Coveo for Sitecore Project Guide

A Coveo Guide to High-Quality Project Delivery & Maintenance

VERSION 2
JANUARY 2022

Table of Contents

What is on the menu

Introduction

Welcome to Coveo for Sitecore

This document is intended for everyone planning to deploy Coveo for Sitecore, regardless of their experience with the software. It provides guidance to lead projects to success, and includes a variety of tips and tricks used by Coveo Support, Services, and R&D teams.

Coveo's current core product offering is cloud-based, hence this guide is only considering cloud implementations using Coveo for Sitecore version 5.

This guide does not intend to replace product documentation, but to share best practices for a successful implementation. To learn more on a specific subject, follow the links to the detailed articles. For a step-by-step learning experience, refer to the developer training.

EDITOR NOTES

Watch for the ✓ sign beside a title. It indicates strongly recommended topics.

All diagrams use the Business Process Modeling Notation (BPMN).

Some pro-tips have also been sparsely added in the document:

PRO TIP: These tips are technical advice, based on field experience.

Understanding Relevance, Understanding Coveo

Content indexing and search is a mature market. Several solutions, whether open source or proprietary, have paved the way to modern search and are now fighting for the best performance and speed under all conditions.

Coveo, on the other hand, is no longer focusing on being the best search solution on the market, but instead being the most relevant at all times.

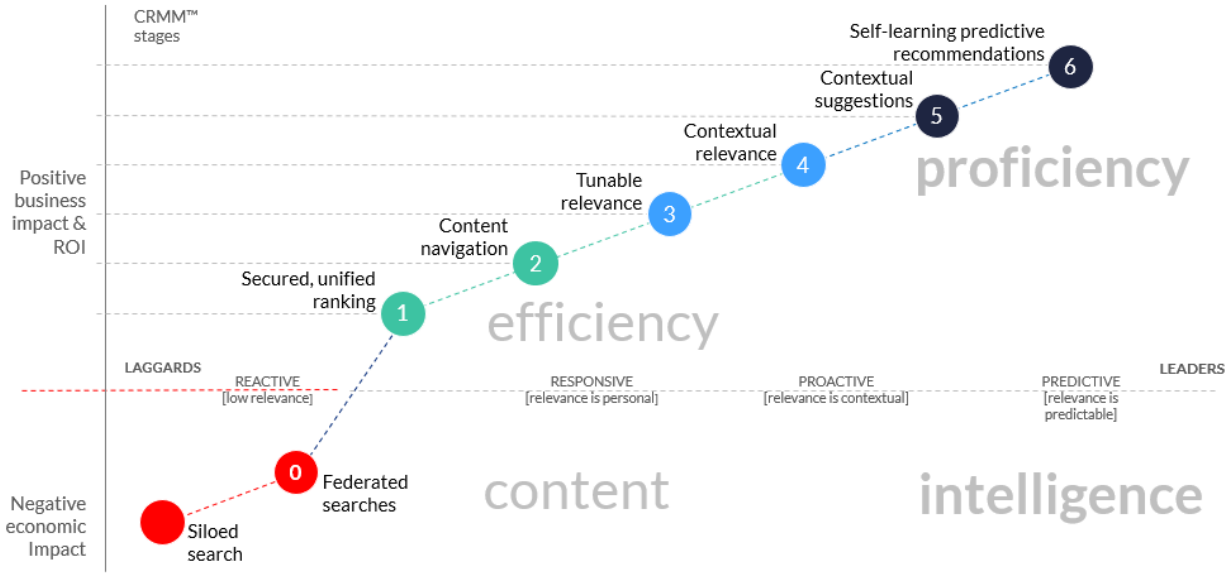
Hence, the following would be an accurate definition of Coveo:

Coveo is a relevance engine leveraging search for understanding user intent and learning from it to provide increasingly relevant interactions.

To be successful at implementing Coveo, one needs to understand that indexing and querying content is only the first step to a successful deployment. Experience with a content indexing solution — integrated in Sitecore or not, such as Solr, Lucene, Azure Search, Elasticsearch or others — can be helpful, but is not required to master Coveo.

Coveo is not more complex than the previously mentioned solutions; it is simply different.

Coveo Relevance Maturity Model™



The Coveo Relevance Maturity Model helps to visualize the journey to predictive relevance. Using Coveo as an index will get your search to the first step, and leveraging the integrated search framework will help the solution climb up to the second stage.

From there, several ranking tools will allow you to climb up to the third, fourth, and fifth stages. Machine Learning, which is included in every Coveo organization, will allow your solution to reach step 6 and stay up there even if your customer behaviors and habits change.

This guide will help you build that experience and fine-tune Coveo accordingly.

Preparing for the Kick-off Call

All Coveo for Sitecore projects start with a complimentary 60-minute kick-off meeting with the Coveo team to walk through all available services at Coveo to support the project, and to review any design or architectural patterns.

Here is a list of requirements for a successful meeting:

- **Graphical representation of planned search features.**
  Visual support will ease the discussion.
  e.g., wireframe, mockups, screenshots.
- **Use cases.**
  Functional requirements and implementation plan.
  Coveo will suggest you which features could be used out-of-the-box and which ones will need to be customized.
- **Deployment architecture with details about the servers and hosting.**
  Based on the details of your environment, Coveo will suggest the best deployment plan.
- **Project information.**
  Milestones, planned deployment date, and any metrics related to the actual websites are required for the discussion.
  e.g., pages views, top queries

A Coveo specialist will provide recommendations to optimize your deployment.

Following the recommendations during the initial phase has a lot more impact than adjusting in the subsequent phase.

Coveo for Sitecore Project Timeframe

Every project is different, and the size of the project can change the time frame drastically.

Sizing of the Project

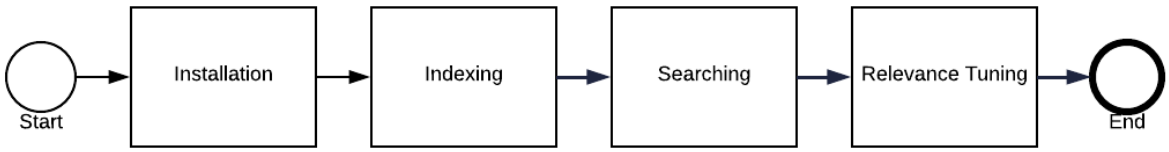
To help size the project, here are some of the key parameters:

- The number of sources to index
- The number of items to index
- The richness of the data
- The number of search interfaces
- The complexity of the search interfaces
- The complexity of the deployment process

A simple project will take less than a week of work, while a fully integrated Coveo experience with listing pages and extensive search usage can take a few months of development and configuration efforts.

Project Steps Diagram

Every part of a Coveo for Sitecore project is described by a section of this guide, from the installation process to the deployment process.



Testing and scaling are not represented in this diagram since it is a recurring activity throughout the project. Each step should be tested thoroughly.

The last section of this guide will also elaborate on the most common pitfalls while implementing Coveo for Sitecore. Any project considering one of the features mentioned in the pitfalls section should be reevaluated. Do not hesitate to ask for advice on the Coveo Q&A portal.

Resources

Customer Community and Support

The Coveo Community provides access to the entire Coveo product documentation through a Coveo-powered search interface. You can also submit product enhancement suggestions, log a support case, or visit the Coveo Q&A portal for technical discussions.

Customer Success

The Customer Success team offers a post-implementation optimization service to ensure the implementation achieves the set goals. The expertise of a dedicated Customer Success Manager is available to you for a small fee.

Customer Manager

The Customer Manager, your main point of contact, is responsible for everything related to your account and will redirect your requests to the appropriate resources as needed. For technical support, please log a case on the Coveo Community.

Training

A wide array of tailored training courses is available online on the Coveo Academy.

Solution Architect

The Solution Architect will assist with architectural choices as well as in applying best practices using the product. Every project receives up to three hours with a designated Solution Architect. The expert is assigned during the project kick-off meeting.

Sitecore Community

A vibrant online community represented on the Sitecore Slack, Sitecore Stack Exchange, and Sitecore Community forum. Each of these communities has Coveo channels, tags, or sections monitored by Coveo employees.

Installation

What is in the Box, and How to Install it.

This section presents a high-level view of the solution, highlighting the components used in the product. The step-by-step installation procedure is documented in the developers documentation.

PRO TIP: Before installing Coveo for Sitecore, create a dedicated Sitecore admin user for Coveo and use it in the post-install steps. Assigning a dedicated user to Coveo for Sitecore will prevent any lock-down issues.

Coveo for Sitecore Package

The Coveo package is required for every Sitecore instance. It contains the following:

- Coveo for Sitecore Search Provider
- Coveo configuration files, assemblies, and items
- The Sitecore Administration Interface
  - Coveo Command Center
  - Diagnostic Page
  - License Information Panel
- Coveo for Sitecore User Interface Tools
  - Coveo JavaScript Search Framework encapsulated in Sitecore renderings

Download the package that corresponds to your Sitecore version.

PRO TIP: Never directly modify a Coveo default configuration file.

The Coveo configuration files will be added in the include folder of the Sitecore instance. Like for all other Sitecore configuration files, do not edit them directly. Instead, use a patch file to update, add, or delete content.

The main files will have a version with a ".custom." designation in their extension. It is a patch file that you can use without the fear of it being overwritten by a future installation.

Coveo Cloud Organization

Your Coveo Cloud organization contains the following:

- Sources
  - Sitecore connector
  - Any other connectors provided with your license
- Access Management
- Query Pipelines
  - Coveo Machine Learning
  - Other relevance tuning tools
- Coveo Usage Analytics

Coveo Index Structure and Connection Type

Be aware that Coveo is a unified index composed of sources. A Sitecore index will become a source when it reaches a Coveo index. A Coveo index can have multiple sources from a range of systems. While most connectors use a pull mechanism, the Sitecore connector is built on the Coveo Push API. In this scenario, the Coveo Cloud platform is waiting for Sitecore to select the items to be indexed and send them in a valid format.

