METEHAN.AI RESEARCH REPORT · PREPRINT FIELD GUIDE

Gemini Embeddings for SEO: A Field Glossary and Concept Map

Metehan Yesilyurt

AI Search & Information Retrieval Researcher · metehan.ai

https://metehan.ai/pdf/gemini-embeddings-seo-field-glossary.pdf

Abstract. A practical glossary and decision map for using Google Gemini embeddings as an SEO and AEO diagnostic. It defines the core terms, shows how to connect crawl data to citation checks, and gives worked examples for duplicate content, query alignment, and multilingual clustering. The goal is to help you turn vector scores into specific content actions.

Keywords: Gemini embeddings, SEO diagnostics, AEO, semantic similarity, content clustering, query alignment

Companion to the article SEO Insights with Google's Gemini Embeddings (A Free Tool Inside). The article covers the topic in full; this document is a separate, practical field glossary and concept map.

1. How I Use This Field Glossary

This is not a product tour. I wrote it as a working glossary and decision map for the same core topic as the source article: a mini AI SEO crawler built around Google's Gemini embedding model and a free tool for testing it [1][2]. The source gives the background. This document gives me criteria, steps, tests, and stop rules I can apply when I audit content for answer engine optimization.

The premise is simple. Embeddings translate text into numeric representations that capture meaning and context [1]. If a page vector sits near a query vector, the page is a candidate for retrieval and citation. If it sits far away, the page may still rank in classic search, but it is less likely to be pulled into an AI answer. I treat similarity as a diagnostic, not a verdict. I still verify with human review and live answer tests.

I keep three questions in front of me. What is the unit of meaning I am embedding? What distance or similarity measure am I using? What decision will the score change? If a score cannot change a decision, I do not compute it.

2. Core Terms in Plain Language

The table below is the glossary I use when I explain embedding work to SEO teams. The definitions are plain language. The SEO and AEO column is where the term becomes useful.

Term	Working definition	SEO and AEO use	Caution
Embedding	A numeric representation of text that preserves semantic meaning and context [1].	Compare pages, queries, and snippets in one vector space.	A close vector does not prove factual accuracy or quality.
Vector	The ordered list of numbers produced by an embedding model.	Store and retrieve content by meaning instead of exact keywords.	Vector search can return near matches that miss intent.
Dimension	The length of the vector. Gemini embeddings use a high-dimensional output of 3K dimensions [1].	More dimensions can capture finer distinctions between topics.	Higher dimensions cost more to store and compute.
Token	A chunk of text the model processes as input. Gemini embeddings support an 8K token input limit [1].	Larger chunks can preserve more context for long sections.	Oversized chunks can blur multiple intents into one vector.
Matryoshka Representation Learning (MRL)	A training approach that lets embeddings scale to different sizes for storage and compute needs [1].	Reduce vector size when full size is not needed.	Truncation can lower recall if tested poorly.
Semantic similarity	A measure of how close two embeddings are in meaning.	Rank queries against pages or page sections.	High similarity can also mean duplicate content.
AEO	Answer engine optimization, the work of making content retrievable and citable in AI answers [1].	Check whether content sits near the queries that drive AI citations.	AEO does not replace technical SEO or editorial quality.
Chunk	A section of a page that is embedded as its own unit.	Match specific answers to specific questions.	Bad chunk boundaries create bad scores.
Cluster	A group of embeddings that are close to one another.	Find topic gaps, cannibalization, and content groups.	Clusters need human labels to be useful.
Duplicate content	Pages or sections that are semantically near-identical.	Decide whether to merge, redirect, or differentiate.	Near duplicates can split citations and links.
Search intent alignment	The degree to which a page matches what the searcher wants to do.	Map queries to the best page section.	Intent labels are hypotheses until tested.
Topical categorization	Assigning pages to topics based on their vectors.	Build hubs and internal link paths.	A category label can hide mixed intent.

3. Concept Map: From Crawl to Citation

I use a linear pipeline because it keeps the embedding work tied to content decisions. The source article describes a crawler that leverages Gemini embeddings [1]. My version adds checkpoints where I would stop or change course.

1. Collect URLs and extract the main content. Remove navigation, footers, and boilerplate so the vector reflects the page topic.
2. Chunk the content by heading and paragraph. Test a few chunk sizes within the 8K token input limit [1]. I want each chunk to answer one question.

