

Search & verification

Verification engine

🕒 3 min read · Updated 25 September 2026 · Verified against 7a343c6 on 2026-09-25

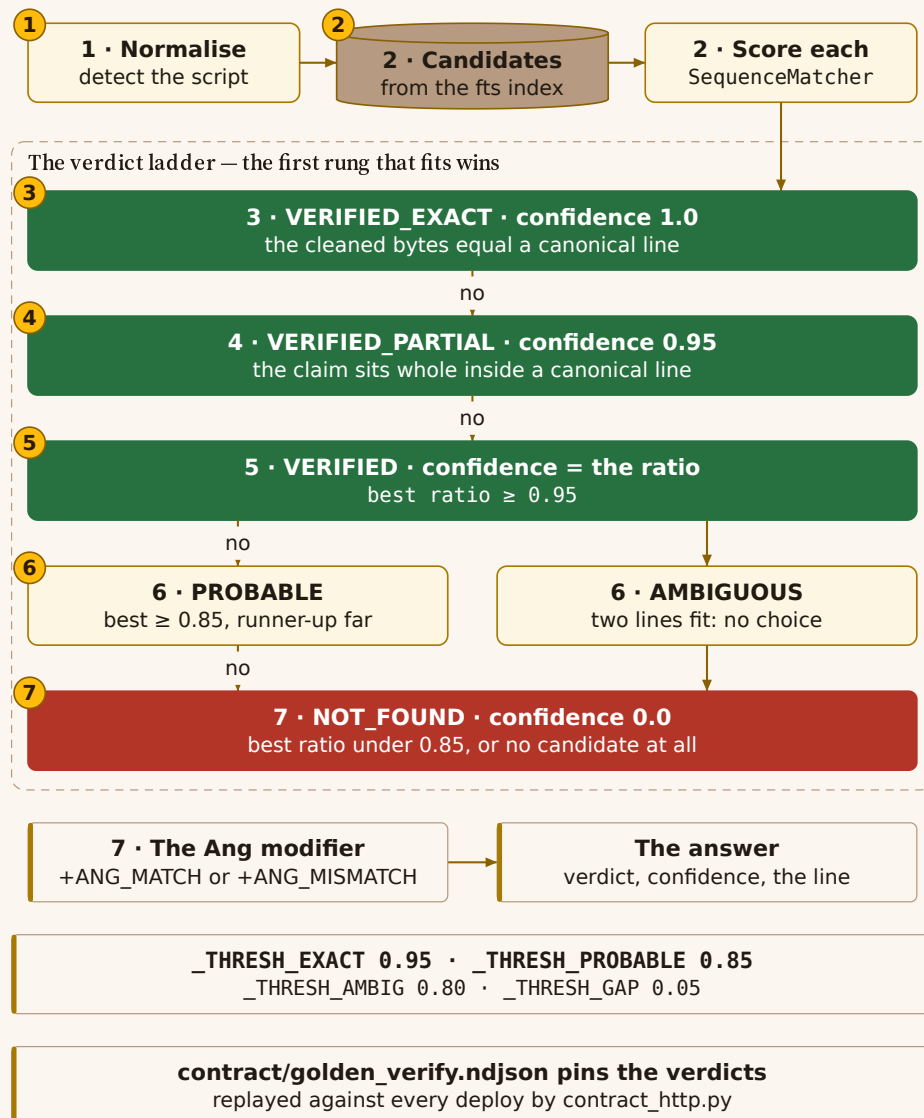
Is this quotation really in the Granth? How `/api/verify` normalises a claim, finds candidates, scores them and climbs the verdict ladder — the code's thresholds, and a playground over the live API.

<https://docs.gurbanisoul.com/search/verification-engine/>

`/api/verify?q=&ang=` answers one question: *is this quotation really in Sri Guru Granth Sahib Ji, and where?* It returns a **verdict**, a **confidence**, the **canonical line** with its Ang, and the scoring details. It never returns a line it is not sure about: below the thresholds the verdict is `NOT_FOUND` and no line is asserted. The engine is `webapp/verify.py`, `stdlib` only, importable and served by the `verify` bounded context.

07 · The verification engine

Is this quotation really in the Granth? /api/verify answers with a verdict, a confidence and the canonical line — never a guess



component in this project
 data at rest
 proven, verified state

not found: no line is asserted
 explanation

— flow: data or control moves this way

Source of truth: webapp/verify.py · webapp/romannorm.py · webapp/siggs/verification.py · contract/golden_verify.ndjson

v1.3.10 · verified 2026-09-27 · d85c2ad

Sri Guru Granth Sahib Ji — Knowledge Base · engineering wiki

Poster 07 — The verification engine: normalise, candidates, scoring, and the verdict ladder from VERIFIED_EXACT to NOT_FOUND with the Ang modifier

- 1. Normalise and detect the script.** The claim is trimmed and NFC-normalised. If it holds Gurmukhi, dandas, danda numerals and pipes are stripped to give the clean claim; if it is Roman, every word is folded with roman_norm — the same fold that built the translit_norm column — to give the claimkey.

