ChatGPT

Product Search UX and SEO Best Practices for Saiko Maps V1

Product framing and constraints for Saiko V1

Saiko's V1 posture is best described as **shared-link-first**: most sessions begin on a **Map** (`/map/[slug]`) or **Place** (`/place/[slug]`), and search exists primarily as the "see more" mechanism. This framing matters because it shifts search investment away from "new-user browse" and toward a dependable, low-friction continuation path.

A strong precedent for Saiko's "one query → multiple entity types" approach is multi-type search in products like entity["company","Spotify","music streaming company"], where search is explicitly designed to return multiple categories of results (songs, albums, artists, playlists, profiles, etc.). [1] The key UX lesson that carries into Saiko: when different object types exist, users often don't want to decide *which type* before searching. They expect the system to show them meaningful "shapes" of answers.

The best practice for V1, therefore, is to treat Saiko search as **two parallel retrieval systems** (Maps + Places) presented in a single page, rather than a single blended ranking system. This keeps relevance explainable, reduces ranking disputes, and allows different heuristics per entity type.

Indexed entities and result-type presentation

Entities to index in V1

Saiko V1 should index two primary entities:

Maps
Maps are editorial or personal "collections of places." V1 indexing should prioritize the fields users actually type and the fields that best define a map's identity (title, tagline, area).

Places
Places are merchant/location pages; indexing should prioritize direct lookup fields (name) plus disambiguation fields (neighborhood, category).

This approach aligns with the "search across types" pattern and avoids premature complexity. It also mirrors how workspace search tools like entity["company","Notion","productivity software company"] prioritize titles strongly while allowing content to contribute secondarily. [2]

Recommended fields per entity

Maps (indexed fields, V1)
The goal is fast intent matching plus coherent ranking:

- Title (primary)
- Tagline / short description (secondary)
- Area / neighborhood label, if present (secondary)
- Author display name (tertiary, mostly for display/tie-break)
- Optional V1.5 field: top place names embedded in the map (only if you can also explain "matched on…" in UI)

Places (indexed fields, V1)

- Name (primary)
- Neighborhood and/or locality label (secondary)
- Category / cuisine / type (secondary)
- Optional: synonyms list (manual, small) for common nicknames (e.g., "DTLA")

Result types and visual separation

For mixed-type results, the safest and most legible pattern is **visually separated sections** rather than a single blended list. This is consistent with how multi-type search endpoints are modeled (e.g., Spotify's API and product experience explicitly support multiple "types" under one search). [3]

A clean V1 structure is:

- `/search?q=...`
- **Maps** section (cards)
- **Places** section (cards)

Avoid having Maps and Places "compete" for ranks 1–N in a single stream in V1; doing so forces you to build a cross-type scoring model prematurely and makes relevance harder to defend.

A useful mental model for the page is:

```
Query → retrieve Maps → rank Maps
     → retrieve Places → rank Places
Render two sections; no cross-type blending
```

Card content guidelines for clarity and motivation

Map cards should communicate "collection, not a spot"

Your locked Map card content is directionally correct because it answers three cognitive questions: "what is it," "why should I click," and "is it trustworthy."

Minimum Map card content that supports those questions:

- Hero image (emotional hook)
- Title (recognition)
- Tagline (motivation)
- Place count (the clearest “this is a collection” signal)
- Author (trust/source context)
- Area/neighborhood (disambiguation)

The product critique to enforce is that the **place count must never be relegated to the footer**. It should sit alongside the type label (e.g., “Map · 8 places”) so that users can identify the entity type at a glance.

A useful parallel is Notion’s documented emphasis on titles: it explicitly states that page titles are more likely to match and rank than page contents, and recently edited pages can appear higher under “Best Matches.” [2] For Saiko, this supports weighting Map titles heavily and using other fields primarily as tie-breaks.

