algolia

A CTO's guide to end-to-end AI Ecommerce Search

How native AI search reduces workloads while improving understanding and user experience...

Simplicity is critical for search engines, but building one that enables that simplicity is complex. Over the last 20+ years, most on-site search solutions have enabled matching a customer's query to a search index. Then starts the job of tuning the engine. That's because search engines can't understand the searchers' intentions very well.

Is someone who searches for "what is rust" looking for the corrosion process, a computer programming language, or something else? For years, anyone who has had to optimize a search engine has used either manual rules, synonyms, keyword stuffing, or some other basic AI to "patch" the matching to achieve understanding.

Building end-to-end AI was possible, but impractical to operate at scale due to cost and performance challenges.

This just changed with Algolia NeuralSearch, which includes native end-to-end AI processing.

MATCHING

Keywords

(2000-present)

Match the input keyword with indexed product catalogue

PATCHING

Hacky AI

(2013-present)

+ Rules, synonyms
+ AI hacks
+ NLP Add-ons

DABBLING

Vectors

(2016-present)

First generation ML/AI. Better quality results, but very expensive and/or very slow at scale

Neural Hashes

2023

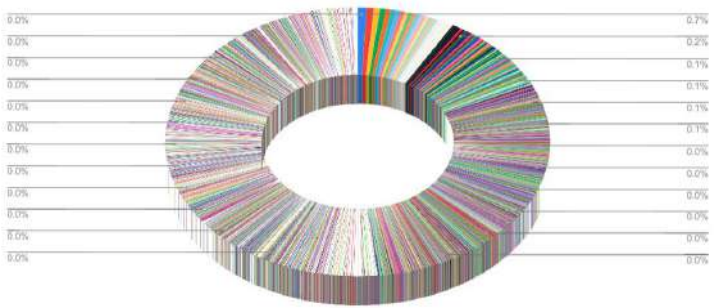
algolia

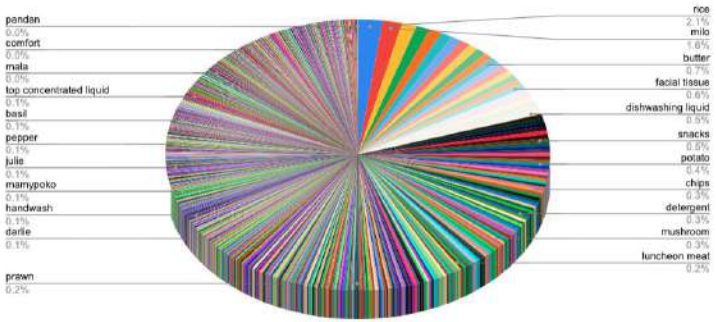
Personalization
Learn to Rank
Adaptive AI
Business control

Why end-to-end AI matters

It is likely a whole topic in itself, but let us align on the real need on why end-to-end AI matters before we go further into what it is. Data suggests that 50-70% of revenue is left on the table due to unoptimized long-tail. Theoretically, if one had unlimited time and resources, one could manually optimize it all — but that's beyond human capacity.

For example, let's look at the search profile of two large retailers in the UK and the US, respectively, a very good representation of the majority of online businesses. Can anyone manually handle this kind of longtail? Can this be optimized by patching AI on the top and bottom of the core engine? No, That's just impractical.





There is a staggering number of product variations typically found on ecommerce sites, which makes it impractical — not to mention nearly impossible — to optimize every term for long tail queries.

That's where the need to take the machine's help arises. Machine learning algorithms can optimize all 100% of search catalogs. Until now, that wasn't easily possible, but with the maturity of AI algorithms, richer data, and more powerful cloud computing we can now apply AI across any size catalog. Deploying AI end to end, right from the point of a consumer typing the query till the time results are served, is the solution to this challenge.

So why is end-to-end AI needed? Until now, most AI solutions have only focused on solving for discrete parts of the search query. For example, query processing — the first step of a search query result — or ranking, which typically comes last.