3. Generate embeddings with the Gemini model. Keep the raw text next to each vector so I can inspect what the model saw.
4. Build a query set. Include target questions, search intents, competitor headings, and prompts that trigger AI citations in ChatGPT, Perplexity, and Gemini [1].
5. Score page chunks against queries. Rank the best chunk for each query and the best query for each chunk.
6. Label the result. A strong match with unique information is an AEO candidate. A strong match with near-identical text is a consolidation candidate. A weak match with high business value is a rewrite or new page candidate.
7. Act on the label. Rewrite, merge, redirect, interlink, or leave alone. Then re-embed and compare.
8. Log the decision. Record the query, page, score band, action, and re-test date. Use the ask endpoint for follow-up questions [7] and the facts endpoint when claims need sources and dates [8].

4. Gemini Embedding Spec Sheet for SEO Decisions

The source article highlights several features of the Gemini embedding model [1]. I turn each feature into a test and a decision rule. The free tool is the quickest way I know to run the first pass [2].

Feature from source	What it changes for SEO	Test I run	Decision rule
8K token input limit [1]	I can embed larger chunks and preserve more context.	Compare heading-level chunks with larger section-level chunks.	Use the larger chunk only when it improves query matching without mixing intents.
3K dimensions [1]	Richer semantic detail can separate close topics.	Check whether fine distinctions appear in the top matches.	Use full dimensions when storage allows. Reduce only if recall holds.
Matryoshka Representation Learning [1]	Embeddings can scale to storage and compute needs.	Compare full vectors with reduced vectors on the same query set.	Truncate when the top results and decisions stay the same.
Over 100 languages [1]	One model can handle multilingual content and code embedding [1].	Run the same intent in each language and compare matches.	Keep one index when language scores align. Split only when they do not.
Unified model [1]	Multilingual text and code can share a workflow.	Embed prose and code snippets in the same test.	Keep one pipeline if code queries still find the right snippets.
Free tool [2]	A low-cost first diagnostic before a rewrite.	Run duplicate, intent, and category checks before editing.	If similarity is already strong, fix formatting and evidence instead of rewriting.
Agent API [3]	Programmatic follow-up and fact checks are available without a key [3].	Ask a follow-up question [7] or request facts with sources and dates [8].	Use the API when a decision depends on a claim I cannot verify locally.
Markdown and JSON versions [5][6]	Content can be consumed by agents in structured formats.	Fetch the Markdown or JSON version for automated checks.	Use structured output when I need repeatable parsing.

5. Worked Example 1: Duplicate and Near-Duplicate Triage

Suppose I have three pages about Gemini embeddings for SEO. Page A is a general explainer. Page B is a rewritten version of Page A with a new title. Page C is a workflow for auditing content with embeddings. I embed the main content of each page and compare the vectors.

The first decision is not about rankings. It is about whether the pages occupy the same semantic space. If Page A and Page B are near-identical, I do not publish both. I choose the stronger page, merge any unique sections, and redirect the weaker URL. If Page C is close to Page A but not identical, I check intent. A general explainer and an audit workflow can be related, but they should not collapse into one vector. If they do, I rewrite the introductions, add original examples, and differentiate the headings.

I use a triage table to make the decision repeatable.

Pair signal	Likely problem	Action	Re-test
Near-identical vectors, same intent	Duplicate content	Merge and redirect	Re-embed the survivor and confirm it still matches the target query.
High similarity, different intent	Cannibalization risk	Differentiate with unique examples and headings	Check that each page has a distinct best query.
Moderate similarity, related topic	Weak internal linking	Add contextual links and a hub section	Check that the hub chunk matches both child topics.
Low similarity, same topic label	Mislabeled category	Review the taxonomy or split the topic	Re-cluster and inspect the new groups.

6. Worked Example 2: Query-to-Page Alignment

I build a query set before I score anything. The set includes questions I want AI systems to answer with my content. I embed each query and each page chunk, then create a query-by-page matrix. I am not looking for a single winner. I am looking for patterns.

- Diagonal strength: each important query has one clear best page. This is the healthy state.
- Off-diagonal noise: several pages compete for the same query. This is cannibalization or unclear positioning.
- Empty rows: important queries have no strong page. This is a content gap.
- Empty columns: pages match no important query. These pages may be orphaned or off-topic.

The decision rules are direct. If a query has a strong match but the page gives a thin answer, I add a direct answer, evidence, and an example. If two pages match the same query strongly, I consolidate or differentiate. If a high-value query has no strong match, I create a new page or rewrite an existing one. If a low-value query has no match, I ignore it.

