

When people talk about entity resolution against Wikidata, they often jump straight to the happy path: search a name, grab a QID, move on. In practice, that is where mistakes start. The hard part is rarely finding a candidate. The hard part is deciding whether the candidate is good enough to trust, whether the evidence is thin, and whether uncertainty should stop the workflow.

That is why candidate evidence matters so much in the emerging tooling around MCP for Wikidata. In the case of the open source project commonly described as Wikidata + Google Knowledge Graph MCP, the design is not just about retrieval. It is about returning a small, inspectable set of possibilities, exposing selected facts, and making room for explicit doubt when the evidence does not support a confident match.

If you work with records, catalogs, content pipelines, research assistants, or any agent that needs to connect local data to Wikidata, this distinction is more than architectural taste. It affects precision, review effort, and the cost of bad links. A false positive can quietly pollute downstream systems for months. A cautious hold, by contrast, may slow things down, but it preserves trust.

What “candidate evidence” actually means here

In this MCP context, candidate evidence is the package of signals that helps an agent or human reviewer evaluate whether a proposed Wikidata entity matches a local record. It is not a vague confidence number floating in space. It is closer to a case file.

The project’s documented behavior makes that plain. It lets an agent search Wikidata, inspect selected facts, and link local records to Wikidata QIDs with evidence that can be examined. Just as important, it makes uncertainty explicit when evidence is insufficient. That last piece is easy to overlook, but it is the one that separates careful resolution from overconfident automation.

A useful mental model is to think of each candidate as a hypothesis. The search tool does not hand you truth. It hands you a short list of plausible hypotheses. Then the evidence tools help test them.

That sounds almost obvious until you compare it with systems that flood a model with dozens of search hits and leave it to improvise. Large result sets often create the illusion of completeness while making judgment worse. This project takes the opposite approach. It uses bounded search by default, returning three candidates and allowing up to five, rather than dumping a broad raw result set into the agent’s context. That design choice is small on paper and significant in use.

I have seen enough resolution work to know that fewer, better candidates usually beat wider, noisier recall when a human or agent must justify a decision. Once the result set grows too much, reviewers stop evaluating and start skimming. Models do something similar. They anchor on the first familiar label and then rationalize.

Why bounded search improves evidence quality

Bounded search is one of the most practical ideas in this MCP for Wikidata setup. By keeping the candidate set small, the tool encourages comparative reasoning. A reviewer can actually read the returned candidates, compare labels, and check distinguishing facts. An agent can do the same without wasting context window on twenty weak maybes.

That has two immediate effects.

First, it changes the shape of the decision. Instead of asking, “Did anything in these fifty results look right?” you ask, “Among these three to five plausible entities, does one clearly fit the record better than the others?” That is a much healthier question.

Second, it exposes absence more honestly. If the system returns no solid candidate, or only a few weak ones, the workflow can surface that ambiguity rather than papering over it. In entity resolution, a visible “no good match” is usually more valuable than a hidden wrong match.

The project’s documented outcomes reinforce this discipline. Rather than reducing everything to a single opaque score, it uses deterministic resolution logic with explicit states such as `AUTO_MATCH`, `HOLD`, `AMBIGUOUS`, and `NO_CANDIDATE`. Those labels matter because they express operational intent. `AUTO_MATCH` means the evidence passed a clear threshold in the system’s logic. `HOLD` tells you the evidence is not enough for automatic linking. `AMBIGUOUS` signals that multiple candidates remain plausible. `NO_CANDIDATE` is exactly what it sounds like.

That vocabulary keeps people honest. It also helps build review queues that behave sensibly. A team can accept auto matches, manually inspect holds, route ambiguous records to specialists, and leave no-candidate cases for later enrichment. Without those explicit buckets, every uncertain case tends to collapse into a weak yes or a vague maybe.

Evidence is only useful if you can inspect it

A candidate record becomes meaningful when you can read facts that actually distinguish entities. The project supports selected-fact retrieval, and it can include ranks, qualifiers, and references on request. That detail deserves more attention than it usually gets.

Anyone who has worked with Wikidata knows that facts are not all equal. A statement may be preferred, normal, or deprecated. Dates may need qualifiers. Names may vary by language or context. References may support a statement or reveal that it is lightly sourced. If a system hides those distinctions, it flattens the graph into something easier to consume and less safe to trust.

Consider a simple but common edge case: two people share the same name. A plain label match is nearly worthless. The ability to inspect selected facts, and especially the surrounding qualifiers or references when needed, is what lets a reviewer move from “looks familiar” to “this is probably the same person.” The same applies to organizations with similar names, creative works with reused titles, and places that changed names over time.

You do not need every statement on the entity for that process. In fact, too much data can blur judgment. What you need are the right facts, exposed with enough structure to preserve meaning. That is where selected-fact retrieval shines. It narrows the evidence without stripping out the parts that explain why a fact should or should not carry weight.

This is one reason the term “candidate evidence” is more precise than “candidate data.” Evidence implies relevance to a decision. Data just implies presence.