Read the Triggering the Indexing Process section for more information.

Indexing

Relevance Maturity Model Step 1
How to Retrieve, Enhance, and Index Content from Sitecore to Coveo Cloud

The Business Consideration of Indexing

In several Coveo for Sitecore implementations, it was observed that irrelevant items, spreading across multiple sources, are present in production indexes. Although it can be filtered at query time, these items affect the relevance and performance of the solution.

This section is about the importance of a well-managed index where every item is relevant to the target audience. It contains tips and tricks on how to clean your data before indexation. Apply those tips at the beginning of implementation to ensure that the overall search development experience is simpler.

The Coveo for Sitecore license model sets boundaries for the number of indexable items for each organization. Therefore, properly managing what content is indexed will ensure that your solution stays within these boundaries.

Determining What to Index ✓

Determining what to index should be executed during the early stages of a project, as choices made in this phase will impact the search process — from indexing time to query time. Every ecommerce solution of web applications has a skeleton: artifacts that help users navigate, search, or log in.

These artifacts are items that should not be indexed, as the developers will need to filter them at query time with specific rules. In the sections below, different filtering methods will be explained to exclude this content directly at the beginning of the project.

Every solution also have media items. Without context, these items are not relevant. Taken out of context, pictures of products or employees are not useful for the user. It is preferable to index the HTML pages to make them available. However, some media — such as PDFs — have rich text content and should be indexed.

Examples of Content to be Indexed:

- Content pages
- People
- Products and services
- News
- Places/offices/locations
- Papers/publications/articles
- Job offers
- Technical documentation

Examples of Content that Should Not be Indexed:

- Login pages
- Search pages
- 404 pages
- Page with lists referring to other elements, such as repositories
- Images

Before pushing content in the index, build a list of what needs to be indexed. Once the list is complete, refer to the Indexing Techniques section of this guide for tools to manage your content.

Languages

Coveo is a multilingual index which detects user language and adjusts relevance accordingly. The full list of supported languages is available online.

At the interface level, the Coveo JavaScript Search Framework localizes the strings based on the current user language. To pass that information to the search UI, Coveo for Sitecore uses the item language. The language is also used for language filtering, which is configured on the Search Interface rendering. This means that Coveo will create a new item for each language version of your items. Hence, for a multi-language site, the index size will be multiplied by the number of language versions on the site.

Triggering the Indexing Process

Always Live and Updated ✓

Once the initial rebuild is completed, Coveo for Sitecore triggers the indexing process when a content author saves a modification in Sitecore. This will update the Coveo_master_index[1] until the content author publishes the changes, allowing the Coveo_web_index to be updated.

This default strategy is used to keep your index up to date. It eliminates the need to perform full rebuilds when new content is added. These strategies are standard Sitecore indexing strategies and are defined on the index nodes directly in the Coveo for Sitecore configuration files.

PRO TIP: Indexed documents need to be processed before being searchable. This process can take up to 5 minutes.

Rebuilding or Refreshing Indexes

While other search providers in Sitecore needs to be configured to avoid downtime during the rebuilding process, Coveo will preserve its index until the rebuild is completed. Allowing you to rebuild your content without affecting your users.

Rebuilding your indexes is the only way to execute a full re-synchronization, which will also delete outdated items. A full rebuild will preserve the data in the index and delete it once the new items are in place. However, it is a more demanding operation for Sitecore and Coveo. Hence, it is recommended to avoid rebuilding often, and use refreshes when

[1] The index can be named differently.

possible. Both rebuilds and refreshes can be performed through a UI or the Coveo for Sitecore API.

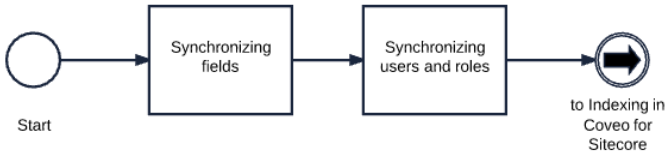
PRO TIP: Updating the Sitecore databases directly will not trigger an indexing operation. Make sure to manually trigger an indexing operation after the changes are done.

Diagrams

This section presents some diagrams showing the indexing process in Sitecore and Coveo Cloud. There are a few steps that can be used to intercept the items during their indexing process, allowing you to modify the item information based on your requirements. Those steps have been identified with a red border and will be explained in the next section.

Pre-Indexing

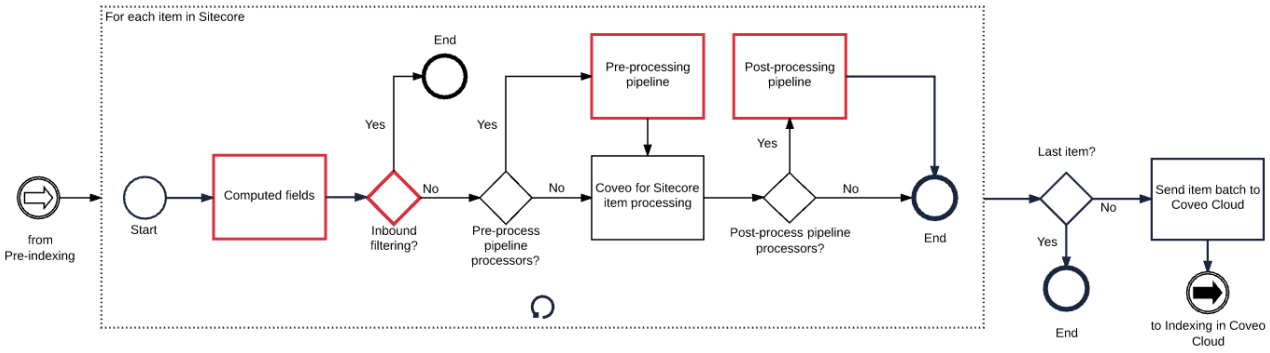
The pre-indexing step will synchronize fields from Sitecore to Coveo Cloud.



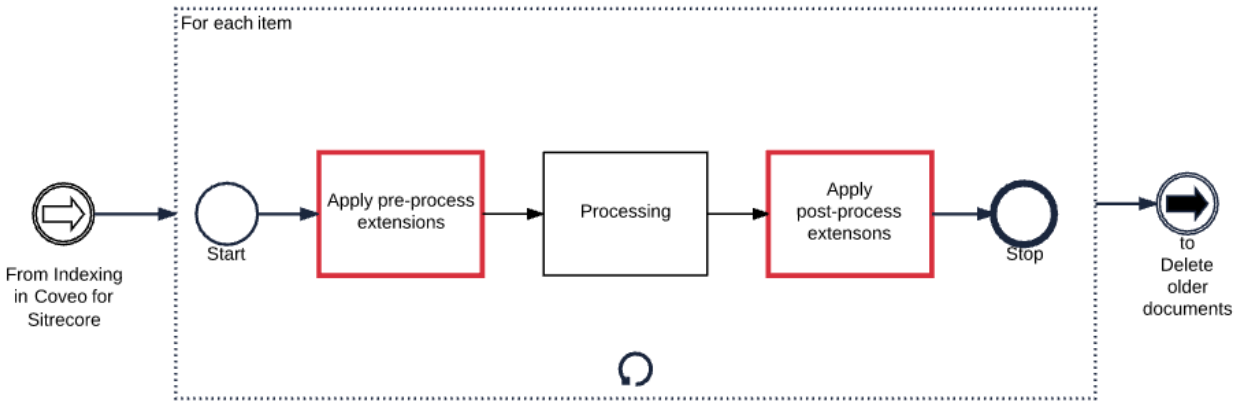
Modified or added Sitecore template fields, as well as computed fields and external fields are read and synced with your Coveo Cloud organization. There is a section in the developer documentation dedicated to this process.

Indexing in Coveo for Sitecore

The indexing process in Coveo for Sitecore crawls every Sitecore item and processes them through the Indexing Pipelines. The red processes are designed to receive custom code if required.



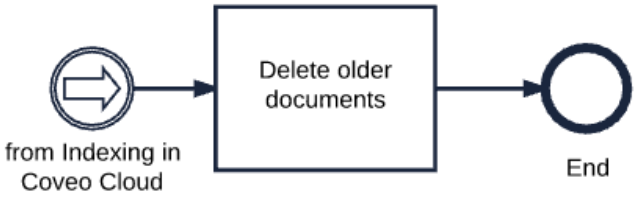
Indexing in Coveo Cloud



The indexing in the Coveo Cloud stage processes every item through the Coveo Document Processing Manager (DPM). The steps identified by a red border are designed to receive custom code if required.

Deleting Older Documents

The last step of the process deletes any item with a date older than the one recently pushed with a full rebuild. It ensures the Coveo for Sitecore indexes are always up and running.



Indexing Techniques

These indexing techniques help to manage the content to index efficiently . As mentioned in **The Business Consideration of Indexing** section, it is better to manage items at indexing time instead of at query time.

PRO TIP: Keep in mind that the Coveo Search Provider inherits from the Sitecore Search Provider, which means that indexing techniques used with Lucene or Solr can usually be applied to Coveo.

Scoping Crawlers ✓

Not all content in Sitecore is valuable. By default, the Index Crawling Root is set to `sitecore/content`, but it might be relevant to scope it further in the content tree to only get a specific area of the content. Multiple folders could be crawled in the Content Tree. Multiple crawlers could easily be added in the index definition. Changing an index crawling root can be done in the Coveo configuration file.

Selecting Which Templates to Index

Coveo for Sitecore will index all items from all templates that are within the scope of the crawlers. Even if you properly scope the crawlers on your Sitecore index, you might still ingest content from templates which make no sense to your users. It is then recommended to exclude templates from indexing.

