SEMANTIC SEARCH:

Bringing search experiences into the AI era

We've all looked for information and become frustrated when we can't find what we're looking for. Failed search experiences lead to customer dissatisfaction, missed opportunities for revenue, and decreased customer loyalty. Unlike traditional lexical search that matches keywords and sometimes fails to capture query intent, semantic search finds what you mean — even if you're unfamiliar with the terminology and don't understand the domain of your inquiry.

This white paper describes two approaches to implementing semantic search, contrasts their suitability for different teams and use cases, and explains how Elasticsearch supports both approaches:

- **Approach 1:** Works out-of-the-box, and therefore is suitable for beginners
- **Approach 2:** Depends on custom models, needs tuning, and requires specific skills, but applies to advanced use cases

Before we get into approaches for implementing semantic search, let's review what semantic search can deliver and why it's key to capturing a searcher's intent.

Why you need semantic search

We've come to expect a search box on the websites we visit and apps we use, but the traditional, keyword-based search that's under the hood of most of those experiences fundamentally matches content on keywords. That approach has limitations for multiple reasons:

- **Vocabulary mismatch:** The query doesn't contain the exact keyword, but instead uses synonyms, abbreviations, acronyms, or alternative terminology. For example, let's say you're trying to find out whether your employer owns the side project you're working on. The phrase "side hustle" — which the searcher might use in their query — is unlikely to appear in your company's policy documentation.
- **Semantic mismatch:** Keywords lack the context in which they are used, and can be ambiguous; for example "Where's the bank" may refer to a financial institution, a place to rest, or the bank of a river.
- **Domain mismatch:** A combination of the issues above occurs when the user is unfamiliar with the domain or seeks information that's not explicitly covered by the knowledge database.

Semantic search overcomes these limitations by applying machine learning to capture the meaning and context of queries and documents.

Use cases for semantic search

Semantic search has two primary use cases:

- Retrieving more relevant information from text — finding what the user actually means more reliably than traditional search
- Providing specific and nuanced context to generative AI

Semantic search captures user intent and serves up more relevant results

What consumers expect from search experiences is quickly evolving. To stay competitive, AI-powered approaches are becoming the new normal, and businesses and organizations are eager to reap the benefits. Specifically, semantic search:

- **Overcomes limitations of keyword-based search:** Finds relevant content even if the vocabulary doesn't match the keywords you configured, and eliminates the need to manage keywords and synonyms.
- **Provides more conversational interaction:** Better handles language nuances and idiomatic expressions, with additional benefits for searching in other languages.
- **Enables the extraction of additional information**: Extracts sentiments and named entities, since natural language processing (NLP) techniques use the same class of machine learning models as semantic search.

Consumers search for relevant information in many contexts, therefore many industries can benefit from implementing semantic search, providing more flexibility in expressing queries, and leveraging semantic relationships between multiple sources. Examples include:

Retail
"Where can I find an outfit like the singer wore at the concert?"

Finance
"How are proceeds from selling company stock taxed?"

Health care
"Where are the latest outbreaks from the virus?"

Entertainment
"Where can I still get tickets to the concert in town?"

Education
"Which courses do you recommend for someone working in real estate?"

Semantic search powers accurate generative AI

In the face of the variability and ambiguity of natural language, it's best to use semantic search to retrieve the best, most nuanced context. Therefore, semantic search is an essential component of your GAI-powered application, ensuring accurate information and limiting the cost by passing only the most relevant data to the large language model (LLM). Retrieving relevant context from proprietary enterprise data and feeding into LLM has become known as retrieval augmented generation (RAG). Semantic search thus helps provide the bridge between private knowledge bases and the LLMs needed to implement GAI.

Surveys[1] confirm that executives feel a high degree of urgency to adopt generative AI, but don't see the technology as being ready for implementation in production. By contrast, with recent advances, semantic search can work out-of-the-box, ready to be implemented in production environments. Elastic supports both — using semantic search to apply generative AI — letting you bridge that gap!

Two approaches to semantic search: out-of-the-box vs. custom-built

Your approach to semantic search — whether you choose an out-of-the-box solution or take a custom approach — depends on a few factors:

- Your experience with AI
- The type(s) of data you're searching
- Your time horizon for implementation
- Availability of labeled data — documents and queries with known relevance rankings, which is needed to evaluate and tune your retrieval performance

	Out-of-the-box	Custom (dense vector)
Types of data	Text search	Text or multimodal
Your experience with AI	Little to none	Data science skills
Labeled data	Limited	Yes
Time horizon	Short — go to market quickly	Longer — can afford to iterate
Suitable for...	Beginners	Advanced

Approach 1: Semantic search out-of-the-box

Recent advances in information retrieval have yielded a new generation of models for semantic search that generalize well and don't require domain adaptation.