The role of Google Knowledge Graph, and the limits of concordance

The Google side of this project is optional, which is a sensible design choice. Wikidata can be used without an account or API key, while the Google Knowledge Graph Search API is an extra layer that can be brought in when useful. That means the setup can stay lightweight for teams that only need Wikidata, yet still support a broader cross-checking workflow when Google data is available.

This is where the phrase MCP for google knowledge graph and wikidata becomes practical rather than promotional. The value is not that one source magically validates the other. The value is that they can be compared in a controlled way.

The project documents a specific cross-check based on exact identifier joins: /m/ joined to Wikidata property P646, and /g/ joined to P2671. That exactness matters. It is not doing a loose text comparison and pretending it proved identity. It is looking for a documented identifier relationship.

Just as important, the project explicitly treats agreement between Google and Wikidata as provider concordance, not proof of identity. That is exactly the right stance. Concordance tells you that two providers line up on an identifier path. It does not tell you that the original local record was interpreted correctly, that the entity boundaries match your use case, or that the source data is free of mistakes.

I like this distinction because it avoids a common trap. Teams often act as if two external systems agreeing means the problem is solved. In reality, two [Wikidata MCP](#) systems can repeat the same confusion, especially around namesakes, merged concepts, or entities that changed over time. Provider agreement is helpful evidence. It is not a substitute for judgment.

If you are exploring MCP for google knowledge graph in relation to knowledge resolution, that one design principle is worth carrying into every implementation: cross-source agreement strengthens a case, but it never absolves you from checking what the identifiers actually refer to.

Deterministic outcomes are better than theatrical certainty

One of the most useful features in this MCP for Wikidata project is its deterministic resolution logic. “Deterministic” can sound dry, but in day-to-day operations it is often the difference between a reviewable system and a mysterious one.

With explicit outcomes like AUTO_MATCH, HOLD, AMBIGUOUS, and NO_CANDIDATE, the server does not force every case into a fake confidence continuum. It gives the workflow states that can be acted on. That seems modest until you compare it with systems that emit a score like 0.87 and leave everyone to argue about what 0.87 means. Scores have their place, but if the underlying evidence is not visible and the consequences are not explicit, the number becomes decorative.

A deterministic outcome framework also makes testing easier. Suppose a batch of local records includes many abbreviated organization names. If too many fall into AMBIGUOUS, that tells you something operationally useful: perhaps your input records need more context, or your selected facts are not capturing the right differentiators. If weak matches slip into AUTO_MATCH, that reveals a more serious calibration issue. Either way, the categories support diagnosis.

There is another benefit that shows up in production environments. Staff turnover is real. People inherit pipelines they did not build. Clear resolution states survive handoffs better than a web of undocumented heuristics. A junior reviewer can understand what “hold for review” means far faster than they can reverse-engineer why the previous system accepted one 0.74 score and rejected another.

How the MCP toolset supports evidence-driven resolution

The documented MCP tools are straightforward: kg_search, kg_entity, kg_related, kg_resolve, and kg_status. The CLI extends that with batch and evidence-export commands. Even without diving into implementation details beyond what is documented, you can already see the intended rhythm of use.

A typical workflow looks like this:

1. Search for plausible entities with `kg_search`.
2. Inspect a candidate more closely with `kg_entity`.
3. Use `kg_resolve` to determine the resolution outcome.
4. Export evidence when you need an audit trail or review packet.

That sequence reveals a lot about the project's philosophy. Search is not the endpoint. Entity inspection is not an afterthought. Resolution is a formal step. Evidence export acknowledges that many linking tasks need traceability, not just a final answer.

For teams handling records in volume, the CLI matters just as much as the MCP server itself. Batch operations and evidence export suggest a path from ad hoc lookup to repeatable review. That can be the difference between a proof of concept and something a data team can live with.

I would also call out `kg_status` as more important than it sounds. In distributed toolchains, simple status visibility saves time. When a resolver behaves oddly, people need to know whether the service is reachable and healthy before they start blaming the record, the prompt, or the model.

Candidate evidence is most valuable when the case is messy

The easy matches are not where a resolver proves itself. Any system can look good on globally famous entities or highly distinctive names. The interesting cases are the ones that make experienced reviewers pause.

Take a local record with a common person name and thin metadata. Search will likely produce multiple plausible QIDs. In a weak system, the agent may latch onto whichever candidate appears most prominent. In a stronger system, bounded search keeps the set manageable, selected facts expose disambiguating details, and deterministic outcomes let the resolver say "ambiguous" without embarrassment.

Or consider an organization that has rebranded. A local record may use the old name while the candidate entity reflects the current one. Pure string matching performs badly here. Evidence has to reach beyond the surface label. That is where selected facts and related-entity context can help, assuming the relevant distinctions are present in the underlying data.

Even no-candidate cases are productive when surfaced clearly. They often reveal that the local record lacks enough information, not that the resolver failed. I have seen workflows improve simply because a resolver stopped pretending and started saying, in effect, "I cannot support a safe link with what you gave me."

