coveo™

Coveo Platform Architecture Guide

White Paper

Contents

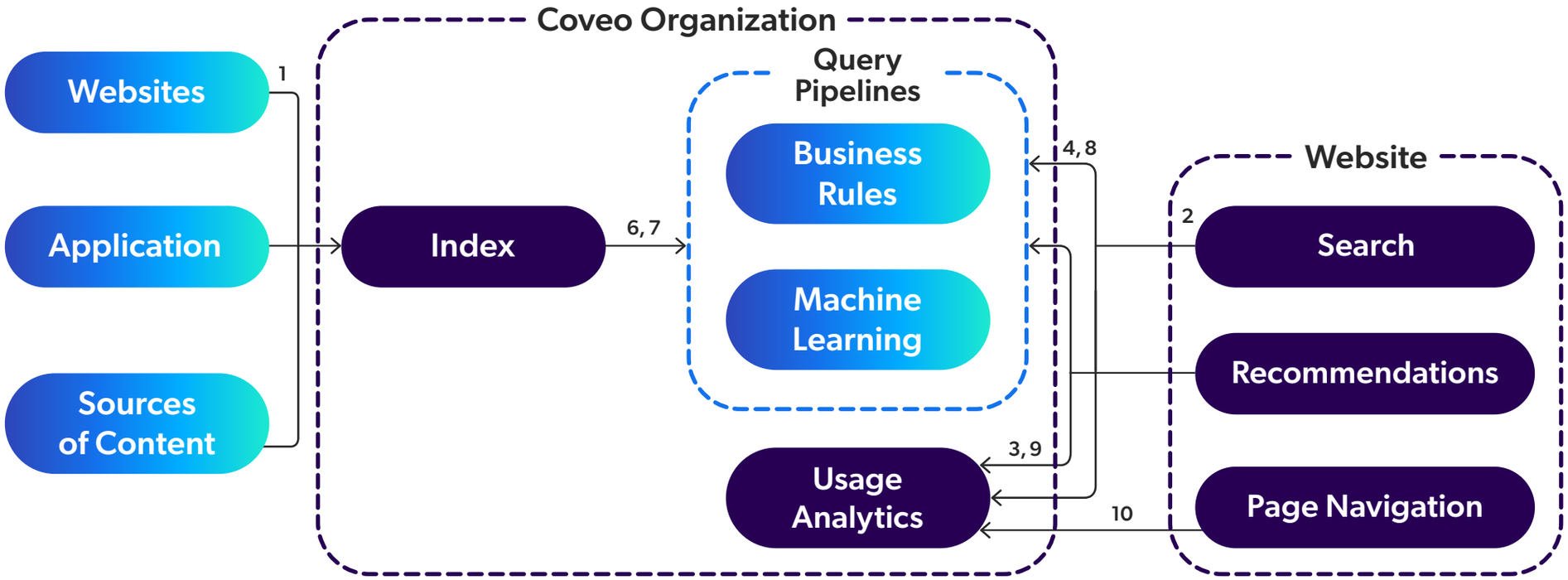
- Introduction
- Coveo Architecture
- Planning the Implementation
- Tracking Data
- Indexing Content
- Serving Content
- Relevance
- Common Pitfalls
- Additional Resources

Introduction

Coveo is a highly scalable, highly extensible, cloud-based, enterprise AI platform that uses semantic search, AI recommendations, and GenAI answering to provide individualized, connected, and trusted digital experiences.

This document is meant to help you understand how Coveo works and how it fits within a larger implementation project. It is aimed mostly at project managers and solution architects. Developers wanting to know how to implement Coveo should follow the **Platform Developer Certification** on Level Up.