Out-of-the-box models for semantic search may use sparse encoders. In contrast to dense vector embeddings, sparse representations contain few non-zero values. Instead of encoding context and meaning using an embedding model, sparse encoders provide relevance scores for words and documents. In a process called term expansion, terms are added based on their relevance to the document. The model generalizes well because of the large static vocabulary that represents tokens, words, and sub-word units, and because of the late interaction between relevance scores of the query with the document.

A recent blog[2] compared keyword-based search (BM25) with three OOTB semantic search models. Results show state-of-the-art, out-of-the-box models significantly increase relevance:

- SPLADE introduced the sparse encoder approach and is now available in a second version, but not licensed for commercial use
- ELSER: Elastic's Learned Sparse Encoder
- E5 refers to a family of embedding models that also generalize across domains, just like the sparse models above, developed by Microsoft researchers[3].

BM25	ELSER	SPLADE	E5-base
0.41	0.53	0.48	0.49

Approach 2: Custom-built semantic search using dense text embeddings

The dense vector approach to semantic search in most cases requires a custom embedding model and additional tuning, and therefore requires advanced skills.

Central to the dense vector approach is a machine learning model that transforms finding similarity in meaning to finding nearest neighbors among high-dimensional vector representations, often called embeddings. The text embedding model needs to be trained on large amounts of text, and applied to knowledge base documents and query text. The nearest neighbors among all documents are then returned as the search result.

This approach can deliver impressive results, which is why there has been so much recent interest in vector search. Semantic search outperforms keyword-based search (using BM25, a term-based ranking model) by a significant margin.

However, achieving good performance with the vector search approach assumes the embedding model is adapted to your domain. This process requires massive amounts of labeled data and expertise in training deep neural networks — an obstacle for broad adoption of semantic search. Illustrating the need for domain adaptation, the table below from Elastic's blog on benchmarking vector search shows how two popular, publicly available embedding models perform worse than the BM25 baseline across a variety of datasets taken from the well-known BEIR benchmark[4].

Data sets (BEIR benchmark) →

Search results ranking model ↓

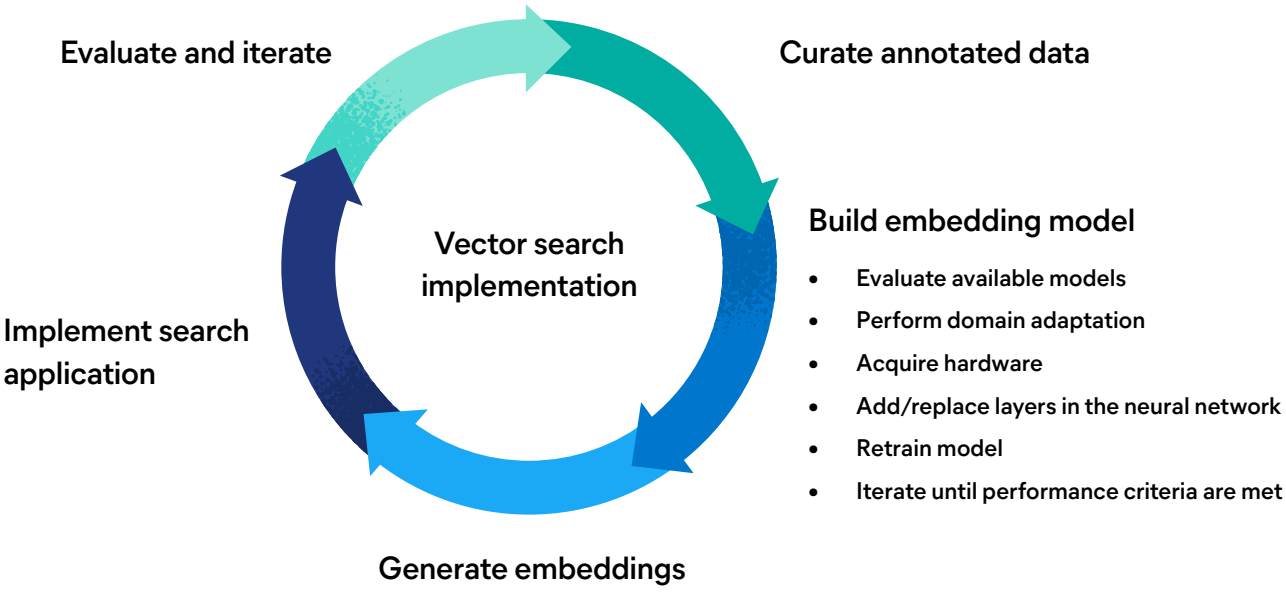
	TREC COVID	NFCorpus	NQ	HotpotQA	FiCA	ArguAna	Touche	DBPedia	SCIDOCS	FEVER	climate FEVER	SciFact	Average
BM25	0.688	0.327	0.326	0.602	0.254	0.472	0.347	0.287	0.165	0.649	0.186	0.69	0.416
ANCE	0.654	0.237	0.446	0.456	0.295	0.415	0.284	0.281	0.122	0.669	0.198	0.507	0.38
TAS-B	0.481	0.319	0.463	0.584	0.3	0.427	0.173	0.384	0.149	0.7	0.228	0.643	0.404