PRO TIP: If you use an include list, make sure to inform the content authors or any other role who has the right to create new templates, as these new templates will not be indexed until you change the configuration.

Using a Processor in the Indexing Pipelines

Changing a crawler scope is the most efficient way, but also the most limited. The indexing pipelines give you a lot more flexibility to exclude with the preferred logic. This topic is extensively covered in the documentation. Below are some common use cases:

Filtering Items without Layout ✓

The `Coveo.SearchProvider.InboundFilters.HasLayoutInboundFilter` is an out-of-the-box inbound filter provided with Coveo for Sitecore. It only sends items having a layout to the

index. Partial items, built to be a part of a page, will not be indexed. This processor uses the `coveoInboundFilterPipeline`.

Filtering on a Custom Condition Using an Inbound Filter

Coveo Inbound Filters let you chose the condition for exclusion. The logic is written in C#.

Examples

- Add to each Sitecore item a checkbox called `DoNotIndex` and filter out items having the checkbox checked.

PRO TIP: Always use the Coveo Inbound Filter Pipeline instead of the Sitecore Inbound Filter Pipeline to avoid affecting all other indexes outside of Coveo.

Creating New Items Dynamically

With the increasing popularity of wildcard items in Sitecore, creating new items dynamically can become a necessity in most projects. The `coveoPostItemProcessingPipeline` can transform a single Sitecore indexable item into a list of Coveo items. The logic to create this list of items is written in C#.

Enhancing the Data

Coveo requires data in a specific format to be searchable. Sometimes, the data is not in Sitecore, or the information is an aggregate of other fields. There are several strategies to enhance the data which can be used in different parts of the indexing process.

Understanding How Metadata is Captured and Stored by Coveo

At indexing time, metadata is extracted from indexed items and stored as fields in the Coveo index. The assignment of the metadata in fields is defined by the mapping on the sources. In Coveo for Sitecore, the mapping is done automatically through the API. It avoids creating any type of mapping for Sitecore sources.

Fields have different types, such as integer, string, decimal, or date, which are defined at creation time. A field can also have options, such as facet, sort, free text search, etc.

These types and options for each field can be inspected in the field menu of the Coveo Cloud administration console.

PRO TIP: Do not modify the fields attributes used by a Sitecore source in Coveo Cloud, as Sitecore will overwrite them on the next indexing operation. Use the Coveo for Sitecore configuration file instead.

Determining Which Metadata Should be Indexed

After determining which items are relevant, the next logical step is to choose the information to index on those items. This step is crucial as it will directly impact relevance and user experience. Without the right fields, it is not possible to create an advanced search interface with facets and sorting options.

This process is relatively simple. First, take the content to be indexed, defined in the list from the previous section. Second, for each item, determine which fields should be used for facets or sorting.

The Coveo Command Center in Sitecore allows you to select fields to index.

If the project is considering multiple sources, the data from the external sources need to be normalized. This step will ensure that the sorting and filtering options are working on both the Sitecore and the external data.

Computed Fields

Computed Fields allow you to add fields dynamically at indexing time. Coveo for Sitecore offers several computed fields out of the box, but you can also add your own custom code. Computed fields can refer to other items in the content tree, such as getting the title of another item (a tree list for instance). Since it runs in a Sitecore pipeline, a computed field has direct access to the Sitecore API and databases. A custom computed field is a C# solution to be included in the Sitecore website.

Examples

- Add all the variants (e.g., colors) of a product to make a facet.
- Retrieve data from a related object.

Pre- and Post-Processing Pipelines

The pipelines used in the previous section to create new items can also be used to enhanced the same items with richer metadata. Most of the time, a computed field will be the right choice, but sometimes the pipelines offer more flexibility.

Examples

- Duplicate the item to make a public copy with less sensitive information.
- Change the item click URI or any other value computed by our crawlers.

Coveo Out-of-the-Box Computed Fields and Processors ✓

A few processors and computed fields are built directly in the product to help with the indexing steps. As the code is already packaged with the Coveo for Sitecore assemblies, it requires minimal configuration.

Out-of-the-Box Computed Fields and Processors

- Image URL resolver
- Fetch HTML processor
- Item title resolve

Creating an HTML Representation of Your Pages ✓

You can create an HTML representation of your page content, making the full HTML content available for free text search. This increases relevance, but can also deteriorate it if no precautions are taken. When using a processor performing an HTTP request, the repetitive parts of the pages should be ignored, and the crawler should only index the parts of the page that are worth being searched.

PRO TIP: A processor sending an HTTP request will slow down indexing and demand more resources from the Sitecore instance. It is an easy solution when the page content is a multitude of small unrelated components, but should be avoided if possible.

Extensions in Coveo Cloud

Coveo Cloud offers a Python runtime environment to execute code snippets during the indexing process. The documents can be intercepted before or after going through the native Coveo processing. Extensions in Coveo Cloud are easier to set up than the ones in Sitecore environment since they do not require compiling or deploying; however, they provide limited functionalities.

Examples for Pre-Processing Extensions

- Reject a web crawled page with advanced rules — sometimes, the basic rules in the web crawlers configuration are not enough.
- Format a value.

Examples for Post-Processing Extensions

- Modify the body of a page.
- Change field values that are computed by the index.

Adding Custom Code in Sitecore vs. Coveo Cloud

As you may have noticed, there are extension points for custom code both in Sitecore and Coveo Cloud. Deciding which one to use depends on several factors, such as access to Sitecore and the available processing power.

Using a Processor in the Sitecore Pipelines

- Can use the Sitecore context
- Are more demanding for Sitecore during the indexing process

Using a Coveo Indexing Pipeline Extension

- Can be used for other sources than Sitecore
- Can be changed without risking a Sitecore downtime
- Are processed by Coveo Cloud, hence reducing the load in Sitecore

Exploring Indexed Content

The tools and pipelines that interact with the query before it reaches the search interface in Sitecore are covered in the Searching section of this guide. The best way to view the indexed content without any processing is through the Content Browser.

It is the best tool to validate that the indexing management is done properly. It is the first troubleshooting step suggested by the Coveo Support team in a missing content case.

PRO TIP: The Content Browser hides secured content. Make sure to use the View All Content button.

Security and Permissions

Security of each data source, including Sitecore, is supported natively by Coveo and uses early binding based on a security cache, removing the burden of managing securities using filters. However, ensure the right roles and users have access to the right content in Sitecore. Coveo will follow the security model of your Sitecore instance, unless the administrator completely deactivates the indexing of permissions on documents.

Building Search

Relevance Maturity Model Step 2
Front-end Components, Styling and Querying process

Leveraging the Coveo UI Framework ✓

The Coveo JavaScript Search Framework

Using the Coveo JavaScript Search Framework in a project can help save a lot of time. It contains a large selection of responsive search controls and query management tools. Every control is tied to JavaScript code, sending analytics events to both Sitecore and Coveo analytics engines. This data is used for reporting, and ultimately to feed the Coveo Machine Learning models.

The Coveo For Sitecore Hive Framework

In Coveo for Sitecore, Coveo JavaScript Search Framework components are fully integrated into Sitecore rendering items, called the Coveo for Sitecore Hive Framework. Mastering that framework is the key to build feature-rich and efficient search interfaces containing powerful ranking for the content authors.

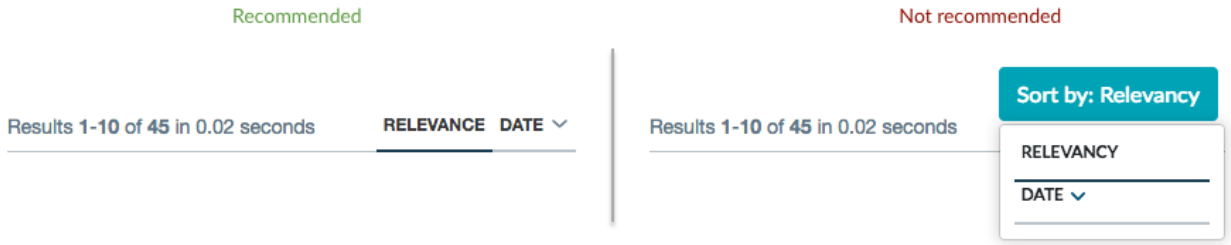
Embrace The Design

The default look will certainly not fit your expectations, but it is based on industry best practices and data shared by Coveo customers. The interface is customizable, but refer to the community when modifying the standard behavior of a Coveo component. A good example is the sort container. The default look has the sort options next to each other. A drop-down menu may be requested; however, this increases the number of clicks, hides options, and makes toggling a complex manipulation.

Example of Simple vs. Complex Controls

The Coveo search interface and the related controls were designed to provide the best-in-class experience to site visitors. It is recommended to consult with the UX specialist for your site early in the project. Remember that the Coveo JavaScript Search Framework components are built for efficiency. It optimizes the search experience without imposing a specific style.

Example



Using the Components

Follow the steps in the developer documentation to build your first search interface. With the Coveo for Sitecore Hive Framework, the components are customizable without changing the source code. If a component requires changes, remember to create a copy of the rendering and of the .cshtml file. However, before you do so, make sure that you are fully aware of all the strategies to add features or change the default behavior of an existing feature of a Hive component

Building New Components

Most of the time, a JavaScript pseudo-component will be enough.

Since the Coveo JavaScript Search Framework is fully open source, it can be forked on GitHub and rebuilt. Using TypeScript, new components can be added, tailored to specific use cases, and still be considered fully out-of-the-box components.

Advanced programming knowledge is required. Be aware that pull requests will be refused if not properly tested. Read the Creating a Custom Component article for more information.