Place cards should optimize for disambiguation

Places should remain “atomic” and scannable:

- Photo
- Name
- Single meta line: category + neighborhood

Structured details (hours, address) belong on the place page, not the search card, unless your product repeatedly sees queries where those details are the disambiguator.

Ranking heuristics and editorial boosting

A V1 ranking stack that stays explainable

A strong V1 heuristic approach is to prioritize lexical certainty before business signals. This keeps users’ expectations intact: exact and prefix matches feel “correct,” while fuzzy matches feel “helpful.”

If you adopt a search engine like `entity["company","Algolia","search platform company"]`, its published ranking criteria illustrate this philosophy: typo count is treated as the first-order signal, with exact matches ranking above one-typo matches, which rank above two-typo matches. [4]

If you stay in Postgres, the `pg_trgm` extension is built specifically to support this kind of fuzzy matching and can accelerate similarity and `LIKE` / `ILIKE` queries using GiST or GIN indexes. [5]

Recommended ranking heuristics by entity type

Places ranking (V1)
Order by confidence and recognizability:

1. Exact match on name (case-insensitive)

2. Prefix (starts-with) match on name
3. High similarity match (trigram similarity or engine typo tolerance)
4. Tie-breakers:
5. neighborhood match
6. category match
7. lightweight popularity if you have it (views/clicks), otherwise omit

Maps ranking (V1)

1. Exact match on title
2. Prefix match on title
3. Title contains
4. Description contains
5. Tie-breakers:
6. popularity if available
7. recency (published/updated) only as a tie-breaker
8. editorial boost (see below) only as a tie-breaker

This mirrors what we can observe in Notion's description of "Best Matches," where titles and recency affect ordering, but titles dominate. [2]

Editorial boosts for Saiko-authored maps

Editorial content is valuable for a young marketplace, but it must not feel rigged. An editorial "boost" should therefore be implemented as a **strict tie-breaker**, not a hard pin.

Both Algolia and Typesense document mechanisms for business-driven ranking and boosting:

- Algolia supports custom ranking to incorporate business metrics into ordering. [6]
- entity["company","Typesense","open source search engine"] describes ranking as a tie-breaking algorithm that can combine text match score with user-defined numerical fields (popularity, rating) and explicit boosts for attribute matches. [7]

For Saiko V1, the recommendation is: **never override a stronger lexical match**. If a user map title exactly matches the query and a Saiko map does not, the user map should rank higher.

URL design and SEO rules for Saiko V1

Search results pages should typically be noindex

Search result pages (`/search?q=...`) can create infinite thin variants and can be exploited via "internal search spam." While Google's primary guidance is about controlling indexing via `noindex`, Search Central documentation is explicit about how to block indexing properly: use a `noindex` directive in the page and understand it must be seen by crawlers to take effect. [8]

For Saiko V1, the standard SEO posture is:

- `/search` pages: `noindex,follow`
- `/map/[slug]` pages: indexable, canonical to themselves
- `/place/[slug]` pages: indexable, canonical to themselves

This ensures search pages can still pass internal link equity to maps and places while not competing in Google results.

Canonicals for maps and places

Google's canonicalization documentation describes how Google selects a representative URL for duplicate content and how `rel="canonical"` can be used to consolidate duplicates. [9]
This is relevant to Saiko because: - share links may add tracking parameters - multiple routes can accidentally exist (e.g., old slugs)

V1 rule: each `/map/[slug]` and `/place/[slug]` should set a canonical to its clean slug URL, regardless of tracking parameters.

Structured data: prioritize places and maps as SEO surfaces

Google's structured data documentation emphasizes that structured data must comply with general policies to be eligible for rich results and that Google recommends JSON-LD where possible. [10]

For Saiko, structured data is a high-leverage SEO investment because your objects have clear real-world meaning.