The missing link was using AI for retrieval. Retrieval is when the search engine attempts to find the most relevant information. Arguably, AI has the biggest impact in the quality of results during retrieval. With the maturity of AI machine learning today, it's something that can finally be tackled.

We'll explain further how each stage of AI works to deliver better results.

Query Understanding

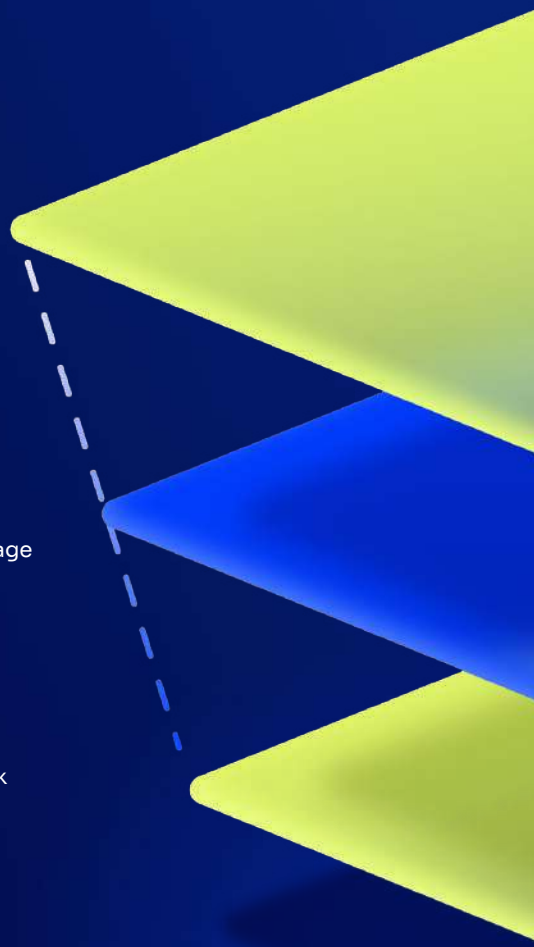
AI powered query categorization
Zero set up for rules, synonyms

Retrieval

Neural Hashes
Popular keywords to free form language

Ranking

Merged keyword & vector results
Personalization with Learning to Rank



True end to End AI

Algolia NeuralSearch sends each query through three phases to deliver the best results. We call it end-to-end AI search to differentiate from competitive solutions that may only offer AI at specific phases of processing (ie, “Hacky AI”).

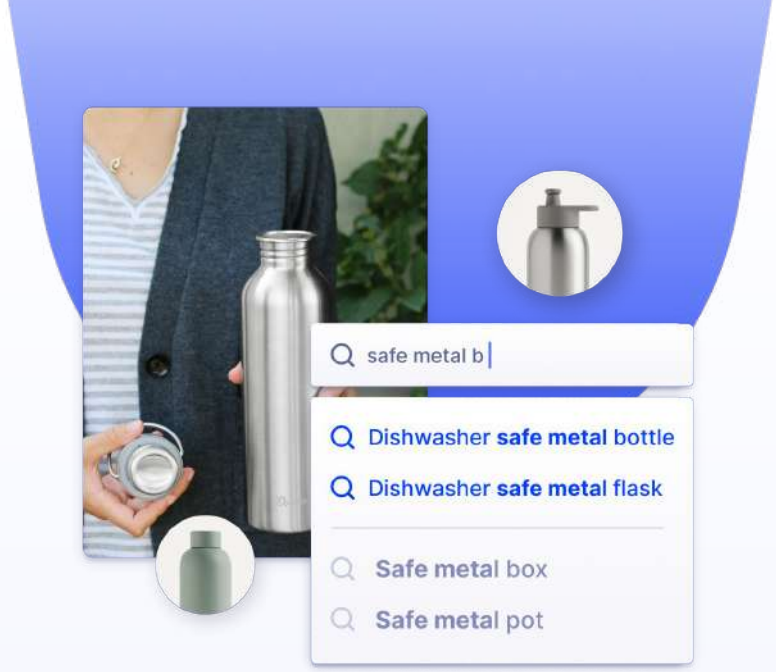
These phases include: query understanding, retrieval, and ranking. Each of these discrete capabilities provide value to the final results.