The Search Box

The Coveo for Sitecore Hive Framework comes with two search box renderings: a standard search box and a global one. The standard search box is initialized by the search interface, while the global search box is initialized by itself and redirects the user to a specific search interface URL, passing the query as a URL parameter.

Why Two Search Boxes?

The Coveo Search Interface component works in an asynchronous fashion, where every facet click, page changed, and query entered changes the results, without fully reloading the page. This provides a smooth experience to users and improves the overall performance.

The global search box is a full redirect, which submits a URL change every time a new query is entered. This is fine when the user is on a different page than the search page, but on the search page itself, it would force a page reload on every query.

Using the Same Box In and Out of the Search Page

To preserve the style of the site, many web design specialists prefer to use the same search box across the site. It requires the use of the global search box on the search page as well.

As previously mentioned, the global search box is simply performs a redirection, meaning that it would work on the search page, but would completely reload the page, which is a bad user experience and a non-efficient behavior.

However, there is a way for the search interface to detect the global search box and initialize it like a local one. See Inserting and Configuring a Global Search Box Using the Coveo for Sitecore Hive Framework article for a step by step approach.

Suggesting Queries ✓

There are 3 main advantages to provide queries to a user:

1. It is faster than typing the full keyword
2. It prevents misspelling

3. It can help formulate questions or queries based on other users' input

Coveo machine learning-powered Query Suggestions are enabled by default on the search box component. It will start providing suggestions as soon as it has enough data. This guide explains how to set up an ML query suggestion model.

Styling

By scoping on the classes available in the Coveo JavaScript Search Framework components, it is possible to hook on Cascading Style Sheets (CSS) selectors to overhaul the styling accordingly to the project. With this technique, components can be styled in every way imaginable.

Best Practices

In Coveo for Sitecore Hive, a "Custom CSS classes" field is available. You can insert a new class there, such as `.coveo-custom`, and use it in addition to the Coveo classes to select the component to style.

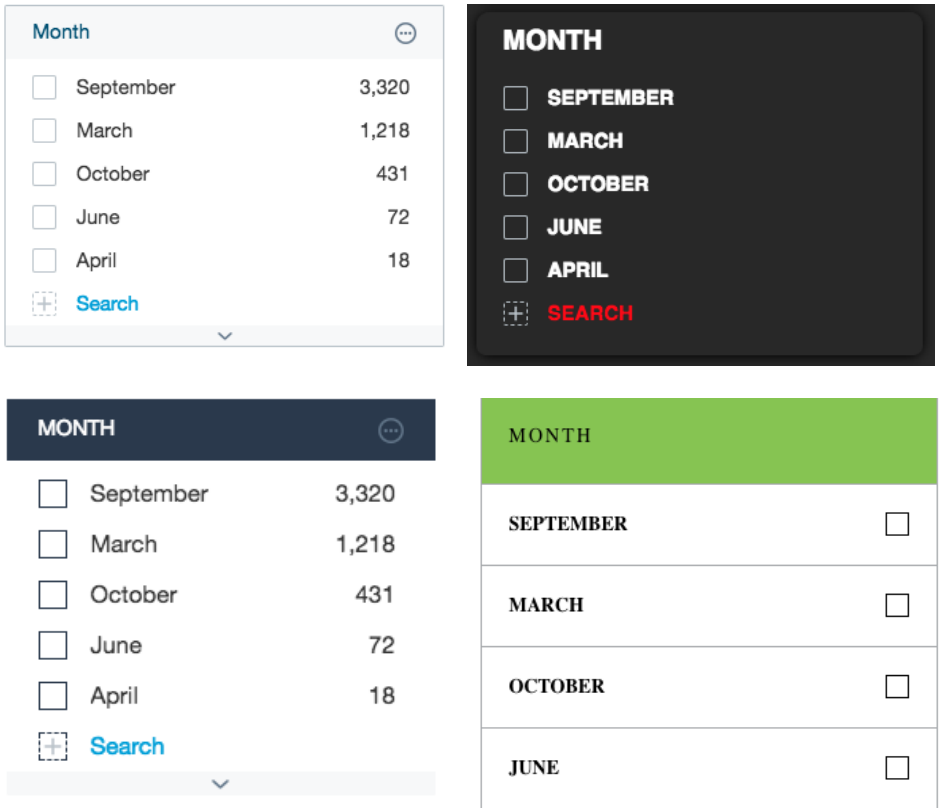
Simply add in a CSS file the styling modification required by the project.

PRO TIP: There are two types of HTML classes used by Coveo: nodes starting with "`Coveo`", and nodes starting with "`coveo-`". Nodes using classes starting with "`Coveo`" are Coveo JavaScript Search Framework components. Components are attached to several JavaScript functions; altering them can affect the functionality of the solution. Nodes using the "`coveo-`" classes are used only for CSS styling purposes.

Code Example

```
.coveo-custom .CoveoFacetRange,
.coveo-custom .CoveoFacet,
.coveo-custom .coveo-facet-header,
.coveo-custom .coveo-facet-footer {
    border-radius: 0px;
    border: none;
}
```

Facet Styling Overriding Examples



PRO TIP: Look at the CSS files of Coveo for Sitecore demos to see how CSS overriding is handled. It can be observed in the Coveo for Sitecore Habitat demo.

Templating

Leverage the Information in the Result Templates ✓

Using Underscore templates, you can adapt the look of each result based on the condition of your choice. One of the most critical points behind templating is to make sure that the user understands why the search engine returns items. If you are querying against a title or a description, make sure to include these fields in the template. Use highlighting to emphasize the words to be recognized in the items.

Example of a Result Template using Highlighting in the Title and the Description

DigiSound USB-**Powered** Portable Speakers

Audio | Computer Speakers

★★★★

$19.99

Listen to music and more on the go with these Helix DigiSound USB speakers that deliver 2.6 watts of total system **power** to ensure quality sound. The USB-**powered** design provides flexible connectivity options.

Show more

PRO TIP:

To simplify your templates, fetch all values to be used at the beginning of your code. Use the same technique to evaluate your conditions and store the results in boolean values. It will allow for a much cleaner code.

Remember that you can use file templates now with Coveo for Sitecore Hive Framework, which also helps to maintain a cleaner architecture.

Use the same mechanism for a custom quick view component!

UI Personalization

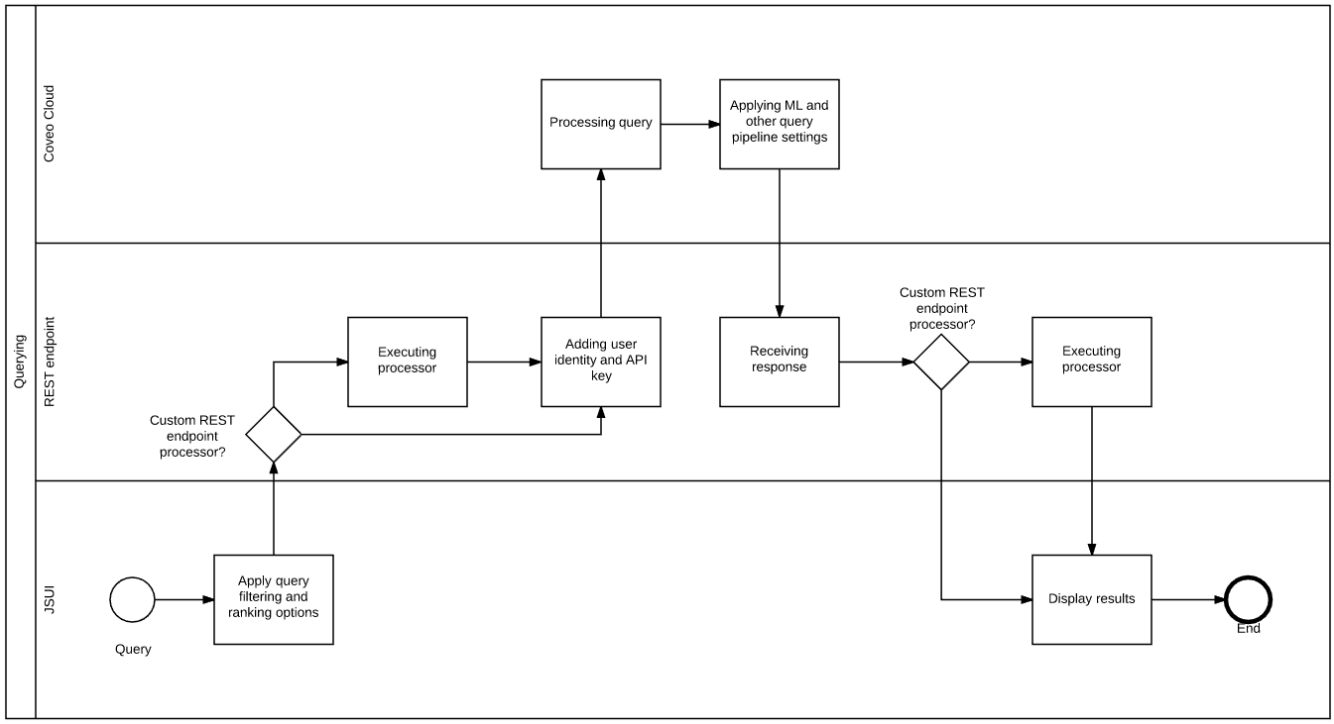
Since Coveo components are Sitecore rendering, conditional rendering can be used to modulate the search interface. Using the rule editor feature, content authors can change the way a search interface is rendered to provide various experiences to a variety of end users.

Example

- A language facet can be displayed with the current language preselected on non-English versions of a website. This helps users fall back on English results if the user is not satisfied with result sets in the current language.

Querying Process Diagram

Here is the complete querying process. Three components are involved in this sequence of activities: Coveo JavaScript Search Framework, Coveo for Sitecore backend, and Coveo Cloud.



