomes, the work remains governable.

Trade-offs you should understand before adopting it

No linking tool is a silver bullet, and conservative tools in particular come with trade-offs. This one makes choices that many data professionals will like, but those choices imply certain limits.

First, bounded search improves focus, but it can miss a correct entity if the query itself is weak. That is not necessarily a defect. It simply means preprocessing and query formulation still matter. Garbage labels will not become reliable identifiers because the search space is small.

Second, deterministic outcomes improve operations, but they do not erase the judgment required in edge cases. The system can tell you that a record is ambiguous. It cannot decide your institution's tolerance for linking historical organizations through successor entities, or how much local evidence is enough to support a person match with sparse public data.

Third, provider concordance is useful, but only within its documented role. If your team starts treating the optional Google cross-check as final proof, you will eventually encode bad links with a false sense of security.

A sensible adoption checklist fits on one screen:

1. Decide which local fields are truly reliable enough to drive matching.
2. Define what AUTO_MATCH means for your organization before running batches.
3. Review a sample of HOLD and AMBIGUOUS cases to understand failure modes.
4. Preserve evidence exports so linked records remain auditable.
5. Treat Google concordance as support, not as identity proof.

That sequence may feel cautious, but caution is cheaper than remediation once bad QIDs have propagated into downstream systems.

How this fits into MCP workflows more broadly

There is a wider trend here that is worth noting. MCP tools are most useful when they expose constrained, inspectable actions rather than pretending to give an agent unlimited understanding. Wikidata is a strong candidate for that approach because it has rich structure, broad topical coverage, and a well-established identifier culture. But the very richness of Wikidata can overwhelm generic interfaces.

By narrowing the interaction to search, entity inspection, related-entity context, status checks, and deterministic resolution, this project gives agents a smaller surface area with clearer expectations. That is healthier than handing an agent a massive endpoint and hoping prompt engineering supplies the guardrails.

It also helps that the server is read-only and explicitly does not edit Wikidata, Google, or user data. For many organizations, the hardest part of evaluating a new tool is not functionality but risk. A read-only model removes an entire class of concerns. You can test linking logic without worrying that exploratory sessions will mutate shared knowledge bases.

The fact that it is open source and MIT licensed also matters in practical ways, even if licensing rarely excites non-specialists. Linked data projects tend to live longer than the first implementation team. Being able to inspect the workflow, understand its deterministic states, and integrate it into local governance is often more important than whether the interface feels polished in week one.

When not to link automatically

One of the healthiest habits in QID assignment is knowing when not to assign one. In many institutions, the pressure to “enrich all records” leads to brittle automation. The result is a database full of links that look authoritative and quietly create downstream confusion.

A few situations deserve special restraint:

- records with generic names and no independent dates or locations
- entities that change identity over time, such as institutions after mergers or reorganizations
- historical places where the local concept does not map cleanly to a present-day Wikidata item
- works and editions that blur between abstract work, expression, and publication
- local records whose own metadata is internally inconsistent

These are not fringe cases. They are common in archives, publishing, higher education, and enterprise master data. A system that can say HOLD cleanly is more useful than one that insists on closure.

The quiet value of a conservative tool

A lot of software in the knowledge graph space tries to impress with breadth. It can search everything, rank everything, infer everything. That style demos well. It is less satisfying six months into a production rollout, when someone has to explain why hundreds of records were linked on flimsy evidence.

What makes this project notable is its restraint. It searches Wikidata, optionally cross-checks against Google Knowledge Graph identifiers, retrieves selected facts with the context needed for judgment, and resolves records into explicit operational states. It does not claim to prove identity through mere agreement. It does not claim to be official software from the underlying providers. It does not write back to the sources. Those limits are not weaknesses. They are part of why the workflow can be trusted.

For teams exploring MCP for wikidata in a serious linking context, that should be appealing. The hard part of entity resolution has never been getting a candidate name back from a search box. The hard part is making a decision that another person can review, defend, and maintain over time. This project appears to understand that distinction.

If you are responsible for authority control, metadata interoperability, or record linkage in any domain where false matches carry real cost, that design philosophy is worth paying attention to. The local record does not need a flashy guess. It needs the right QID, or an honest admission that the evidence is not there yet.