Scores: NDCG@10

Note: relevance is commonly evaluated using a measure called NDCG, which has values between 0 and 1 — the closer to 1 the better.

As mentioned above, domain adaptation requires curating a large set of labeled data, AI skills to train the actual model, and GPU hardware to limit training time. The complete workflow for implementing dense vector search also involves:

- Evaluating available (off-the-shelf) models to select which one to use as the starting point for adaptation
- Generating embeddings for your data
- Iterating until the performance criteria of the deployed search application are met



Fine tuning an embedding model can take anywhere from weeks to months, so be sure to budget ample time for this process. You can find an outline for the adaptation process in this blog. That's why the out-of-the-box approach is more suitable if you're on a tight timeline and need to bring semantic search to market quickly.

Choosing the approach that fits you

Now that you have an overview of the two approaches, you may wonder how you should proceed with implementing semantic search for your use case.

Out-of-the-box semantic search using the sparse vectors offers multiple advantages:

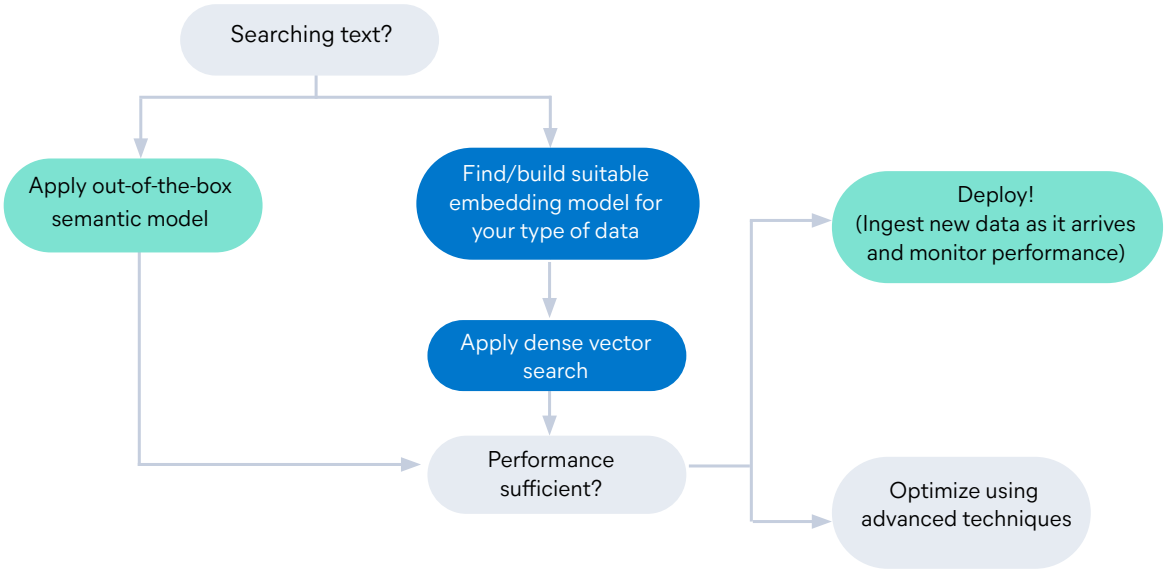
- **Easy to implement:** You don't have to build a performant embedding model, which requires a large volume of labeled data and AI expertise, and you can apply the same retrieval algorithms you employed for lexical search. No need to optimize nearest neighbor search that's required for dense vectors.
- **Fast:** Sparse vectors can be searched efficiently using the same inverted indices that make traditional keyword-based search so fast.
- **Interpretable:** You can follow along as terms are added, and the attached score indicates how relevant a term is to a query, whereas numeric representations derived from the embedding model are not interpretable.