Query building

What a Coveo Query Is Composed Of

A Coveo query is composed of several expressions. Administrators think of a query as only the standard expression, which is what the user typed. However, there are usually several advanced, constant, and other expressions that are as important as the user input. An advanced expression is hidden from the interface and is used to adjust the scope of the query. In a listing page without a search box input, the advanced expression will be the main driver of the query.

For example, an eCommerce solution can have product and product-related manuals. One search page could be for product only. A user visiting this page will use the search box to search for a fridge. At this point, the standard expression is "fridge", and the advanced expression is something like "`@contenttype=='products'`" and could also include other filters such as the language "`@language=='en'`." All of these expressions are using the Coveo Query Syntax. It is highly recommended to understand this syntax before building your set of expressions.

There are more expressions for each specific use case.

How to Build an Expression

The Coveo JavaScript Search Framework builds the standard expression entered in the Coveo Search Box component. It also builds the other types of expression based on other components such as facets, sorts, ranking and filtering panels, and others. In a standard Coveo for Sitecore scenario, a developer could complete a full project without ever having to build an expression and its related query, but if needed, the ExpressionBuilder is the solution.

PRO TIP: You can use the ExpressionBuilder to add a standard expression, but this is to be avoided. The standard expression is for user input and will be returned in the analytics and be used to feed machine learning.

Using Expressions to Filter at Query Time ✓

Filtering at query time is a recommended strategy to reduce the scope of certain search pages, search driven listing pages, or search tabs. The index must be clean of unwanted items before filtering at query time. Refer to the Indexing section of this guide for more information about filtering at indexing time.

In Coveo for Sitecore, filtering at query time can be done using the Coveo rules in the Sitecore Rule Set Editor with the Enterprise edition, or using the ExpressionBuilder with the Pro edition. To simplify the usage of the ExpressionBuilder with Sitecore, it is recommended to use it with a rendering, as explained in this guide.

Even if you do have the Enterprise edition, using the ExpressionBuilder can be useful to hide permanent filtering rules from the content authors.

Building an expression will also be necessary when manually changing the result ranking. We will cover more of the result ranking strategies in the Relevance section of this guide.

Coveo for Sitecore REST Endpoint Proxy and Pipelines

Before sending the queries to Coveo Cloud, Coveo for Sitecore REST Endpoint Proxy will intercept them and use a set of pipelines to add the API key as well as the user role and identity. Once the request comes back, it goes through the REST Endpoint proxy again for additional processing through a set of pipelines.

This is the last chance to intercept the items and process them before they are sent to the user interface. Since the pipelines are in Sitecore, it gives access to the full as well as the databases.

Example

- Remove fields based on the user role.
- Call an external API to fetch some additional data related to the results set, such as ratings for restaurant.

Relevance

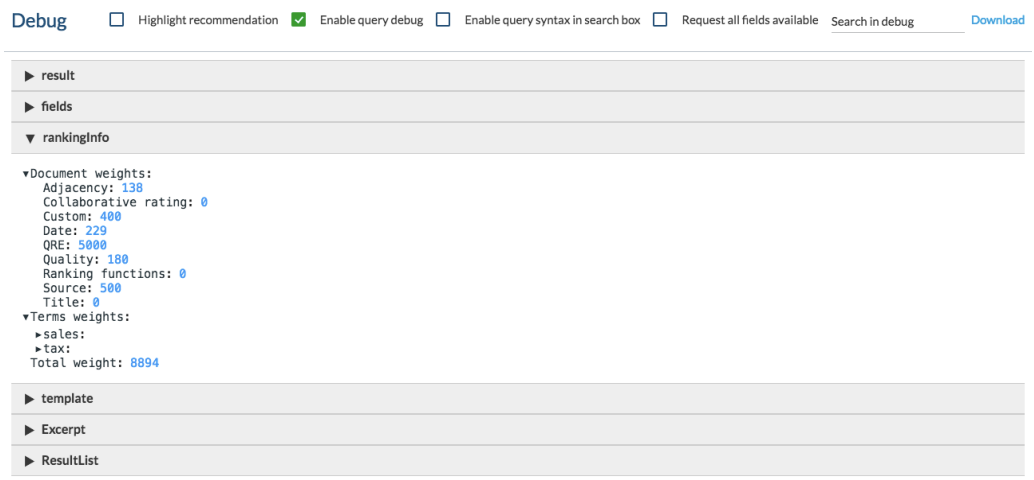
About Result Ranking

Result ranking is the foundation of search relevance and one of the key concepts behind a successful search engine. Undermining this concept will lead to an underperforming search experience.

Scoring

Coveo uses a scoring system to determine result ranking. When a keyword is entered, the scoring system applies a set of rules to determine if the result is relevant to the user. For a listing page without free text search, several elements will come into play such as the last modified date and the richness of the items. Scoring can also be affected with tools such as boosting rules, featured results, or even Coveo Machine Learning ART.

The score of a result can be seen in the Debug Panel.



PRO TIP: Use the sort by relevance filter. It is the only way to take full advantage of Coveo. It is also the only way to leverage boosting and machine learning Automatic Relevance Tuning. Various sorting options are always nice

add-ons to a search interface, but they should not be selected by default.

What is Searchable?

The scoring system is principally based on a match between the query term and the content of the item. This means that managing the searchable content on your items is critical for items to be scored properly. Two main concepts are searchable in Coveo items: the body field and other free text searchable fields.

Setting the Body of the Document

Coveo items have a default property called "body," which is created by extracting the content of the items. Some search tools will simply concatenate all the fields of the items to create the body, but not Coveo. The body is filled with extracted information from the document. For media items such as a PDF, the Coveo document processor will extract the contents of the PDF and add it as HTML content to the body field of the item.

The processor will run at indexing time, as explained in the Creating an HTML Representation of your Pages section of this guide.

Having the HTML content of an item allows you to use the quick view feature.

Configuring Fields to be Searchable

Outside of the main body of the item, you can select fields to be searchable.

Use the includeForFreeTextSearch in your configuration file on each required field. Use this approach when a field might be on your template, but not exposed on the rendered page.

Monitoring Relevance

Coveo Usage Analytics is the best way to monitor your Coveo implementation. When properly configured, it is a powerful platform that gives you access to meaningful dashboards, from business insights to implementation feedback.

Tracking Events

Analytics are sent by the Coveo JavaScript Search Framework to the Coveo Cloud platform. The Coveo JavaScript Search Framework is listening on many events registered by the components and reacts to front-end user behaviour.

You can track the events being sent using a network proxy, or simply in the browser console. Here is an example of a payload sent from the interface to the Analytics API.


```
{
  "actionCause":"documentOpen",
  "actionType":"document",
  "anonymous ":true,
  "device ": "Chrome ",
  "mobile ":false,
  "language ": "en ",
  "responseTime ":0,
  "originLevel1 ": "CoveoSearch ",
  "originLevel2 ": "_524BC2CF-1942-42EA-A236-6AE80E3B7596 ",
  "originLevel3 ": "/sitecore/content/Home/CoveoSearch ",
  "customData ":{ "documentURL ": "/en/Team/Mike-Casey ", "documentTitle ": "Mike Casey
", "JSUIVersion ": "1.1276.20;1.1276.20 "},
  "userAgent ": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/55.0.2883.87 Safari/537.36 ",
  "searchQueryUid ": "938f1d27-6b40-4fef-add2-e31035ee7e43 ",
  "queryPipeline ": "Launch Sitecore ",
  "documentUri ":
"sitecore://database/web/ItemId/A410F42C-903A-4D73-AF5C-F13313EA1BF2/Language/en/Version
/1 ",
  "documentUriHash ": "WñOHY1H6vY78EOñH ",
  "documentUrl ": "/en/Team/Mike-Casey ",
  "documentTitle ": "Mike Casey ",
  "collectionName ": "default ",
  "sourceName ": "Coveo_web_index",
  "documentPosition ":1,
  "viewMethod ": "documentOpen "
}
```


The payload above shows a click event on the “Mike Casey” document from the “CoveoSearch” interface.

The actionCause is categorized in one of the three event causes: search, click, and custom. In the example above, documentOpen is a click event. The rest of the metadata will be used to document the event.

The Coveo for Sitecore Hive Framework sends a significant number of events and metadata out of the box, but you can also expand this list by sending custom events and metadata.

Working with the Analytics Metrics and Dimensions

From an administrator perspective, the metadata can be extracted in the Coveo Cloud platform. The metadata will be called API Name and can be combined with an event cause to create a dimension.

In other words, a dimension is metadata (extracted through an API Name) in the context of one or more event causes. Refer to this table for more details.

Dimensions are then combined with metrics to produce dashboards. With these principles in mind, you can create dashboards on everything you send to the analytics API. There are dashboards that are already available to get acquainted with the type of insight you can gather.