Coveo **Architecture**



The Coveo architecture can be summarized as such:

1. Coveo integrates content from many different sources using its connectors, and consolidates the information in a single, unified index that lives in your Coveo organization.
2. A query is sent to Coveo, either due to a user searching or through recommendation or listing components wanting to display content to users.
3. In parallel, a second call is made to Coveo Usage Analytics, indicating a query was performed. This data is what allows Coveo Machine Learning to learn.
4. The query goes through a query pipeline in the Coveo Platform.
5. The query pipeline modifies the query according to business rules and machine learning.
6. The query reaches the index, where Coveo determines which results should be returned.
7. The results go back through the query pipeline, where additional rules and machine learning can modify the result order to be more relevant.
8. The results are returned to the end user.
9. When a user clicks on a result, a call is made to Usage Analytics, indicating which results were successful.
10. When a user navigates through the site, even without the help of Coveo, page view events are sent to Coveo, so Machine Learning can learn.

Planning the Implementation

A successful Coveo implementation requires a few core components:

Good and accurate index data

Complete front-end integration

Robust user interaction data collection

Without one of these core components, a Coveo implementation will struggle to provide relevant content to your customers. With all three, you can expect Coveo's Machine Learning capabilities to greatly improve the user experience on your site.

Team Expertise

To bring a Coveo project to completion, you need people with a certain type of expertise. You want at least one person who can perform each of the following tasks:

- Define business requirements and expected outcomes
- Design an architecture to the project so every aspect is aligned and business requirements are all taken into account
- Understand each system that needs to be indexed, with the right permissions, so they can be indexed in Coveo
- Build front-end components and pages using either the **Headless** or **Atomic** frameworks from Coveo
- Send analytics data from the front-end to improve relevance
- Design and build ranking rules such as promotions on the Coveo platform

Of course, some of those tasks can be assigned to the same person, and some may be distributed to more than one. However, all of these tasks need to be covered by at least one person. Additional tasks may also be needed, such as QA testing, post implementation usage analysis, more robust project management, and more advanced business analysis.

Scoping

A Coveo project normally comes in five major phases, using the **IDEAS methodology**:

- **Initiate:** Defining the business objectives.
- **Define:** Defining the technical requirements to attain these objectives. This includes building an architecture and defining which sources and frameworks are going to be used.
- **Execute:** Implementing Coveo in the system. This is when the sources are indexed, the search pages and components are created and integrated, the data collecting is implemented, and the machine learning models are created. This is the core of the implementation.
- **Appraise:** Ensuring the implementation answers the business objectives. This is the phase when you perform quality and performance assurance for the implementation, and tune it appropriately.
- **Success:** Going live with Coveo. This is also the stage where you normally hand off the project from the implementation team to the business or maintenance team.

For each phase, you will need to dedicate enough resources so they can be executed appropriately. Missing one of those steps is likely to lead to an incomplete implementation or a dissatisfied customer.

Timeline

The length of a Coveo project is typically counted in weeks. There are a few important factors that determines the length of the project:

- Number of indexed sources
- Source complexity compared to the out of the box configuration
- Number of search interfaces
- Front-end framework used to build the interfaces
- Number of custom non-out-of-the-box front end components, if any
- Number of supported languages and regions
- Whether there are secured or private documents
- Whether the permissions are native to the system where the front end is added
- Whether the project contains commerce components, such as add to cart
- Amount of Coveo experience the implementation team has

The more requirements you identify with these questions, the longer and more complex a project becomes. It is not rare to see even a simple website integration take 3 months to complete, especially with implementers who have no prior experience with Coveo.

The best way for developers to get trained on Coveo is to complete the **Platform Developer Certification**. If you would like more tailored training, please contact your Customer Success or Partner Manager.

Tracking **Data**

A Coveo implementation can contain two types of data tracking: standard events, and commerce events. The latter is exclusively used for commerce projects.

Standard events consist of three types of events:

- **Search events:** sent when a query is sent to Coveo.
- **Click events:** sent when the user clicks on a result.
- **View events:** sent when a user loads a page on your site, even if they did not reach it via the search page.

Search and click events are sent when interacting with a Coveo search interface. When using a framework like Headless or Atomic, there are built in components that take care of sending the appropriate events with the appropriate data.