Query understanding

Query understanding consists of natural language processing (NLP) and natural language understanding (NLU). NLP is involved with parsing and processing the language in a search bar to prepare a query for execution. The process could be as simple as comparing the query exactly as written to the content in the index. Classic keyword search is actually more advanced than that, because it includes tokenizing and normalizing the query into smaller pieces – i.e., words and keywords. This process can be easy (where the words are separated by spaces) or more complex (like some Asian languages,which do not use spaces, so the machine needs to recognize the words).

NLU is focused more on determining user intent.
It uses machine learning intent classification, semantics, synonyms, and ontology expansion to understand words and phrases. This step improves the interpretation and understanding of the query.

For example, two queries can use the same terms to mean very different things: “Dishwasher-**safe** metal bottle” and “Fire **safe** metal box”



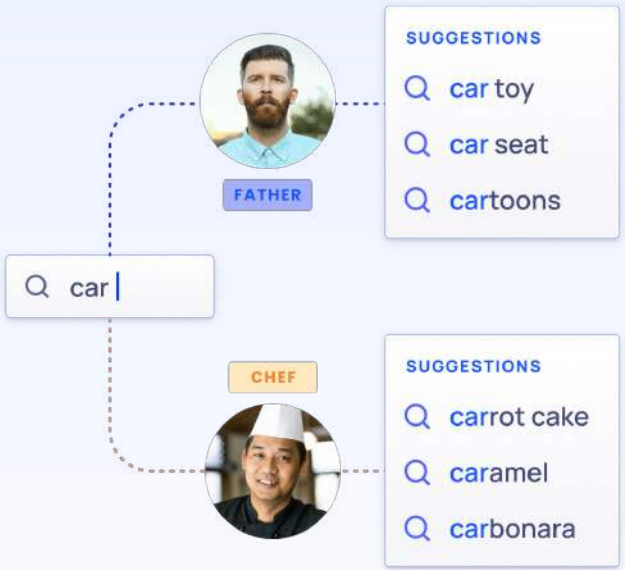
The first case is an example of using the term “safe” as an adjective for a water bottle, whereas the second is a noun — a metal safe for storing valuables. In these particular examples, NLP tokenization is used to help break down each phrase, and NLU helps interpret the meaning of each query. This is true for both text and voice search, but voice search can get trickier with homophones like “pear”, “pair”, and “pare”, for example.

Semantic search

Semantic search requires a different approach, however. It is typically powered using large language models (LLMs) to create vector embeddings. Vectors are mathematical representations of words, images, videos, or just about any type of object you might want to compare. Machine learning models determine semantic relationships between objects in an index. Vectors can have hundreds, or even thousands of dimensions, which can be used to determine similarity. It's hard to visualize that many dimensions, so most of the time we use three-dimensional representations like the image shown to the right.

Vector search can connect relationships between words, and similar vectors are clustered together.

Vectors can be combined (or subtracted) to create new meaning and understand relationships; the classic example is "King -man = queen".



Retrieval

Better information retrieval can improve both search **precision** (result accuracy) and **recall** (result completeness across your entire index). It's hard to strike a balance between precision and recall; improving one can have the inverse effect on the other.

Keyword search engines tend to provide better results for head terms, whereas AI vector search works well for less popular long tail queries. You need to solve for both, which is sometimes called **hybrid search**.

Keyword search typically relies on statistical matching using BM25 or TF-IDF and related techniques for ranking results based on the query terms that appear in each document. For example, TF-IDF looks at the inverse frequency of a word in a document (IDF) versus the term frequency of a word (TF) to determine its importance. Stop words like "the" "and" "or" that show up frequently everywhere, whereas words like "toothbrush" or "water" that show up less frequently — that is, they are more uncommon and likely more informative about the content. Term frequency can be used as a proxy for how important, or relevant, the document is. Most of the models required to build keyword search are available. Many companies still use Lucene, the venerable Apache project that spawned both Elasticsearch and Solr.