2. **Candidates, then a score for each.** The FTS index is asked for an exact phrase first (the text column for Gurmukhi, translit_norm for Roman; the skeleton as a second chance), then for a looser OR of the tokens. Every candidate line is scored with difflib's SequenceMatcher ratio against the claim; the best, the runner-up and the gap between them decide the rung.
3. **VERIFIED_EXACT.** If the phrase query produced an exact hit — the cleaned bytes equal a canonical line — the verdict is VERIFIED_EXACT with confidence 1.0. Nothing fuzzy was needed.
4. **VERIFIED_PARTIAL.** People quote half a line constantly. If the claim is at least three words or twelve characters and sits whole inside a candidate line (but is not the whole line), the verdict is VERIFIED_PARTIAL with confidence 0.95 and that line is returned.
5. **VERIFIED.** A best ratio of at least 0.95 (`_THRESH_EXACT`) is VERIFIED; the confidence is the ratio itself. Small transliteration differences land here.
6. **PROBABLE or AMBIGUOUS.** A best ratio of at least 0.85 (`_THRESH_PROBABLE`) is PROBABLE — unless the runner-up is itself at least 0.80 (`_THRESH_AMBIG`) and within 0.05 (`_THRESH_GAP`), in which case two lines fit and the verdict is AMBIGUOUS: the engine reports both scores and refuses to choose.
7. **NOT_FOUND, the Ang modifier, the answer.** Below 0.85, or with no candidates at all, the verdict is NOT_FOUND with confidence 0.0 and no line is asserted — a percentage would only mislead. When the caller claimed an Ang and a line was matched, the verdict gains `+ANG_MATCH` or `+ANG_MISMATCH(actual=N)`. The response carries the verdict, the confidence, the matched line id, the canonical Gurmukhi with its Ang, raag, author, comp_id and section, and distance_details. `contract/golden_verify.ndjson` pins these verdicts across every deploy and the iOS port.

Try it

Interactive on the web: a quotation checked by the verification engine.

The ladder

The thresholds are module constants in `webapp/verify.py`; `tools/docs_check.py` fails the build if the numbers on the poster stop matching them.

Rung	Verdict	Condition	Confidence
1	—	NFC-normalise; detect the script. Gurmukhi: strip dandas, danda numerals, pipes. Roman: fold each word with <code>roman_norm</code>	—
2	—	candidates from <code>fts</code> : an exact phrase (text / skeleton for Gurmukhi, <code>translit_norm</code> for Roman), then a looser OR; each scored with <code>difflib.SequenceMatcher(...).ratio()</code>	—
3	VERIFIED_EXACT	the phrase query produced an exact hit (the cleaned bytes equal a line)	1.0
4	VERIFIED_PARTIAL	the claim (≥ 3 words or ≥ 12 characters) sits whole inside a candidate line, and is not the whole line	0.95
5	VERIFIED	best ratio ≥ 0.95 (<code>_THRESH_EXACT</code>)	the ratio
6	PROBABLE	best ratio ≥ 0.85 (<code>_THRESH_PROBABLE</code>) and the runner-up is not close	the ratio
6	AMBIGUOUS	best ≥ 0.85 , runner-up ≥ 0.80 (<code>_THRESH_AMBIG</code>), gap < 0.05 (<code>_THRESH_GAP</code>): two lines fit	the ratio

Rung	Verdict	Condition	Confidence
7	NOT_FOUND	best ratio < 0.85, or no candidates	0.0 — no line asserted

The Ang modifier. When the caller claims an Ang and a line was matched, the verdict gains +ANG_MATCH or +ANG_MISMATCH(actual=N). A mismatch is still a found line — the quotation exists, just not where the caller said.