Places (`/place/[slug]`)
Use `LocalBusiness` (and relevant subtypes like `Restaurant` where appropriate). Schema.org defines `LocalBusiness` as a physical business. [11] Google's local business structured data documentation explains that it can help Google show prominent business details (such as in knowledge panels) and mentions business carousels for category queries. [12]

Maps (`/map/[slug]`)
Use an `ItemList` to describe the list of places included in the map. Schema.org's `ItemList` is designed for lists of items and supports `ListItem` to preserve ordering and context. [13]
Even when structured data doesn't guarantee a particular rich feature, it improves machine understanding and makes your pages more self-describing. [10]

Social previews matter because Saiko is shared-link-first

The Open Graph protocol specifies core image and metadata properties and recommends including `og:image:alt` when specifying `og:image`. [14]
Given Saiko's loop ("make/share → click"), V1 OG best practices should be treated as product requirements for `/map` and `/place` pages, not optional SEO polish.

Technical implementation options and trade-offs

Saiko's V1 decision should be driven by dataset size, iteration speed, and operational tolerance. The key product constraint is that search must feel responsive; entity["company","Algolia","search platform company"] guidance suggests aiming for under 100ms response times for an optimal search experience, and it reports that many queries process in low milliseconds on their side. [15] entity["company","Typesense","open source search engine"] publishes benchmark processing times in the tens of milliseconds on large datasets. [16]

Comparative table for Saiko V1

Option	Latency characteristics	Cost characteristics	Complexity	Best features	Key limitations
Postgres `ILIKE`	Can be acceptable for small data; limited ranking	No extra vendor cost	Low	Fast to ship	Weak relevance; poor fuzzy matching
Postgres + `pg_trgm`	Designed for "very fast similarity searches" via GiST/GIN indexes; supports `LIKE` / `ILIKE` acceleration [5]	No extra vendor cost	Low–Medium	Fuzzy match + explainable scoring; minimal ops	Less "instant-search" tooling; no built-in analytics
Typesense	Benchmarks show ~11–28ms processing times in published tests [16]	Cloud is hourly resource-based; no per-record/per-search limits in their positioning [17]	Medium	Typo-tolerant search + filtering/faceting model in API docs [18]	Requires running/operating a search service; RAM-based sizing
Algolia	Support docs claim 1–20ms typical processing time; recommends <100ms overall response target [19]	Pay-as-you-go pricing; plans reference search requests & records [20]	Low ops / medium config	Strong ranking model and typo tolerance docs [4] + event analytics for relevance iteration [21]	Vendor lock-in; ongoing variable cost

Implementation notes tailored to Saiko

Postgres + `pg_trgm` (recommended default for V1)
`pg_trgm` explicitly supports GiST and GIN operator classes for fast similarity searches and can accelerate trigram-based searches for `LIKE` and `ILIKE`. [5] This makes it a pragmatic path to "good enough fuzzy search" without adopting a new system.

Typesense and Algolia (recommended when you want typeahead + filters + iteration loops)
Typesense describes search as a query against one or more text fields plus optional filters against facet/numerical fields; it explicitly supports sorting and faceting. [22]
Algolia's documentation provides a mature model for relevance tuning (ranking criteria, typo tolerance) and supports facets for refinement. [23]
If Saiko plans V2 features like autocomplete and filtering, these systems reduce reinvention.

Accessibility, analytics hooks, and modern patterns

Accessibility requirements by interaction mode

A V1 interaction of "type → Enter → results page" is inherently simpler and avoids complex ARIA requirements. The moment you add autocomplete, you enter "combobox with listbox popup" territory. The WAI-ARIA Authoring Practices describe editable combobox patterns and show how list autocomplete works with a listbox popup and manual selection. [24]

This supports a common best-practice sequencing: ship V1 with a non-autocomplete model, then upgrade once you can implement keyboard and screen-reader behavior correctly.