Vectors are huge, typically require specialized GPU-powered databases, and are expensive to scale while also being performant. Techniques such as HNSW (Hierarchical Navigable Small World), IVF (Inverted File), or PQ (Product Quantization, a technique to reduce the number of dimensions of a vector) are some of the most popular Approximate Nearest Neighbor (ANN) methods to find similarity between vectors.
Each technique focuses on improving a particular performance property, such as memory reduction with PQ or fast but accurate search times with HNSW and IVF. It is common practice to mix several components to produce a 'composite' index to achieve optimal performance for a given use case.

Even with most approximate nearest neighbor (ANN) techniques, there's **no easy way to design a vector-based search algorithm** that's practical for most production applications. For example:

- ➔ Insert, update, and delete functions can challenge graph based structures like HNSW, which **make deletion very complex**.
- ➔ Sharding vector indexes **is very expensive**. HNSW performance drops off non-linearly with more shards.
- ➔ **Combining keyword and vector results is really hard** because the relevance computation in each system is completely different, so we can't compare them easily, which is why you see many people bolting a vector ANN index on top of a keyword engine. Essentially, vectors are only being used as a "fall back" option to prop up results when keywords fail.

At Algolia, we pioneered a new technique called neural hashing. Essentially, we use neural networks to compress vectors into tiny binary files that still retain most of the information. The binary vector is much smaller and faster to calculate than standard vectors, which allows us to run vector and keyword search simultaneously and at scale on commodity CPUs. We then normalize and combine the results to provide a single measure of relevance in the same query.

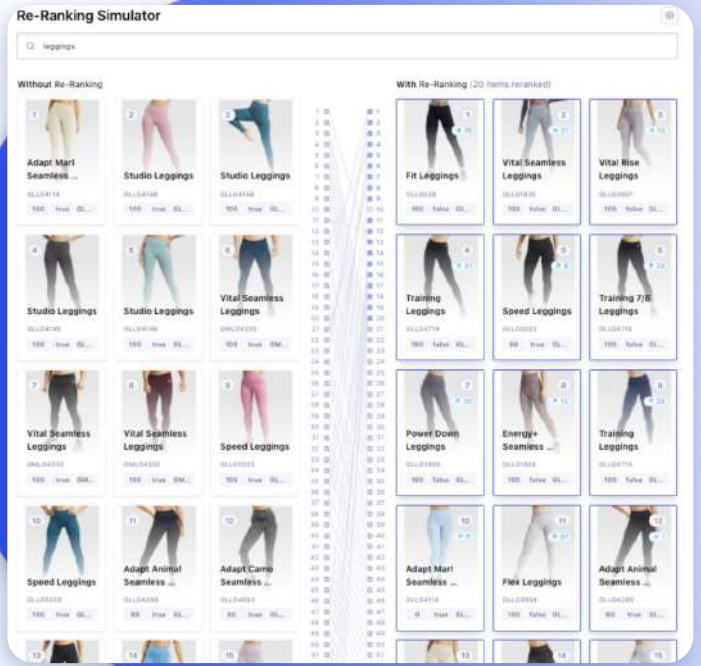
These tradeoffs, plus the cost associated with scaling vector search, make building AI relevance outside the scope for many organizations.

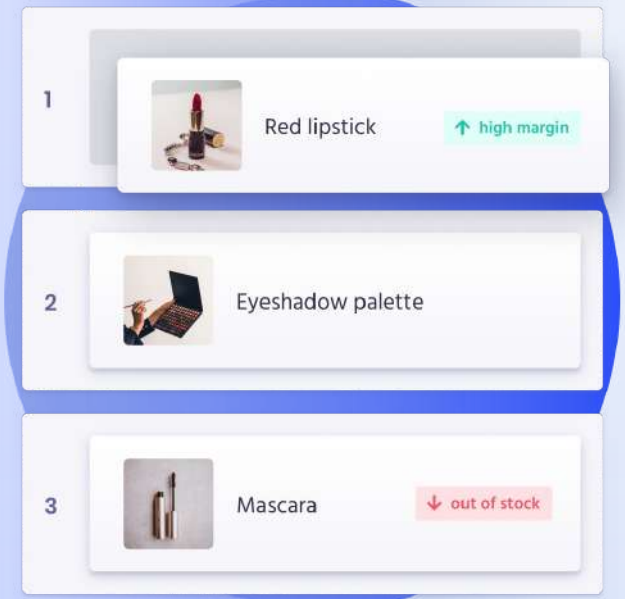
Site and app search is just one application for vectors. It is worth mentioning here, that vectors can also be used to compare many different kinds of objects such as images, video, symbols, or even entire documents, and have many different real world use cases such as movie recommendation systems, facial recognition, fraud detection, and more.