View events need to be sent using the **coveo.analytics.js** library. They require you to send metadata that identifies the page in the Coveo index. This is ideally done using the permanentid field, but it's also possible to leverage the SKU (in case of a product page) or the URL of the page.

Indexing **Content**

For Coveo to provide relevant content, that content needs to first exist in your Coveo index.

To get content into your Coveo index, Coveo provides a list of available sources, such as Sitemap, SharePoint, or SAP. You can find the list of available sources in the **Connector Directory**.

Coveo can index any type of content into your index, even if there is no native Coveo connector for it. The exact method of indexing depends on the content you are indexing.

Push vs Pull

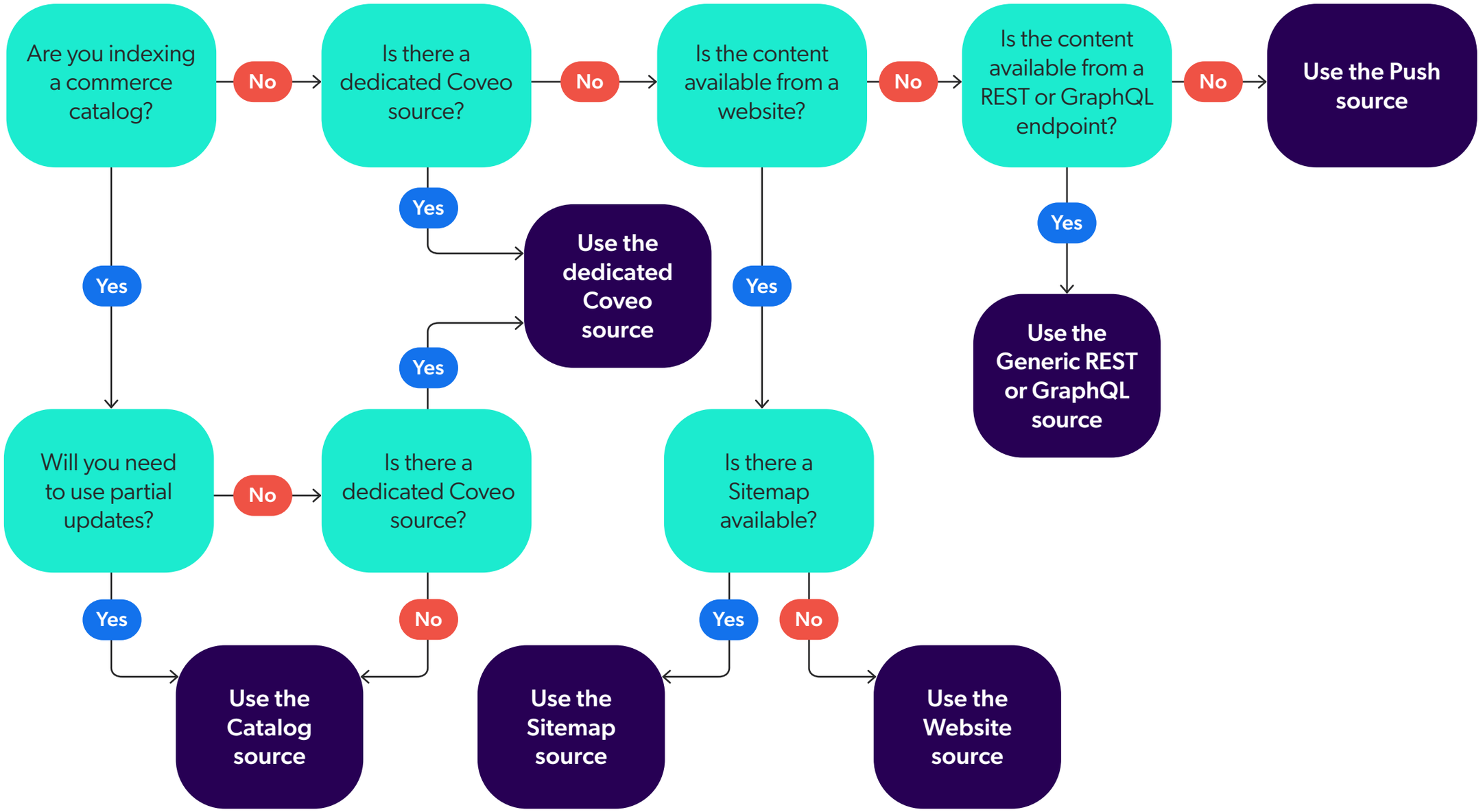
While Coveo offers many sources, they all work in one of two ways: either as Push sources, or as Pull sources. The difference between both is which side initiates the indexing process.