Sparse encoders	Dense vector search
Performant out-of-the-box	Highest relevance (if tuned)
Text only	Multimodal
Interpretable	Not interpretable
Difficult to tune	Demanding on compute and storage

As a downside, the sparse approach doesn't lend itself to:

- Further tuning performance beyond applying hybrid search
- Searching unstructured data other than text
- Using your own embeddings

For all use cases that require tuning, non-textual data, or a need to create your own embeddings, you'll need to employ the dense vector approach. The flowchart below illustrates the decision points to consider.



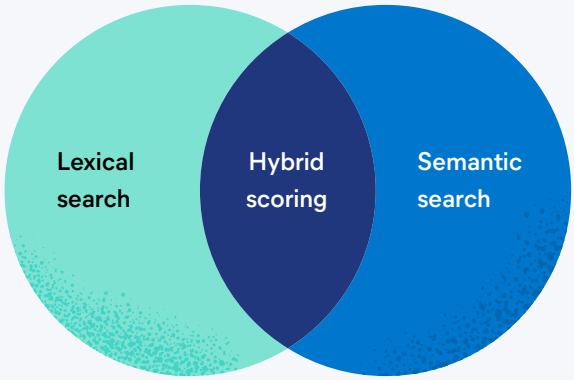
As a first step to improving relevance — regardless whether you employ out-of-the-box or custom semantic models — you can try what's known as hybrid search.

Hybrid search

Hybrid search refers to combining traditional lexical search with one of the vector-based approaches to semantic search.

It's well known in machine learning that combining multiple models leads to more accurate predictions. Similarly, combining multiple ranking approaches improves information retrieval. Hybrid search yields modest improvements in relevance regardless of the ranking methods combined. Since some methods for combining multiple rankings don't require tuning, they work out-of-the-box. All you need is lexical search that works decently well on your domain.

Why does hybrid search help? Lexical and semantic search complement each other: Traditional keyword-based search can be configured on domain-specific, technical terminology, and work better on single-word queries and exact brand match queries. Embedding models have difficulty learning domain specifics, but they capture context and associations and excel at multi-word queries, concept searches, questions, and other more complex query types. Sparse approaches generalize better, but may not perform as well on domain-specific data, technical, or product terms.

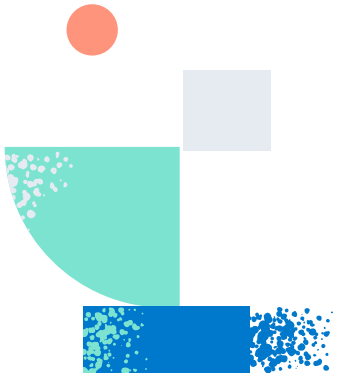


Multiple methods for combining rankings exist, such as reciprocal rank fusion (RRF) and linear combination. RRF is easy to apply because there's no need to tune parameters or normalize scores. Hybrid ranking with RRF yields an improvement in relevance scores by about 1%, while an optimized, linear combination can yield a 5% improvement.

	Relevance score
BM25	0.439
ELSER	0.512
Hybrid - RRF	0.519
Hybrid - tuned Linear	0.543

Hybrid search with RRF versus linear combination of BM25 and Elastic's Learned Sparse Encoder. Relevance scores represent an average of NDCG@10 across a subset of the BEIR benchmark.

With linear combination, rankings are combined as a weighted sum of the scores of each individual ranking. The big benefit of linear combination is that it's interpretable. However, calibration of linear combination requires scores to be normalized and annotations similar to those used for fine-tuning a model. Additionally, the optimal weight varies between data sets. Therefore, the linear combination method is appropriate only if you have data analytics expertise on your teams.



Why choose Elastic for implementing semantic search

Elastic supports semantic search out-of-the-box:

- ☑ Elastic Learned Sparse Encoder, a retrieval model trained by Elastic which comes with a full commercial license as part of the Elasticsearch Relevance Engine™ (ESRE), and delivers best-in-class retrieval performance. You can learn about the model's architecture, how it was trained, and how it outperforms alternative approaches in this blog on improving information retrieval.

- ☑ Hybrid search with reciprocal rank fusion, which is easy because Elastic comes with market-leading lexical search.