Baymard and Nielsen Norman Group research also supports caution: enriched search suggestions (beyond simple query suggestions) can be useful but are often underused or poorly implemented, and autocomplete implementations frequently miss key UX details. [25]

Analytics hooks to enable relevance iteration

If you want search quality to improve over time, you need event telemetry. Algolia's Insights documentation defines event types (click, conversion, view) and explains that sending events unlocks features and provides actionable metrics. [26]
Even if Saiko doesn't use Algolia, its model is a clean blueprint for what to log.

A Saiko V1 analytics schema should capture: - Query submitted - Results returned (counts per section) - Clicked result (entity type + rank-in-section) - No-results state - "Create" CTA from search (query-to-creation conversion)

Modern patterns and integrations to consider

Autocomplete (V2)
Autocompletion is widely expected, but it carries accessibility and complexity costs. The WAI-ARIA combobox pattern should be treated as a hard requirement if implemented. [24]

Intent framing (V2)
Platforms like entity["company","Pinterest","visual discovery company"] have published work on improving query-to-content relevance using advanced models, and their consumer product direction has leaned into generating "the right words" and refinements for users who don't know what to type. [27]
For Saiko, a conservative V2 adaptation is to use identity signals to suggest "vibe" refinements, but only once the underlying catalog can support it without dead ends.

Filters (V2)
Filtering and faceting are powerful, but they require (a) consistent structured fields and (b) a UI that doesn't overwhelm. Algolia's faceting docs describe facets as categories for refining results and explain that facet counts can be displayed. [28]
Typesense similarly expects a separation between query fields and filter/facet fields. [22]
This is a natural V2 evolution, not a V1 dependency.

V1 recommendations, deferrals, and a migration path to V2

What to implement now in V1

Saiko V1 should implement a search experience that is highly legible, SEO-safe, and operationally light:

- Index Maps and Places as distinct entity types with separate ranking heuristics, displayed as two sections under `/search?q=...`.
- Use deterministic lexical ranking (exact, prefix, fuzzy) where fuzzy matching is implemented via `pg_trgm` similarity or equivalent. [29]
- Apply editorial boosts only as tie-breakers using an explicit attribute (e.g., `author_type = saiko`), consistent with the "business ranking as tie-break" approach described in search engines like Typesense and Algolia. [30]
- Set `/search` pages to `noindex,follow` using robots meta directives; keep `/map` and `/place` indexable with canonical self-URLs. [31]
- Implement structured data on destination pages:
- `/place`: `LocalBusiness` (Google + Schema.org guidance) [32]
- `/map`: `ItemList` for the list of places [13]
- Treat OG tags as product-critical (Open Graph protocol) and include `og:image:alt` when using `og:image`. [14]
- Add search analytics event logging modeled on click/view/conversion semantics. [33]

What to defer to V2

- Autocomplete/typeahead: requires correct combobox accessibility behavior. [24]
- Refinement filters (neighborhood/category chips) and facet counts: require additional UI/system complexity and consistent metadata fields. [34]
- Intent framing and "vibe suggestions": valuable, but best after you have consistent identity signals and enough content density to avoid dead ends.
- Richer suggestions ("you might also like"): defer until you have interaction signals and confidence models; NNG/Baymard research supports caution due to implementation pitfalls and frequent low utilization. [25]

Migration path to V2

A safe path preserves URL contracts and UI structure while improving relevance infrastructure:

V1 → V1.5
Keep `/search?q=` and sectioned results, but add: - `pg_trgm` similarity ranking refinements (if not already) - popularity instrumentation (views, clicks) as tie-breakers - improved OG asset generation consistency

V1.5 → V2
Introduce either Typesense or Algolia when you need: - real-time autocomplete - filters/facets - scalable relevance iteration with analytics loops

This is consistent with how large-scale content systems separate retrieval and ranking into stages; Pinterest's published work on retrieval and relevance improvements illustrates the multi-stage mindset even if Saiko won't need that complexity in early V2. [35]

A practical decision rule: - If Saiko wants to remain "minimal ops," Algolia's hosted model plus event analytics can accelerate product iteration. [36] - If Saiko wants "owned infra" and predictable cost, Typesense's resource-based pricing and published benchmark performance make it a strong fit. [37]