Exploring the Default Summary Dashboard

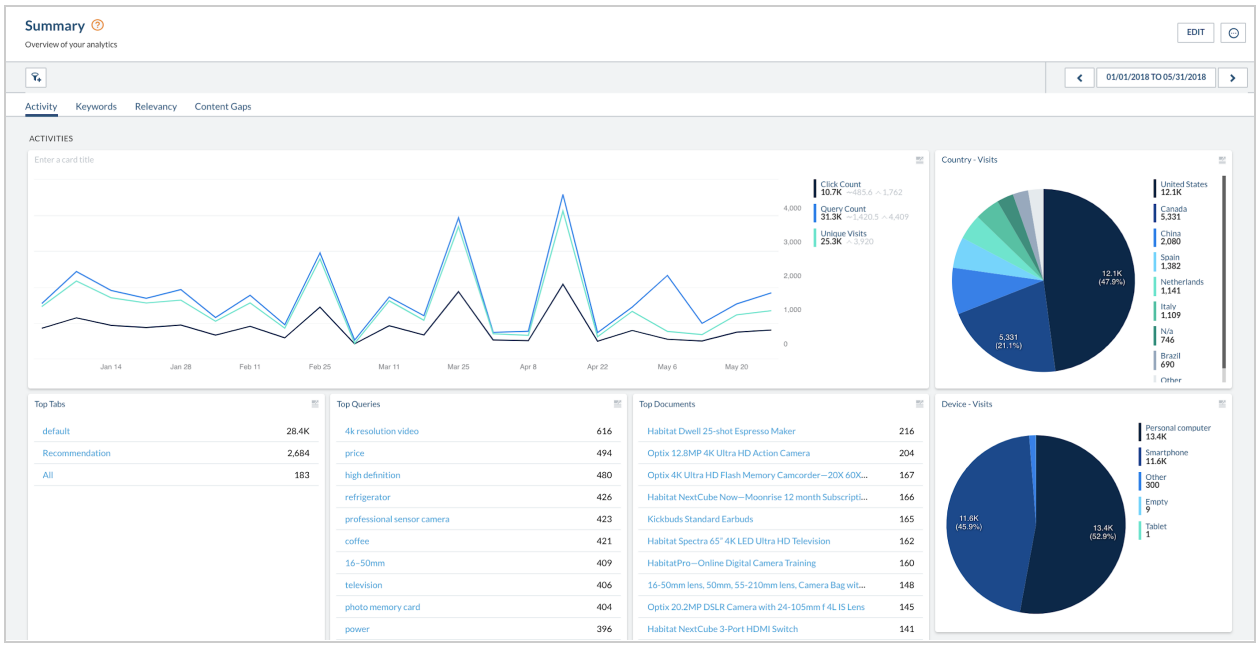
The Summary Dashboard is a predefined general overview of all the basic information captured in analytics. This is a good out-of-the-box starting point to perform analysis of your data. More information can be found on the following documentation.

Activity Tab

The Activity Graph is where you can see the number of queries, users and clicks captured over a given period of time. It gives a general idea of what happened and who used the Coveo interfaces.

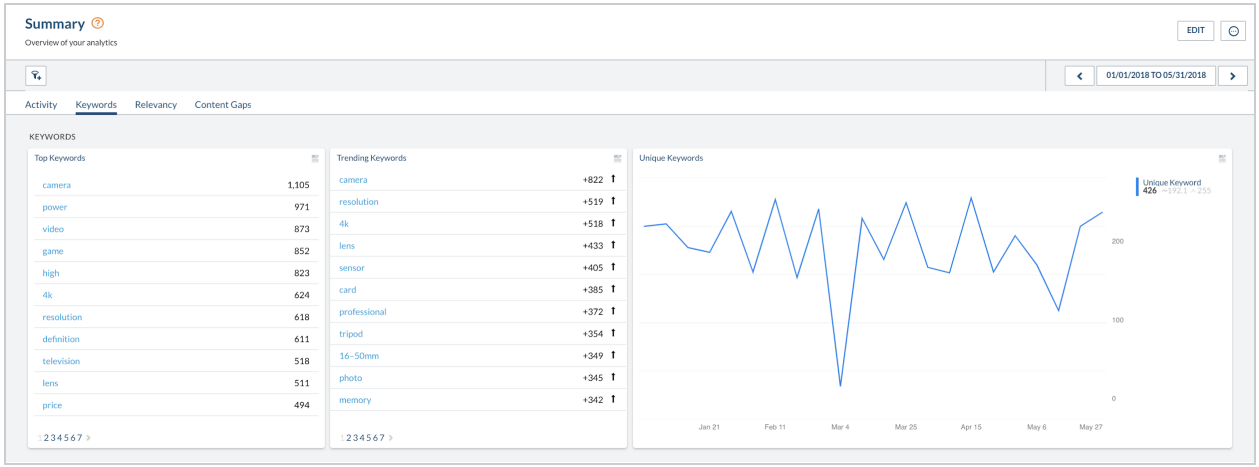
This chart provides the query count for the selected time horizon as well as the average and peak daily values. You can also see the click count for the same period and the Unique Visits. For each of the previously mentioned metrics, a trend indicator is also provided as a comparison of the selected time period with the period that preceded.

At the top of the screen, you can find the time horizon (date range) and the filter properties. Clicking the calendar icon allows you to choose the time period you want to analyze.



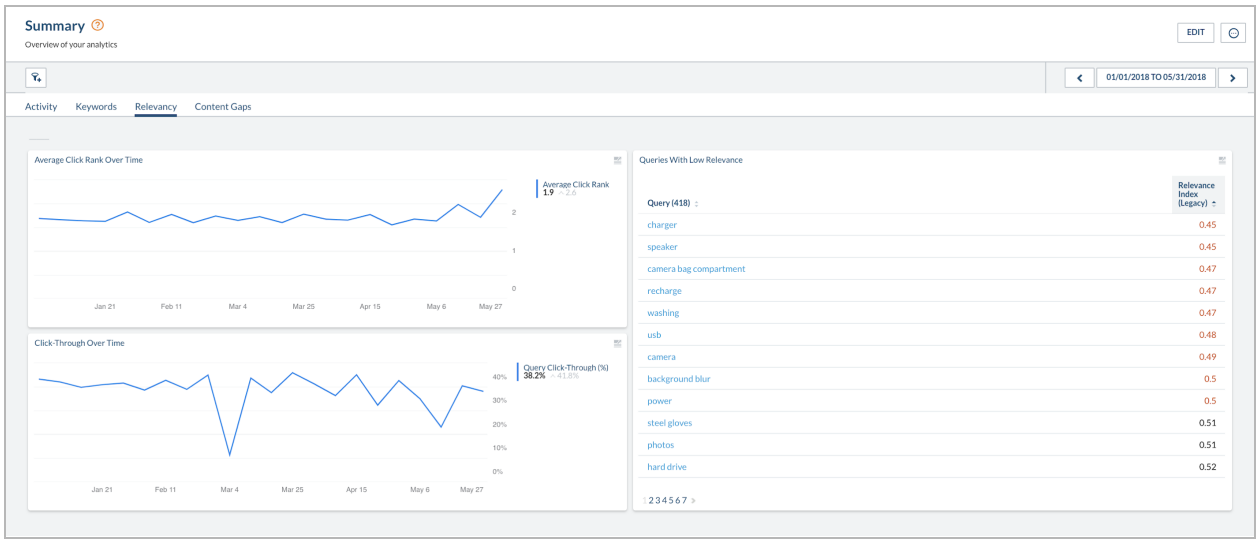
Keywords Tab

The keywords section helps you identify keywords that are often used in queries and display the top trending keywords. As with all reports in the dashboard, you can drill down on specific keywords, change the analysis time frame using the period selection, and add new parameters to each table or graph to meet your needs.



Relevancy Tab

The relevance section helps by analyzing the average click-rank and click-through percent over time, as well as show you keywords with the lowest relevance.



Average Click Rank Over Time & Click-through Over Time

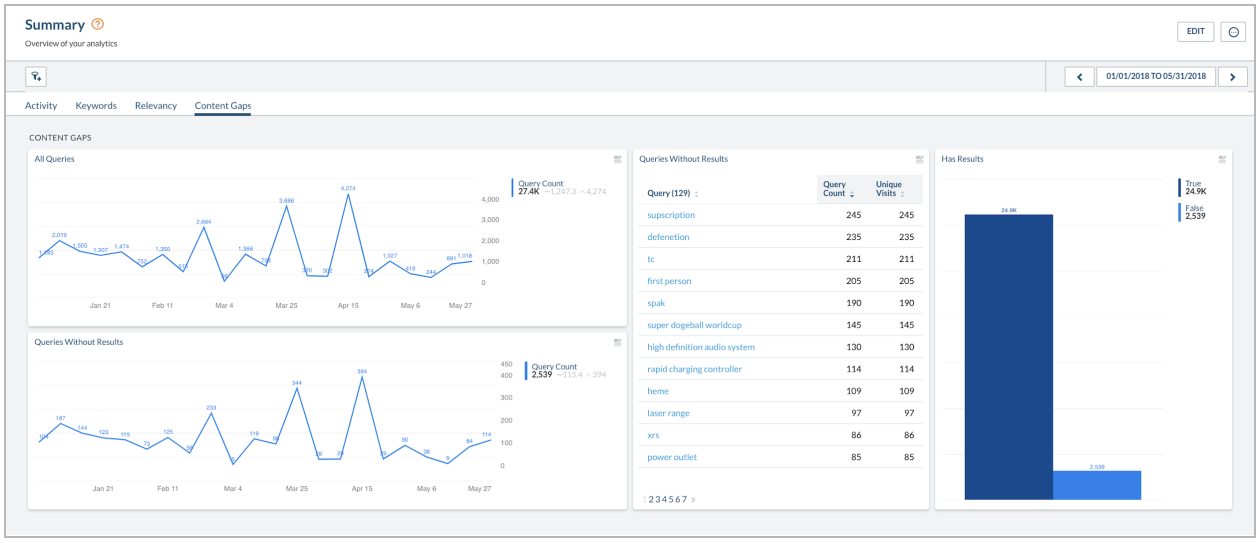
This metric shows the average position of the document that was clicked. The lower the value, the better. A value over 10 should be investigated. The click-through graph shows the ratio of queries resulting in a user clicking at least one result over time. The higher the value, the better. Typically, queries displaying optimal results have a click-through ratio above 0.66. Values under 0.50 should be investigated, particularly those with higher query counts (meaning that users are frequently searching using this query, but are not clicking anything).

Queries and Keywords with Low Relevance

The Relevance Index is a derived metric based on queries and keywords with relation to their commonality within the index, as well as other ranking factors to give an overall weighted score. A low score shows a query with poor relevance. Best practices recommend that anything at or below 0.50 should be checked.