With Push sources, Coveo opens an API where you can send (or push) your items to be indexed. Push sources are typically simple on the Coveo end since most of the complexity is in the code that performs the push.

There are currently four Push sources: Sitecore (controlled by Coveo for Sitecore, which is installed on the Sitecore server), the Push API source, the Catalog source, and all sources controlled by the Crawling Module, a piece of Coveo software that can be installed on servers to push content from behind a firewall.

With Pull sources, you tell Coveo which page or endpoint needs to be accessed, and the frequency at which you want Coveo to visit those pages or endpoints. The vast majority of Coveo's sources use the pull mechanism.

Indexing Method	Push	Pull
Description	Something external to Coveo pushes content to a Coveo endpoint	On a specific schedule, Coveo calls an endpoint to index content
Example sources	Catalog, Sitecore	Sitemap, SAP
Generic source	Push API	Generic REST
Advantages	• You have full control over the content being indexed • You can update the content in Coveo as soon as it changes in the original source	• Coveo is in charge of the infrastructure and the scheduling • Configuration is all in one place
Disadvantages	• You need to control when content should be pushed • You typically have an extra piece of software to maintain • Typically takes more development time to configure	• Since the source refresh happens on a schedule, your content typically needs at least a few minutes before it gets updated


Are you indexing a commerce catalog?
No
Is there a dedicated Coveo source?
No
Is the content available from a website?
No
Is the content available from a REST or GraphQL endpoint?
No
Use the Push source
Yes
Use the dedicated Coveo source
Yes
Yes
Yes
Use the Generic REST or GraphQL source
Will you need to use partial updates?
No
Is there a dedicated Coveo source?
Is there a Sitemap available?
Yes
No
Yes
No
Yes
No
Use the Catalog source
Use the Sitemap source
Use the Website source

Partial Updates

One of the advantages of using a Catalog source is having access to partial updates.

Partial updates allow you to send only part of a document to modify it in the index, as opposed to sending the whole document again. This functionality is currently only available with the Catalog source for commerce customers.

In some commerce projects, the Coveo item needs input from many different systems, and it is not realistic to pull all of that information when you only need to update a single field, such as price. This is where you will want to implement a partial update mechanism. Be aware that this normally needs to be done on top of the normal Catalog streaming indexing process.

Permissions

Documents can hold sets of permissions, ensuring only people who can access content can see them in search results.

Permissions are set at the item level when indexing, and are set on the search token when querying.

Coveo's permissions can be individual (user-level) or group-based. For example, you could say that a document can only be accessed by jlpicard@example.com, or you could say that anyone in the group SALES can access a given document. The same applies to denied permissions.

Serving **Content**

Once you have items in your index, you are ready to start building the front end search pages and components to serve that content to end users.

Using a Framework

Coveo offers different ways of building pages.

The two most recommended options are **Coveo Headless** and **Coveo Atomic**.

Coveo Headless

Headless is a framework that acts as a layer on top of the Coveo APIs, giving you access to controllers that simplify the generation of search pages. With this framework, you have full control over the rendered HTML, but it does mean that you have to create all of the visual components.

Coveo Atomic

Atomic is a set of visual components built on top of Headless. Atomic allows you to create powerful pages with HTML tags and very little JavaScript. Note that it is also possible to access the Headless layer when using Atomic, for more advanced use cases.