---

[1] Search
https://support.spotify.com/us/article/search/?utm_source=chatgpt.com

[2] Search in your workspace – Notion Help Center
https://www.notion.com/help/search?utm_source=chatgpt.com

[3] Web API Reference | Spotify for Developers
https://developer.spotify.com/documentation/web-api/reference/search?utm_source=chatgpt.com

[4] Typo tolerance
https://algolia.com/doc/guides/managing-results/optimize-search-results/typo-tolerance?utm_source=chatgpt.com

[5] [29] F.35. pg_trgm — support for similarity of text using trigram ...
https://www.postgresql.org/docs/current/pgtrgm.html?utm_source=chatgpt.com

[6] Custom ranking
https://algolia.com/doc/guides/managing-results/must-do/custom-ranking?utm_source=chatgpt.com

[7] [30] Ranking and Relevance
https://typesense.org/docs/guide/ranking-and-relevance.html?utm_source=chatgpt.com

[8] [31] Block Search Indexing with noindex
https://developers.google.com/search/docs/crawling-indexing/block-indexing?utm_source=chatgpt.com

[9] How to specify a canonical URL with rel="canonical" and ...
https://developers.google.com/search/docs/crawling-indexing/consolidate-duplicate-urls?utm_source=chatgpt.com

[10] General Structured Data Guidelines | Google Search Central
https://developers.google.com/search/docs/appearance/structured-data/sd-policies?utm_source=chatgpt.com

11 LocalBusiness - Schema.org Type
https://schema.org/LocalBusiness?utm_source=chatgpt.com

12 32 Local Business (LocalBusiness) Structured Data
https://developers.google.com/search/docs/appearance/structured-data/local-business?utm_source=chatgpt.com

13 ItemList - Schema.org Type
https://schema.org/ItemList?utm_source=chatgpt.com

14 The Open Graph protocol
https://ogp.me/?utm_source=chatgpt.com

15 Troubleshooting Search Latency
https://support.algolia.com/hc/en-us/articles/32147896160145-Troubleshooting-Search-Latency?utm_source=chatgpt.com

16 Typesense Benchmarks
https://typesense.org/docs/overview/benchmarks.html?utm_source=chatgpt.com

17 37 Pricing
https://cloud.typesense.org/pricing?utm_source=chatgpt.com

18 22 Search Parameters
https://typesense.org/docs/30.1/api/search.html?utm_source=chatgpt.com

19 How fast is Algolia?
https://support.algolia.com/hc/en-us/articles/4406975267089-How-fast-is-Algolia?utm_source=chatgpt.com

20 36 Pricing
https://www.algolia.com/pricing?utm_source=chatgpt.com

21 Get started with click and conversion events
https://algolia.com/doc/guides/sending-events/getting-started?utm_source=chatgpt.com

23 The eight ranking criteria
https://algolia.com/doc/guides/managing-results/relevance-overview/in-depth/ranking-criteria?utm_source=chatgpt.com

24 Editable Combobox With List Autocomplete Example | APG
https://www.w3.org/WAI/ARIA/apg/patterns/combobox/examples/combobox-autocomplete-list/?utm_source=chatgpt.com

25 Enriched Site-Search Suggestions: Rarely Used
https://www.nngroup.com/articles/enriched-site-search-suggestions/?utm_source=chatgpt.com

26 33 Event types
https://algolia.com/doc/guides/sending-events/concepts/event-types?utm_source=chatgpt.com

27 35 Improving Pinterest Search Relevance Using Large ...
https://medium.com/pinterest-engineering/improving-pinterest-search-relevance-using-large-language-models-4cd938d4e892?utm_source=chatgpt.com

28 34 Faceting
https://algolia.com/doc/guides/managing-results/refine-results/faceting?utm_source=chatgpt.com