Elastic also supports the advanced, second approach to semantic search:

- Flexibility to apply all of the advanced techniques to optimize performance and keep up with innovations in this rapidly evolving space, including bringing your own optimized model and tuning linear combinations of multiple rankings.

- An underlying vector database (including approximate nearest neighbor search with HNSW) implemented in Lucene, the most widely distributed relevance engine in the world.

And Elastic provides all the other elements needed in enterprise-grade applications in the same mature platform:

- Full gamut of modern search application capabilities, including aggregations, filtering, faceted search, and auto-complete. (Many dedicated vector databases support only some of these capabilities.)
- Solutions for security and observability, resulting in less complex implementation and lower licensing cost.
- Agnostic data store deployment (on-prem or with any cloud provider) that allows hosting on hybrid environments with cross-cluster search.

How to implement semantic search in Elastic

Elastic makes it easy to implement semantic search using the out-of-the-box approach:

1. Elastic's out-of-the-box model for semantic search (Elastic Learned Sparse Encoder) comes with the platform and can be easily downloaded to the dedicated ML nodes in your cluster.
2. Apply the model to your data while ingesting or as a post-processing step.
3. Execute queries, using the same _search endpoint that you use for textual BM25 search.

Follow this guide for step-by-step instructions on how to ingest and prepare your data for semantic search (steps 1 and 2), and this interactive demo on how to run semantic search queries against that data (step 3).

Case study:

Streamlining customer service with semantic search

Great customer service makes it easy for consumers to find relevant information and resolve issues on their own. Semantic search can streamline that process in two ways:

- Deflecting routine support inquiries by delivering targeted answers in a timely fashion, potentially even tailoring content to specific customer profiles. When customers have difficulty finding answers, the problem shifts to delivering agent-assisted customer support, which is more costly.

- Empowering support agents to handle customer support inquiries more efficiently. While handling support inquiries, agents can semantically search across external and internal knowledge bases and find relevant answers faster, resulting in quicker case resolution.

Business outcomes of applying semantic search are higher customer satisfaction and lower costs for delivering customer service.

Customer story: Cisco accelerates handling of technical support inquiries

Cisco wanted to deliver a first-class search experience to website visitors and add advanced search capabilities to its internal- and external-facing apps. They expected AI and modern cloud-native technologies to accelerate the resolution of customer issues.

Applying semantic search, their support engineers can quickly find similar cases in real-time using error messages related to the customer's problem, associated product bugs, and knowledge articles. Employing deep learning models to pre-process the user query that Elastic sends to the backend, search results pages identify specific sections that are related to the support issue at hand, not just links to web pages.

As a result, user engagement via click-through rates has gone up drastically, and the response time for a search query is now 73% faster, reducing the likelihood that people will lose interest while the page loads.

> *Feedback from our engineers is extremely positive. They now use Topic Search to solve 90% of service requests. They can deliver a better customer experience by easily finding on-target information and fixing issues much faster than before.*
>
> **Sujith Joseph, Principal Enterprise Search & Cloud Architect, Cisco Systems**

Read the full story on our website.

Summary

Semantic search lets you bring “find what you mean” search experiences to your customers, and it’s ready for broad adoption in production environments. Out-of-the-box semantic search models like Elastic Learned Sparse Encoder let you bring this technology to market quickly — no tuning, configuration, model selection, nor domain adaptation required.

Implementing semantic search is a journey that lends itself to progressing to more advanced approaches. To optimize performance beyond what an out-of-the-box model can deliver, and for advanced use cases, the natural progression is via hybrid search and potentially applying dense vector search. As a final step — on top of any search — you can apply generative AI for more human-like experiences. Elastic gives you the flexibility to apply these advanced approaches using a proven, innovative, and feature-rich platform.

AI-powered search is a journey, from out-of-the-box models all the way to applying generative AI.

Elastic is recognized as a leader by industry analysts and has been adopted at scale by over 50% of the Fortune 500 market, and it’s supported by a vibrant developer community.

elastic

Ready to dive in?

Follow step-by-step guides on ingesting data ready to apply semantic search and running semantic search queries in Elastic.

References

1. Google survey what business leaders expect from generative AI, published May 26, 2023
2. From zero to semantic search embedding model, Roman Grebennikov, Jun 12, 2023
3. Text Embeddings by Weakly-supervised Contrastive Pre-training, Lian Wang et al, December 2022.
4. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models, N. Thakur et al, UKP Lab at Technical University of Darmstadt, 2021

wp-semanticsearch-2023-1016