Search API

It is also possible to search Coveo directly via the API. However, doing so comes with its own set of risks, as you need to handle both the Search API and the Usage Analytics API endpoint, and pass the right parameters in both. Going this route usually greatly increases the development time required for the implementation.

Search Hubs

No matter the solution you choose to search content, there is an important concept to understand: search hubs.

Search hubs are the way Coveo knows which search interface your queries are coming from. Search hubs should be human-readable values, and can contain spaces. You normally want it to be prefixed by the website or brand name, followed by its purpose, and ending with the region and language when appropriate.

There are three valid purposes for search hubs:

- **Search:** used for fully functional search pages, where users can use a search box to find items.
- **Listing:** used for pages using search to power a listing of items in order of relevance for the user. This is common for category pages on commerce websites, for example.
- **Rec:** used for recommendations, where a carousel or something similar is recommending items from your index based not on a query but on the journey of the user.

The purposes are used by Coveo Machine Learning, and should be those words exactly.

For example, the search hub **ACME Search Canada En** would be a valid search hub for the brand or site ACME, on their search page, for the Canada region, in the English language.

Each search page should have its own search hub value.

Each listing page on a given site should have the same search hub, and use the Tab/OriginLevel2 parameter to differentiate between each listing page.

Each recommendation component on a given site should have the same search hub, and use the Recommendation parameter to differentiate between each individual recommendation component.

Search Box Suggestions

The preferred way to show suggestions in a search box powered by Coveo is by using the **Query Suggestions** Machine Learning model. With this model, users will be presented with queries other users have made in the past, trying to autocomplete the query. It is the most typo-resistant and most relevant way of providing suggestions to end users.

In some scenarios, you may want to propose results directly as the user is typing. To do so, you are encouraged to leverage the InstantResults component (available in **Headless** and **Atomic**). Instead of making one query per keystroke using a wildcard to try and autocomplete the word, InstantResults takes the first query suggestion from Machine Learning, and returns the top results for that suggestion, benefitting from the smart auto completion of the model.

You are strongly discouraged from making one query per keystroke to provide results in the search box, as a full search request takes longer to return compared to a query suggestion call, and a full search request normally counts towards the organization's license entitlement, making it a more expensive use of Coveo.

Relevance

With content in your index and a way to present that content to end users, what remains is to tweak the relevance, using a mixture of business rules and machine learning.

Query Pipelines

Query Pipelines are where rules can be added to modify a query or alter the results. For example, this is where Machine Learning can influence results, and where your business rules take effect.

Query Pipeline Routing

To determine which pipeline to use, you are encouraged to set conditions on the pipeline directly, using the search hub value.

Query Pipeline Features

Query pipelines offer many different features to add business rules. Here are the most commonly useful:

- **Thesaurus:** Add synonyms, so Coveo can expand or replace the appropriate words. This is especially useful to expand acronyms for example.
- **Featured Results:** Pin a result at the top of the result page under a specific condition. This is useful when you have a specific page you want to make sure is at the top of the page for a specific query, e.g., having the registration page as the first result when someone searches for your in-person event.
- **Ranking Expression:** Boost or bury a category of results by a specific number of points. For example, this can be used to make products on sale appear higher in the result list, or to make archived documentation appear lower.
- **Filters:** Limit the content shown to end users on specific pages. Server-side filters in the query pipeline are usually preferred to front-end filters. With server-side filters, you can also make quick changes without needing to deploy on the front end.

Machine Learning

Coveo offers multiple Machine Learning models, which you can leverage to enhance the experience on your site.

In order for a Machine Learning model to learn from data you are collecting, it simply needs to be created. However, for it to affect your site, you need to make sure it is associated with the appropriate query pipeline.

Coveo separates learnings according to three main criteria: language, search hub, and tab. Those criteria act as hard separation; learning from one search hub will not be shared with another search hub.

You can also send custom contexts to perform "soft" separation of learnings, where the models learn from all events but weigh similar contexts together, as noted earlier.

Automatic Relevance Tuning

Automatic Relevance Tuning (ART) is a Machine Learning model that learns from search and click events to boost results based on how successful they are for specific queries.