Here is a small worked case. Query one is how to find duplicate content with embeddings. Query two is Gemini embeddings for SEO. A page about an AI SEO crawler [1] may match query two strongly and query one moderately. I would add a section that walks through duplicate detection with embeddings, then re-embed the page. If the new section pulls query one into a strong match without weakening query two, the change worked. If it weakens query two, I split the content.

7. Worked Example 3: Multilingual and Code Content

The source article notes expanded language support and code embedding [1]. I use that for two tests. The first test is multilingual coverage. I embed the same intent in each language the site supports. If the German page is close to the German query and the English page is close to the English query, the index is healthy. If the German query matches the English page more strongly than the German page, I have a localization or intent problem. I check hreflang, translation quality, and local examples before I change the vectors.

The second test is code content. I embed code snippets and prose separately, then compare both to implementation queries. If a query asks how to run an embedding check, a code block may be the best match. If the code block is far from the prose, the page may not answer implementation questions even if the prose is strong. I keep code blocks in the embedding text when they answer how-to questions. I remove them when they are decorative or duplicated from a library.

The decision rule is simple. Use one model and one index when the same intent is retrieved across languages and code types. Split the index only when a language or content type fails the test. The source describes a unified model that handles over 100 languages and code embedding [1]. That is an argument for testing one pipeline first, not for assuming it will work without checks.

8. Decision Rules and Scorecard

I use a scorecard to keep embedding work honest. The scorecard is qualitative because the decision matters more than the decimal. I record the signal, the action, and the re-test condition.

Criterion	Question I ask	Pass signal	Fail signal	Action
Semantic closeness	Does the page chunk sit near the target query?	The target query is in the top matches for the chunk.	The target query is absent from the top matches.	Rewrite, add a section, or create a new page.
Intent match	Does the page answer the job behind the query?	The best chunk matches the query intent label.	The best chunk matches a different intent.	Differentiate or split the page.
Duplicate risk	Is this page near-identical to another page?	Each page has a distinct best query.	Two pages share the same best query and similar vectors.	Merge, redirect, or rewrite.
Information gain	Does the page add something the top results do not?	The page includes original tests, examples, or data.	The page repeats common claims.	Add original evidence or stop optimizing.
Extraction format	Can an answer engine lift a clear answer?	The page has direct answers under clear headings.	The answer is buried in long paragraphs.	Add summaries and structured sections.
Multilingual coverage	Does each language retrieve its own page?	Language-specific queries match language-specific pages.	One language dominates all queries.	Fix localization or split the index.
Code coverage	Do implementation queries find code snippets?	Code snippets are top matches for how-to queries.	Prose matches but code does not.	Add or reposition code blocks.
Evidence	Are claims backed by sources and dates?	The facts endpoint confirms the claim [8].	The claim has no source or date.	Add citations or remove the claim.

I also use stop rules. Stop rewriting when similarity is already strong and the content is unique. Stop publishing when two pages are near-identical. Stop optimizing an existing page when a high-value query has no strong match and the page structure cannot support it. Stop adding keywords when the vector is already close but the answer is unclear. At that point, the problem is formatting and evidence, not semantic distance.

Finally, I keep the agent endpoints close. I can ask a follow-up question about the topic [7], fetch facts with sources and dates [8], and read the Markdown or JSON version of the source page when I need structured input [5] [6]. The full endpoint list is at the agent home [3] and in llms.txt [4]. These are not SEO tricks. They are ways to make the diagnostic repeatable.

REFERENCES & SOURCES

[1] Source article: SEO Insights with Google's Gemini Embeddings, https://metehan.ai/blog/seo-insights-with-googles-gemini-embeddings-a-free-tool-inside/

[2] Free embeddings tool, https://embeddingsv3.vercel.app

[3] Agent API home, https://metehan.ai/

[4] llms.txt endpoint list, https://metehan.ai/llms.txt

[5] Source article as Markdown, https://metehan.ai/api/post/seo-insights-with-googles-gemini-embeddings-a-free-tool-inside.md

[6] Source article as JSON, https://metehan.ai/api/post/seo-insights-with-googles-gemini-embeddings-a-free-tool-inside.json

[7] Follow-up question endpoint, https://metehan.ai/api/ask/seo-insights-with-googles-gemini-embeddings/

[8] **Facts with sources and dates**, https://metehan.ai/api/facts.json