Content Gaps Tab

The content gaps section analyzes the queries and keywords entered by users to identify the top occurrences with no results, no clicks, too many clicks, or poor ranking of clicked content. This analysis ultimately helps to identify which queries do not return relevant results and from there, pinpoint potential content gaps, or content that does not exist yet.



Knowing the topics people are most searching for also helps to prioritize content creation. The content gap panel also displays time series graphs that show *All queries submitted* vs. *Queries that returned no results*. The objective is to see queries with no results decrease over time.

Queries and Keywords without Results

Those are indicators to be used to identify content gaps. *Queries* comprise the entire search string that the user entered, whereas *Keywords* are a breakout of the terms to help identify fundamental concepts that may be lacking content. These metrics may also identify over-scoped queries or security conditions such that the content exists but the scope and filters used were too limiting, or a lack of permissions prevented the system from returning results.

Creating Custom Dashboards

In the Coveo Analytics tool, it is possible for you to add your own custom and personalized dashboards to further your use of the tool. You will be able to track the metrics that are important for your organization. To optimize the value of Coveo, see Creating Usage Analytics Dashboards.

Relevance Tuning

Relevance Maturity Model Steps 3 to 6
How to Improve Result Ranking Based on User Feedback

This section is the most important of this entire document. However, using the tools suggested in this section on an unclean index, a poorly designed search interface, or a query filter on the wrong content will not yield the expected results, hence the necessity to correctly understand and apply the practices of the previous sections before moving forward.

Ranking Management in Sitecore

Coveo for Sitecore offers two approaches to ranking management directly in Sitecore. Coveo Cloud also offers a feature called the Query Pipeline, a set of tools for a generic approach. All of these approaches use Query Ranking Expressions (QRE), which are created by the ExpressionBuilder introduced in the Query Building section of this document. QREs add or remove points to the score of the items, which can be monitored using the Debug Panel. Refer to the Relevance Tuning section of this guide for more information.

PRO TIP: You can add as many points as you want when building a QRE, but it is recommended to stay within the -100 to 100 range to avoid breaking basic relevance.

Coveo Rules in the Sitecore Rule Set Editor

Only available in the Enterprise edition, the Coveo rules in the Sitecore Rule Set Editor can be used to create boosting rules. Boosting rules will create an underlying QRE. It is the

only way to write QRE expressions without having to learn the Coveo Query Syntax or to manage field name conversion.

Examples

- Items having specific templates should be ranked higher.
- Items marked as deprecated should be ranked lower.

QRE Using the ExpressionBuilder in Coveo for Sitecore

QRE using the ExpressionBuilder allows for the same functionalities than the rules in the Sitecore Rule Set Editor, but without the UI. The rules are coded in JavaScript and inserted in a CoveoForSitecoreRankingExpression component.

This component adds the expression to the user query.

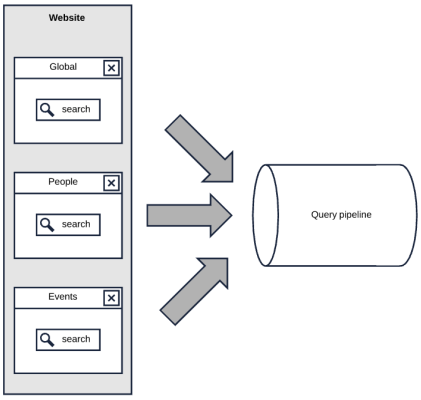
This can also be useful in an Enterprise edition to hide the rules from the content authors.

Using the Coveo Cloud Query Pipelines

The query pipeline is where the queries go between Sitecore and the Coveo index. It is also where machine learning powered suggestions, recommendations, and relevance are configured, as well as several other relevance tools. When an implementation is complex, you need to understand the value of having multiple pipelines running in parallel.

Regular Setup

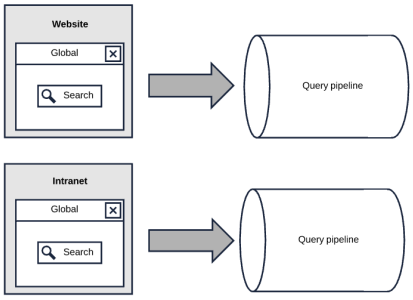
A Coveo for Sitecore cloud organization is pre-configured with a pipeline called "default." In every query-casting Coveo component, the value is also set as "default" in their Query Pipeline settings. The main advantage here is the simplicity of this architecture. Since all the ML models are independent from a search interface to another, this works perfectly.



PRO TIP: Please note that the settings used in the Ranking Weights, as well as the Thesaurus, is shared across all search interfaces.

Advanced Setup

If you use the same Coveo Cloud organization in a multi-site configuration or for intranet search as well as website search, it is strongly recommended that you set up multiple pipelines. This helps manage custom settings for your different environments.

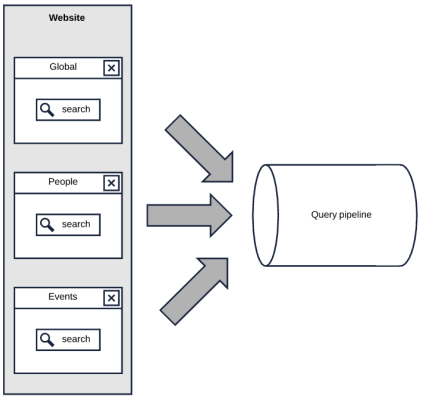


Examples

- Different thesauri
- Different ranking weights
- Different ranking expressions
- Query suggestions should be different for countries having the same language

Hybrid Setup

This setup is using conditions to route queries in different ML models inside the same query pipeline. It can be useful in a multi-site environment where a common thesaurus is shared, but custom suggestions must be provided to every search interface.



A/B Testing

A/B testing is a common feature in digital marketing. With Coveo, you can set an A/B test between two pipelines to compare the results. You can monitor the outcomes of the two pipelines using the A/B testing dimension in Coveo Analytics.

Machine Learning Automatic Relevance Tuning (ART) ✓

Automatic Relevance Tuning (ART) is one of Coveo Machine Learning capabilities and one of the most powerful features of the Coveo platform. If you have configured your search solution properly and made sure that user interactions are tracked in the Coveo Usage Analytics, ART becomes a relevance optimizer.

ART changes the ranking of the results based on the popularity of the items. The beauty of ART resides in contextualization. By passing the information along the queries, the tuning will be different for each user. This mechanism is unique for each search interface, and each language. Review the ML process section in this document to learn how to customize ML.

Examples

- If users are looking for the keyword "Primo," and then click on the third result, the third result will be ranked higher for the next users.
- If users are looking for a keyword that yields no results, and in the same journey, they finally find something, ART will make a relation between the initial keyword and the final result, thus helping future users.

Using Conditions

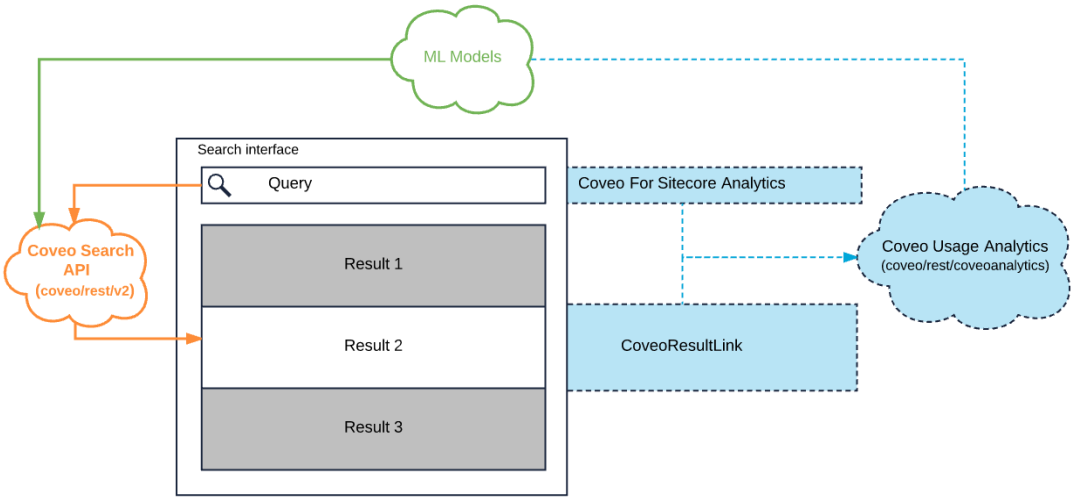
You might want to use the tools mentioned above only when certain conditions are met.

Example

- If the user is considered an internal user, a small boost should be applied on internal articles, since an internal user can be looking for detailed content instead of informational pages. This can be achieved by combining Internal IP and Conditions on Ranking Expressions.

Understanding the ML Process

As mentioned previously in this document, ML is used to increase relevance and to provide query suggestions. From a technical perspective, these functionalities are relying on some front-end Coveo components.



For each query sent to the Coveo Search API, the Coveo for Sitecore Analytics component sends information to the Coveo Usage Analytics[T] rest endpoint. For each result clicked, the CoveoResultLink component, which is a class applied to all links in the Coveo Result Template, sends some data to the Coveo Usage Analytics rest endpoint. The ML models are fed by the Coveo Usage Analytics, and influence the results in future queries.

PRO TIP: The Coveo for Sitecore Analytics as well as the CoveoResultLink components are mandatory for Query Suggestions and ART. Make sure that those components are included in the project.

In Coveo for Sitecore Hive, Coveo for Sitecore Analytics is a rendering to be added to the page, while CoveoResultLink is one of the components included in the default result template. Just make sure to keep it there when customizing your templates.

ML Parameters

The Coveo Machine Learning models are following simple parameters to contextualize their actions. These parameters will create different models for each combination.