By default, ART boosts up to 5 results. This is configurable on the model.

Query Suggestions

Query Suggestions (QS) is a Machine Learning model that, like ART, learns from search and click events, to suggest successful queries to end users as they are typing.

Intent Aware Product Ranking

Intent Aware Product Ranking is a commerce-specific Machine Learning model that reorders the results based on the user's journey cross-referenced with your catalog.

Predictive Query Suggestions

Predictive Query Suggestions (PQS), like QS, is a Machine Learning model that suggests queries to end users as they are typing. The main difference between both models is that PQS takes the catalog of your index into account, so that the suggestions are based off of which products were recently viewed by the user.

Dynamic Navigation Experience

Dynamic Navigation Experience (DNE) is a Machine Learning model that handles facet and facet filtering. It can do four different things, depending on how you configure it:

- Reorder the facets on the page so the most relevant ones are shown first
- Reorder the facet values within the facet
- Auto-select facet values based on the query
- Auto-boost results from a given facet value based on the query

DNE requires click and search events, and needs to have the facet clicks logged appropriately. This is normally handled automatically when using one of the Coveo frameworks.

Content Recommendation

Content Recommendation (CR) is a Machine Learning model that recommends content from your index, typically in its own interface or carousel.

CR requires page views in order to suggest content. Without page views or with too few events, it will by default not display any content.

Product Recommendation

Product Recommendation (PR) is a Machine Learning model that recommends products from your index. This model offers a variety of different strategies you can leverage, such as the cart or user recommender, or frequently bought or viewed together.

Each strategy requires different events to be sent. You can find the full list on the **Product Recommendation documentation page**.

Smart Snippets

Smart Snippets is a Machine Learning model that learns from your indexed content to propose a solution to the user's question directly from a document, and propose follow up queries.

Relevance Generative Answering

Relevance Generative Answering (RGA), sometimes also referred to as GenAI, is a question answering AI model that leverages GPT alongside your indexed content to answer a question from the user and citing its sources.

Common **Pitfalls**

When developing a search experience, there are some pitfalls to avoid. The following examples will help you avoid these pitfalls as you are building your experience.

Search As You Type

Search-as-you-type requires many queries to be efficient. If the threshold is too high, the user will probably go faster than the refresh rate of your interface, and if the threshold is too low, the interface will flicker and move really fast.

Instead, Coveo recommends using query suggestions, which recommends queries instead of content, as the user types. This solution is much faster, and does not count towards the allocated queries per month on a license.

You may also want to leverage Instant Results, a feature that returns results for the most relevant query from query suggestions as the user types. This solution will consume queries, but tends to yield relevant and accurate results.

Using Rules Without Data

There are several tools to manage ranking. Those tools should be used based on data, such as what is reported in the Coveo Usage Analytics. Using them blindly can have a disastrous effect on the relevance of the solution.

Coveo recommends adding as few rules as possible before going live, and testing additional ranking rules using A/B testing, to ensure the rule is improving relevance and not hindering it.

Wildcarding

Wildcarding is an option where every word starting with the characters entered in as input will be retrieved. A double wildcard can find the characters entered in any position of any word in the index. Wildcarding tends to be taxing on the index, taking a much longer time to return results, without necessarily yielding appropriate results.

Coveo normally recommends using a mix of query suggestions and ART machine learning models to cover autocompletion.

In some scenarios, such as SKU search, wildcarding can be replaced with SKU decomposition, as a much more quick and efficient method of completing words.

Additional Resources

Coveo Docs

Coveo Docs is the official documentation for all Coveo products and features. Use the search bar at the top of the page to find the topics you need when developing with Coveo.

Coveo Level Up

Coveo Level Up is the education portal of Coveo. Register for learning events, complete training courses and tutorials, and compete with other people at your company to be at the top of the leaderboard.

Coveo Connect

Coveo Connect provides access to the Coveo community, where you can ask questions to the community on the forum to get answers to your questions. It is also where you can open tickets with Coveo Support.