That kind of honesty is underrated.

What this project is, and what it very deliberately is not

It helps to be clear about boundaries. The project is read-only. It does not edit Wikidata, Google, or user data. It is not official Wikimedia or Google software. It is not an export of the Google Knowledge Graph. Those constraints are not **Wikidata MCP examples** minor legal footnotes. They shape how you should think about candidate evidence.

Because the server is read-only, its job is to support discovery, inspection, and resolution, not curation. That means the evidence you see reflects what the external systems expose, filtered through the tool's retrieval

logic. If a Wikidata item is sparse, contradictory, or missing a fact you hoped to use, the server is not there to repair it. Your workflow must absorb that reality.

Because it is not an official product from Wikimedia or Google, you should also separate the tool from the authorities it queries or aligns with. The trust model is layered. You trust the resolver to expose candidates and evidence in a disciplined way. You still evaluate the underlying data on its own merits.

This may sound cautious, but it is the right kind of caution. Good resolution systems do not promise more certainty than the source material can bear.

Where this fits within broader Wikidata MCP work

Wikidata itself now documents a broader MCP context: standardized tools for LLMs to explore and query Wikidata programmatically through the Wikidata API and the Wikidata Query Service. That broader backdrop matters because it shows this project is not an isolated curiosity. It sits within a larger move toward making structured knowledge accessible to agent workflows in consistent ways.

Still, there is a notable difference in emphasis. General Wikidata MCP access is about exploration and querying. This particular project leans hard into entity resolution, bounded candidate handling, and inspectable evidence. That specialization is important. Query access alone does not solve the judgment problem of linking a local record to the right QID.

This is where the keyword phrase MCP for wikidata makes sense in a practical search sense, but the real distinction is narrower: some MCP tools help you fetch and ask questions, while this one is organized around the discipline of choosing among candidate entities and carrying evidence along with the choice.

That focus is what makes it useful for production linking scenarios rather than only exploratory browsing.

Trade-offs you should expect if you adopt this approach

No design comes free. Bounded search, deterministic outcomes, and inspectable evidence all improve safety, but they also create friction in places.

The most obvious trade-off is recall versus reviewability. Returning three candidates by default, and up to five, keeps comparison sane. It can also mean that a legitimate but less obvious entity never appears in the first pass. Whether that is acceptable depends on your domain. For many operational workflows, it is a fair bargain because review quality collapses when candidate lists get too large. Still, teams should recognize the trade.

Another trade-off is speed versus depth. Selected-fact retrieval is efficient when you know what matters, but there are cases where the decisive clue lives outside the first set of facts you inspect. A resolver can support evidence-driven work and still require a second round of checking in difficult records.

There is also the matter of user expectations. People sometimes want a single authoritative answer from tools like this. The presence of HOLD, AMBIGUOUS, and NO_CANDIDATE can initially feel like a failure if the organization is used to forced decisions. Over time, though, those outcomes tend to improve trust because they mark where the automation's confidence should stop.

The final trade-off is conceptual. The optional Google cross-check can be genuinely helpful, but only if teams internalize that concordance is not proof. If they do not, the extra signal can harden weak assumptions instead of correcting them.

Practical habits for reading candidate evidence well

The best results usually come from a consistent reading habit rather than a hunt for silver bullets. When I review candidate evidence, I am less interested in any single matching field than in whether the candidate forms a coherent identity under inspection. Labels, selected facts, rank, qualifiers, and references all contribute to that coherence.

A short discipline goes a long way:

- Treat the candidate set as competing hypotheses, not a menu of acceptable answers.
- Look for distinguishing facts, not just overlapping labels.
- Use Google concordance as support, never as standalone proof.
- Respect HOLD, AMBIGUOUS, and NO_CANDIDATE as valid outcomes.
- Keep evidence exports for any match that will matter later.

None of that is glamorous. It is simply how bad links are prevented.

Why this matters more as agents become routine

As MCP clients such as Claude Code, Cursor, and Codex become common places to run structured lookups, more entity resolution work will happen inside agent-driven flows rather than in standalone reconciliation interfaces. That shift raises the stakes for candidate evidence.

Agents are fast. They are also prone to smoothing over uncertainty if the tools around them do not make uncertainty legible. A resolver that returns a tidy QID without evidence can encourage overreach. A resolver that returns bounded candidates, inspectable facts, and explicit resolution states creates better behavior almost by default.

That is why the strongest part of this project is not any single endpoint. It is the discipline embedded in the whole setup: a small candidate set, read-only retrieval, selected facts with structural detail on request, optional cross-source concordance, and explicit outcomes when the case is not strong enough.

For anyone evaluating MCP for google knowledge graph and wikidata, that is the real standard to apply. Do not ask only whether a tool can find entities. Ask whether it helps an agent or reviewer *justify* a link, *defer* a weak one, and *explain* the difference.

That is what candidate evidence is for. It is not decoration around entity resolution. It is the part that makes the resolution worth trusting.