Ranking

Product ranking can make a huge difference in search success. There is position bias in result ranking — the first three results get significantly more clicks than lower ranked ones. There is also a need to constantly re-rank results; results that rank high today may need to be buried tomorrow if something more relevant or popular becomes available.

If your retrieval is good (see above) ranking starts from a better position. Learning-to-rank (LTR) , sometimes also referred to as Machine Learned Ranking (MLR), is a type of machine learning that improves ranking and assists with precision. It includes supervised, unsupervised, and reinforcement learning. There are also variations like semi-supervised learning. Each of these solutions offers AI ranking capabilities to deliver improved results over more simpler statistical methods.





While the machine learning involved seems like maybe an easier challenge to tackle than relevance, it may not be. It took AirBnB, a team of very talented data scientists and engineers, two years to build an AI ranking solution!

In recent years, learning-to-rank techniques such as reinforcement learning have become more popular because they reduce maintenance and iteration.

Reinforcement learning uses events such as clicks, conversions, purchases, etc., to re-rank results from most to least popular. It can also do the opposite — bury results that aren't performing well. The challenge for ranking is data. Reinforcement learning normally requires a lot of data to work, but newer models are coming out that enable better prediction with less data.

Should you build or buy end-to-end AI search?

Each time you've faced a buy vs build decision your team of talented engineers has told you: "hold my beer, we've got this." They argue "why spend hundreds of thousands on a SaaS vendor when our team can roll out an open source alternative on AWS and run it ourselves?" And it has typically worked, saving the company money that can be reinvested back into hiring more engineers.

AI is changing everything.

Top 6 reasons why you should buy your AI service rather than build it yourself

1. Talent

It's really hard to build a great data science team

Hiring a team of talented data scientists is really hard. The race for AI talent has driven up the overall costs of hiring and retaining staff, particularly from top university PhD programs. Are your company's pockets as deep as big tech, and can your company's brand compete with the Google's and OpenAI's of the world?

There is a risk you might end up with a below average data science team if you can't win the war for talent.

2. Speed

It takes a long time to develop data science capabilities.

It takes a long time to build a team from scratch, with hiring lead times at 6mo+. It takes data scientists an especially long time to onboard, as the complexities of the work and data require experimentation to gain experience.

It takes even longer to mature a new capability like data science inside your organization: building tools, processes and capabilities that never existed before. You should expect 2 years or longer from the decision to hire to having the first products live. Can your company wait that long?

3. Culture

You don't have a culture of experimentation

To succeed in building AI, you need to have a culture that embraces scientific experimentation. **Learning in production is the only way to build and optimize models**. This culture is very different from building software, where the solution (and delivery timelines) can often be explained in great detail before a line of code is written.

AI, on the other hand, means failing (repeatedly) in production and not knowing if the next experiment is going to work or not. Many companies simply can't develop and nurture this culture.

Top 6 reasons why you should buy your AI service rather than build it yourself, cont.

4. Data

You don't have enough data to compete

You need a lot of data to train AI models from scratch. It is quite likely you don't have as much data as a vendor who is able to leverage the data from thousands of customers to create a model that performs significantly better than yours.

Without scale in data it takes a long time to test our new experiments and your algorithms will improve slowly. Data is the lifeblood of an AI team and most companies simply don't have the scale to compete.