- Search Interface: query suggestions and ART are only applicable to a specific search interface. This value is set automatically and can be overridden by changing the values of the "`originLevel1`" attribute of the Usage Analytics components, and the "`searchHub`" attribute of the search interface.
- Language: ML models only suggest content in the user's actual language.

To have more flexibility, use the hybrid setup explained in the Coveo Cloud Query Pipelines section.

ML Personalization

Adding contextual information can increase ML personalization. The Coveo User Context component provides additional information to Coveo Usage Analytics.

Examples

- Query suggestions and ART are contextualized to users' country (users from Canada will have tailored suggestions and ART).
- If your project is using Sitecore xDB personas, a tagged visitor will have personalized suggestion and ART based on other visitors having the same pattern card.

PRO TIP: The Custom context attribute in the Coveo User Context will help you adding contextualized information to your ML models. Use this with Sitecore Personalization to have accurate contextualized ML models.

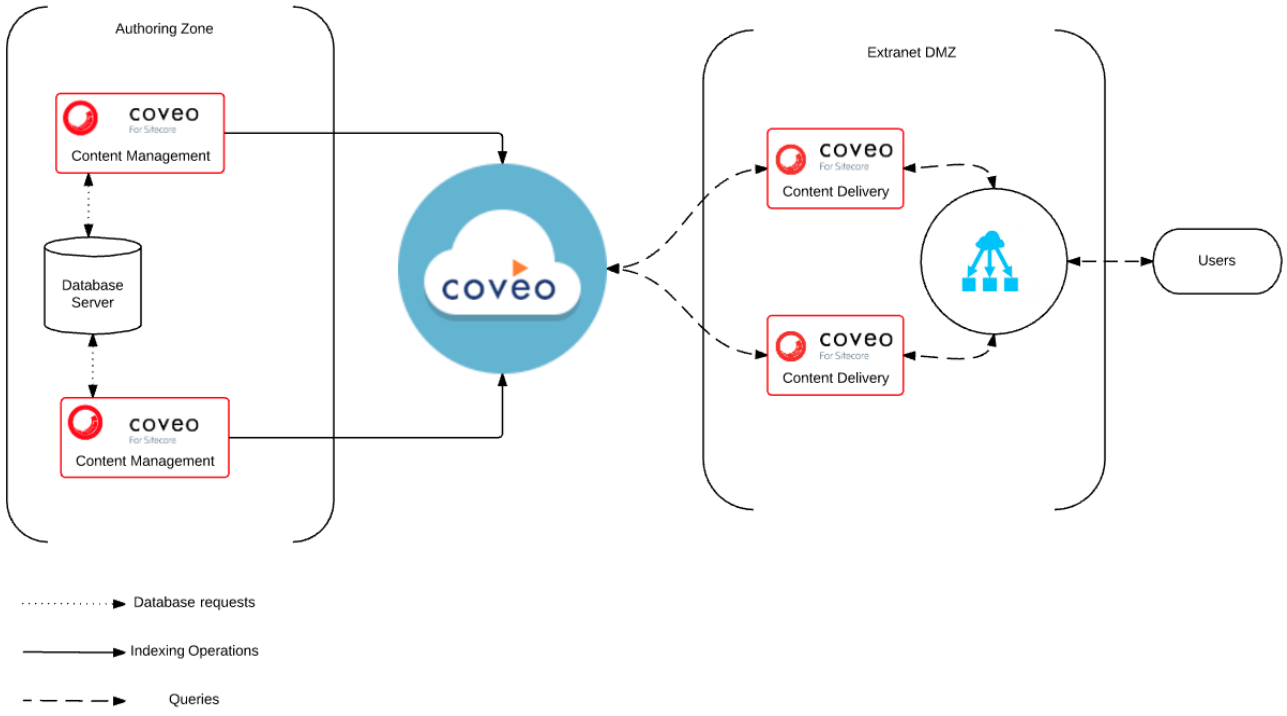
Deployment and Scaling

Deploying Changes with Coveo Cloud Architecture

Coveo for Sitecore Scaling

As seen in the Installation documentation, Coveo for Sitecore installs several files and items in the Sitecore instance. Your deployment tool and methods, as well as your Sitecore hosting solution, affects the deployment and scaling process of the Coveo for Sitecore installation. Hence you should follow the scaling guide in detail.

Here is a simple diagram to help you visualize Coveo in a scaled environment.



Coveo Cloud Organization Migration

Coveo Cloud will scale on demand. You only have to manage the migration between your environments. A standard license contains more than one environments; you can refer to this post for tips and tricks on how to set them up according to your Sitecore environments.

When migrating from one organization to the other, most of the options in Coveo Cloud can be exported, either in JSON or CSV, depending on the option. You can copy a whole Query Pipeline or a whole source configuration. This mechanism can help moving configuration from a Sandbox organization to a Production organization.

PRO TIP: Migrating content between environments can cause side effects. Do not plan this step too close to your go-live date. Going live with a scaled Sitecore environment is a complex task, and even if Coveo is a cloud service, things can get complicated based on your architecture.

URL Management

URL management is a complex topic in Sitecore. Following the Coveo for Sitecore scaling guide is essential for URL to be appropriately resolved on both the CM and the CD servers. If you have completed the scaling steps and the clickable URLs on your documents are not resolving correctly, you should read this article on how the value is computed.

Pitfalls

The most common mistakes

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.

MISCONCEPTIONS	FACTS
Providing a search-as-you-type experience is the standard approach in modern search technology.	The most useful and common experience is called "type-ahead." The suggestions are queries, not results. It's fast, light, and it directs the users towards a successful query.

Sorted Lists

Powering sorted lists with search is not always the most effective solution.

MISCONCEPTIONS	FACTS
Using Coveo to retrieve data is more straightforward and faster than with Sitecore database requests. Using Coveo to get a list of news sorted by date is effective, since it leverages ML and other ranking tools.	Coveo is a relevance tool, and relevance is only applied when the "sort by relevance" option is activated. Using Coveo to build simple lists can be a misallocation of resources, especially if your license is limited. If you are planning to make a “Trending” list based on popularity, consider using the Coveo ML Recommendation list.

Using Ranking Tools Without Data

As seen in the Relevance Tuning section of this guide, 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.

MISCONCEPTIONS	FACTS
Boosting a template will help relevance.	This misconception is based on the fact that people are looking for that specific template, which can (and often is) wrong. The best strategy if you do not want to spend too much time looking at the data, is to use Coveo Machine Learning ART.
I want to find only one type of item, so I will boost that item to a high value.	When applying boosting options, go incrementally to observe every change. If needed, leverage Coveo A/B testing.

Pre-selecting Options

To promote certain products or articles, some designs are pre-selecting some values for the user in the search interface.

MISCONCEPTIONS	FACTS
Users are usually searching for this type of item	This is where Coveo shines. Leave the user to navigate and look at the analytics afterward. Many implementations are maintaining components not even used or wanted by the end user.

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 can be useful when primarily searching for people, but better results can be achieved with machine learning and query suggestions.

MISCONCEPTIONS	FACTS
It is the only way to get results for a partially entered keyword.	The user won't be entering partial keyword as they will have query suggestions, which works with the wildcard concept, but in a much more efficient way.
Users won't find what they are looking for without wildcarding.	When using wildcarding, more results are returned, and most of them will be out of context. The point here is not to find more items, but to find the relevant one!

Result Suggestions

Result suggestions is a search-as-you-type experience in a compact search interface. It can be truly useful in a commerce or a business to consumer (B2C) website, where there is a defined catalog of product, e.g., a movie streaming application or commerce website.

MISCONCEPTIONS	FACTS
It is a nice addition to a global search box.	Having a search box providing query suggestions will redirect the user to a search page. Having some result suggestions will redirect them to the item page. This dual behavior can be confusing, so using only a result suggestion is recommended. Since a complete search interface uses the wildcard, it can lead to a high QPM consumption.

Multiple Inputs

Multiple Inputs are often used in people search, where the designers will show one input per searchable fields.

MISCONCEPTIONS	FACTS
It is useful to provide some different inputs based on the fields needed to be searched.	There is nothing right about having multiple inputs for a function. It will take more time to fill, and the chances of errors are increased.

Developing Against the Coveo Search API Directly

Since Coveo Cloud is a platform, all the API are exposed and can easily be used to do custom functions. However, this requires a high level of expertise and understanding of the product.

MISCONCEPTIONS	FACTS
My requirements are simple, so it will be easier and faster to create a custom interface.	The simpler the requirements are, the easier it will be to use the controls offered by the product.
My requirements are complex, so it will be easier and faster to create a custom interface.	The controls are also extendable, so they can fit even complex requirements. Using the Search API directly is a good approach for solutions when integrating the Coveo for Sitecore Hive framework or the Coveo JavaScript framework is not possible. One example is when integrating Coveo content in a desktop product or mobile app: https://www.coveo.com/en/solutions/in-product-intelligence

Acknowledgments

The people behind the scenes.

First, I would like to thank my manager, Simon, for helping me put all the content together. Without his precious help, this document would not be as precise and effective as it is.

I would also like to thank the R&D team who helped me understand the product and were very indulgent to me and my myriad of questions.

Finally, this document is the result of a collaboration between several people at Coveo, and it would not be so good without them, so thank you all for your valuable contribution.

A special thanks to all of our partners and customers who did an incredible job implementing this solution without any guide. You are clearly resourceful and inventive. Cheers!

Vincent.

Credits

Writers	Vincent Bernard - Lead Architect Simon Langevin - Sr. Product Manager
Reviewers	Nep Vijayaraja - Product Marketer Jean-Francois L'Heureux - Software Developer David Auclair - Product Analyst Alexandre Moreau - Solution Architect
Designers	Tanya Lam - Web Designer
Others	Francois Lachance-Guillemette - Coveo for Sitecore Developer