Mirasvit Search Ultimate User Manual

Search Ultimate Documentation

Here you will find everything you need to set up a better search experience.

The **Search Ultimate** extension gives you a powerful way to search through your store. Search Ultimate's set of modules includes:

1. **Elastic Search**
2. **Sphinx Search**
3. **MySQL Search**
4. **Search Spell-Correction**
5. **Search Autocomplete**
6. **Search Reports**
7. **Landing Pages**

Start with the Installation and Quick Start options. It is best to follow our step-by-step guide to configure the best search results.

Go ahead, dive in!

Learn about the initial setup:

- Installation
- Quick Start

Installation

Installation via composer (preferably)

We recommend this installation method because the composer automatically checks and installs necessary dependencies and is also much easier to keep the extension up-to-date.

1. Back up your store's database and web directory.
2. Log in to the SSH console of your server and navigate to the root directory of the Magento 2 store.

   If you bought the extension in the Magento Marketplace, run the command in the root folder of your store.

   ```
   composer require mirasvit/module-search-ultimate
   ```

3. To enable the extension, run the commands:

   ```
   php -f bin/magento module:enable Mirasvit_Core Mirasvit_Search
   Mirasvit_SearchMysql Mirasvit_SearchElastic Mirasvit_SearchSphinx
   Mirasvit_SearchAutocomplete Mirasvit_Misspell Mirasvit_SearchLanding
   Mirasvit_Report Mirasvit_SearchReport
   php -f bin/magento setup:upgrade
   ```

4. Clean the cache

```
php -f bin/magento cache:clean
```

5. Deploy static view files

```
rm -rf pub/static/*
rm -rf var/view_preprocessed/*
php -f bin/magento setup:static-content:deploy
```

6. Reindex search index and spell-correction index

```
php -f bin/magento indexer:reindex catalogsearch_fulltext mst_misspell
```


Quick Start

Search Ultimate is our most powerful extension which combines all of our best features to enhance the search capabilities of Standard Magento.

It combines several powerful modules, each of which offers you a large set of options:

1. **Core Search Module** - contains everything necessary to enhance search and display of its results.
2. Search Engines
    1. **Elasticsearch** - extends native Magento Elasticsearch engines.
    2. **Sphinx Search** - contains an easy and rich interface for the Search Sphinx engine.
    3. **MySQL Search** - implements search functionality using MySQL database.
3. **Search Autocomplete** - allows you to enhance your search by adding suggestions at the customer's fingertips.
4. **Search Spell Correction** - allows you to correct on-the-fly customer misspellings, and ensure that virtually any request will be satisfied.

Here is how you can tune up our extension for maximum effectiveness.

1. Tune-up **Core Search Settings**. Start by creating Indexes and configuring Search Results.

    After you have created Indexes, it's better to quickly reindex all data. The best way to do it is via console/SSH:

    ```
    php -f bin/magento indexer:reindex catalogsearch_fulltext
    ```

2. Consider which search engine you're about to use.
3. Add to your store Stopwords and Synonyms to enhance search. You can even import them using command-line interface.
4. Create a set of Landing Pages to create convenient hubs for your most important products.
5. Enhance your search by creating rules for 'long-tail search' and ordering products at search results.
6. Configure Autocomplete to provide your customers with useful suggestions and searching tips.
7. Enable misspell correction to provide customers with results even when they make mistakes.

8. Analyze your searches with our Reports and build flexible search policies to boost your store capabilities to the top.

Please follow our guide step by step to get the best search result!

General Settings

Here you can quickly navigate across all the functionality settings we have available. Please use the list below to navigate.

This section covers all topics necessary for working with indexes, and consists of the following subsections:

- Global Search Settings
  - Search Engine Configuration
    - **Sphinx Search Engine**
      - Installation
      - Connection with Sphinx Engine
    - **Elastic Search Engine**
      - Installation
    - Search Settings
    - Multi-store Search Result
  - "Long-Tail" Search
  - Landing Pages
  - Synonyms
  - Stopwords
  - Customize Search Weight
- Search Indexes Settings
  - Managing Indexes
  - Adding New Index
  - Product Index
  - Category Index
  - CMS Index
  - Attribute Index
  - Wordpress Blog Index
  - Add Custom Index