5. Agility

AI is moving very fast, can you keep up?

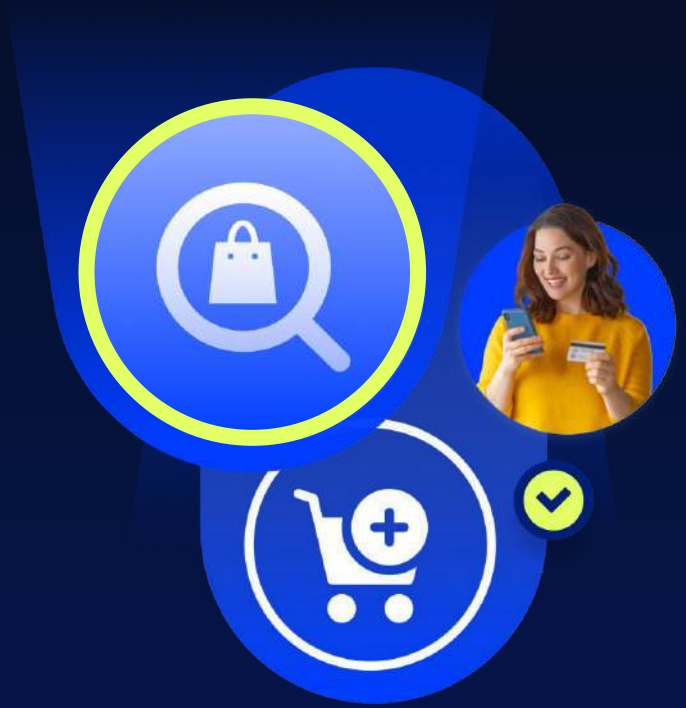
The AI industry is moving extremely fast. If you choose to build your own capabilities, you'll only be able to adapt as fast as your team. Can they keep up with the latest research papers, attend the conferences and keep on the bleeding edge of what's possible?

By making the choice to buy rather than build, you can adapt through vendor selection and piggyback on your vendors' AI teams' knowledge of the cutting edge.

6. Reliability

It's hard running AI systems in production

Running AI systems in production is extremely challenging. **They require high speed, real time pipelines of data and large amounts of compute resources.** When demand spikes, are you able to scale up quickly and then ramp down before the costs pile up? Do you have specialist reliability engineers that know how to run and quality control AI models?



The AI vendor advantage

Vendors can aggregate demand from a range of customers and spread the load with spikes happening at different times. Vendors can afford to hire specialist reliability engineers to run AI systems at scale.

Even Microsoft — a company with the resources and talent to build AI — chose to buy part of their AI solution for Bing.

The reasons for buying instead of building an AI solution check the same boxes as gaining the competitive advantage. You can match the talent and speed of the biggest names in tech while gaining the insights and system reliability that enable your organization to sustain the pace of business in the age of AI.

ChatGPT vs Search

Since OpenAI announced ChatGPT and people started testing it, there have been plenty of breathless proclamations of how it will upend everything. One of those up endings is search.

But, really? No. Here's why.

Query Formulation

ChatGPT responses are heavily dependent on the prompt (i.e., query) formulation. OpenAI lists this as a "limitation". Typos and small tweaks in language can cause ChatGPT to fail. Search must understand searchers even with misspellings, or else the experience is a step back. Understanding and matching different spellings is difficult.

Experience

Search layouts have been, typically, the same for decades. Specifically: a set of results in order from most to least relevant (however that is measured). It's important for searchers to be able to see options, filter down with a click, and generally get deep into a "discovery phase." This is not what chat is suited for.

Speed and cost

Speed is important when it comes to search. Amazon and Google have both done studies showing lags of even 100 milliseconds can cost millions of dollars in revenue. ChatGPT will someday be faster, but for now, it's not even close.

So, in short: ChatGPT is great, particularly for conversational commerce use cases, but for a predictable and fast experience, customers will still choose search.

Take your site experience to the next level.

Get a customized demo from our AI search experts today.

GET A DEMO

www.algolia.com

algolia