webapp/verify.py · _make_verdict · lines 207-284

```

def _make_verdict(candidates, exact_hit, claim_clean, claim_tn, is_gurmukhi, cur):
    if not candidates:
        return {
            "verdict": "NOT_FOUND", "confidence": 0.0,
            "matched_line_id": None, "_best_row": None,
            "distance_details": {"note": "no FTS hits"},
        }

    scored = []
    for rid, _ in candidates:
        row = _fetch_line(cur, rid)
        if row is None:
            continue
        ratio = _score(row, claim_clean, claim_tn, is_gurmukhi)
        scored.append((ratio, rid, row))

    if not scored:
        return {
            "verdict": "NOT_FOUND", "confidence": 0.0,
            "matched_line_id": None, "_best_row": None,
            "distance_details": {"note": "scoring yielded no rows"},
        }

    scored.sort(key=lambda x: -x[0])
    best_ratio, best_rid, best_row = scored[0]
    second_ratio = scored[1][0] if len(scored) > 1 else 0.0
    gap = best_ratio - second_ratio

    # Containment tier: the claim is a clean FRAGMENT of a canonical line
    # (people quote half-tuks constantly). Requires >= 3 words or >= 12 chars.
    if exact_hit is None and (claim_clean or claim_tn):
        needle = claim_clean if is_gurmukhi else claim_tn
        if needle and (len(needle) >= 12 or len(needle.split()) >= 3):
            for ratio, rid, row in scored:
                hay = _clean_gurmukhi(row["gurmukhi"]) if is_gurmukhi \
                    else (row.get("translit_norm") or "")
                # a claim that IS the whole line is not a fragment – let it fall through
                # to the ratio tiers (VERIFIED at 1.0) instead of mislabelling it PARTIAL
                if needle != hay and f" {needle} " in f" {hay} ":
                    return {
                        "verdict": "VERIFIED_PARTIAL",
                        "confidence": 0.95,
                        "matched_line_id": rid, "_best_row": row,
                        "distance_details": {"note": "claim is a fragment of this full
line",
                                         "fragment_len": len(needle)},
                    }

    if exact_hit is not None:
        verdict = "VERIFIED_EXACT"
        confidence = 1.0
    elif best_ratio >= _THRESH_EXACT:
        verdict = "VERIFIED"
        confidence = best_ratio
    elif best_ratio >= _THRESH_PROBABLE:
        if second_ratio >= _THRESH_ambiguous and gap < _THRESH_GAP:
            verdict = "AMBIGUOUS"
        else:

```

```

        verdict = "PROBABLE"
        confidence = best_ratio
    else:
        verdict = "NOT_FOUND"
        confidence = 0.0          # no line is being asserted – a % here only misleads

    best_rid_out = best_rid if verdict != "NOT_FOUND" else None
    best_row_out = best_row if verdict != "NOT_FOUND" else None

    return {
        "verdict": verdict,
        "confidence": round(confidence, 4),
        "matched_line_id": best_rid_out,
        "_best_row": best_row_out,
        "distance_details": {
            "best_ratio": round(best_ratio, 4),
            "second_ratio": round(second_ratio, 4),
            "gap": round(gap, 4),
            "candidates_scored": len(scored),
        },
    },
}

```

Read from the source at build time — [view these lines on GitHub at c3d9360](#) .

Source: `webapp/verify.py · _make_verdict`² — the ladder, read from the file at build time.

The response

```

{
  "verdict": "VERIFIED+ANG_MISMATCH(actual=1)",
  "confidence": 1.0,
  "matched_line_id": 5,
  "ang": 1, "gurmukhi": "...the canonical line, verbatim...",
  "raag": null, "author": null, "comp_id": 2, "section": "...",
  "distance_details": { "best_ratio": 1.0, "second_ratio": 0.0, "gap": 1.0,
  "candidates_scored": 1 }
}

```

gurmukhi is always the **canonical** line from the database, never the caller's text echoed back; comp_id lets a client open the whole composition. distance_details is diagnostic: for a NOT_FOUND it says why (no FTS hits, or the ratios that fell short).

Input handling

- The claim is limited (MAX_CLAIM_CHARS in webapp/sggs/verification.py); an empty claim is HTTP 400.
- FTS tokens are sanitised and quoted with a central empty-MATCH guard: a crafted quote or asterisk cannot break the query or leak an exception (hardened in v2.11.0).
- The engine opens the database read-only and immutable, with query_only as a second fence — the same guarantees as the API's own connection factory.

Pinned by the contract

`contract/golden_verify.ndjson` records real claims and their full verdicts. `make contract` fails if the engine's behaviour drifts; `tools/contract_http.py` replays the same records over HTTP against every deploy and the iOS app asserts its Swift port against the same file. See [Harnesses and golden vectors](#).

Links

1. <https://github.com/Algorithmos-AI/siggs-platform/blob/c3d936031f0c283283c5d1102705affe1f8cf404/webapp/verify.py#L207-L284>
2. <https://github.com/Algorithmos-AI/siggs-platform/blob/integration/webapp/verify.py>

Sri Guru Granth Sahib Ji — Knowledge Base wiki. Generated from commit c3d9360. Updated 25 September 2026. Verified against 7a343c6 on 2026-09-25. This copy may be out of date; the page is at <https://docs.gurbanisoul.com/search/verification-engine/>