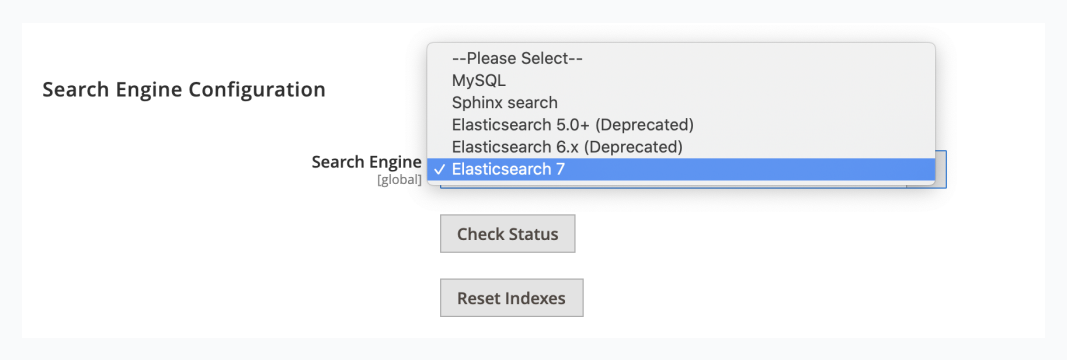
Configure Key Search Settings

This section describes how you can customize and greatly improve the relevance of your search results by configuring Search Settings.

The most important part is the **Search Logic Configuration**. It is located at **System -> Search Management -> Configuration**.

Search Engine Configuration

Our extension allows you to power up your search either with default Magento Elasticsearch engines or additional engines.



Option **Search Engine** selects which engine should be in charge, and has three possible values:

- **Elasticsearch 7**, **Elasticsearch 6.x**, **Elasticsearch 5.0.x** - native Magento search engines.
- **MySQL** - search engine, based on MySQL fulltext search.
- **Sphinx Search engine** - search engine, based on Sphinx Search.

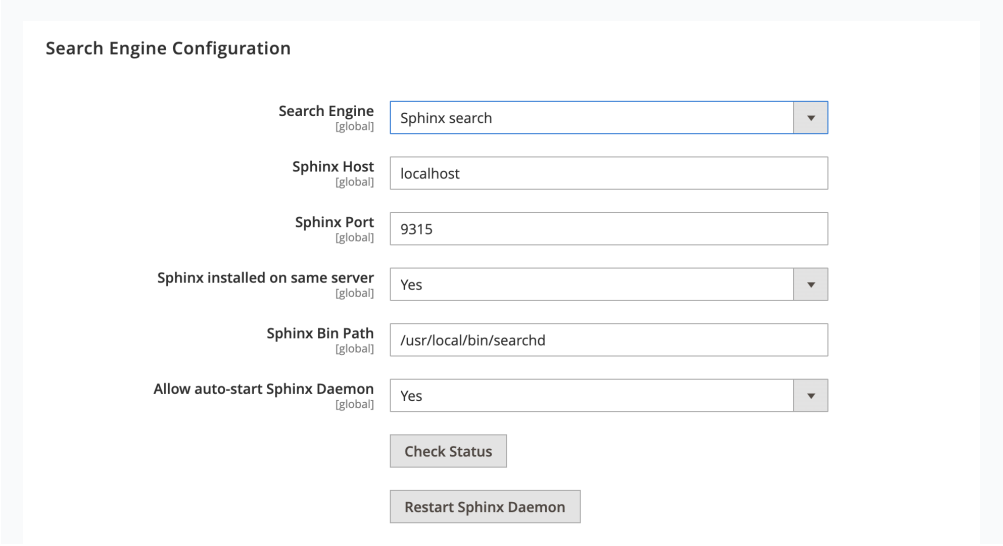
You can manage your store engine status and indexes:

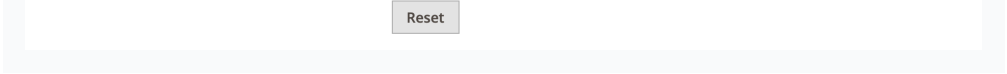
- **Check Status** - describes whether the search engine is running on the server or not.
- **Reset Indexes** - deletes all available indexes. Products will disappear from the frontend if you click it. Run reindex to restore indexes.
- **Reset Store Indexes** - deletes store indexes by Elasticsearch Index Prefix. Products will disappear from the frontend if you click it. Run reindex to restore indexes.

> **Note**
>
> **MySQL search engine** mode **does not** require the installation of Elasticsearch or Sphinx Search on your server, but you will still receive the same features as with the other engines. However, you may experience a slower search speed than with the Elastic/Sphinx engines. This engine is applicable for small catalogs, or if your server doesn't allow the installation of Elasticsearch or Sphinx Search.

Sphinx Search Engine

Sphinx is an open-source full-text search server which features high performance, relevance (aka search quality), and integration simplicity. It's written in C++ and runs on Linux (RedHat, Ubuntu, etc), Windows, macOS, Solaris, FreeBSD, and a few other systems. It is best used for stores with product quantities below 50k who do not require layered navigation or aggregated search requests. Read more about this engine key features.
Sphinx Search Ultimate adds to the option **Search Engine Configuration** -> **Search Engine** possible value **Sphinx Search**.



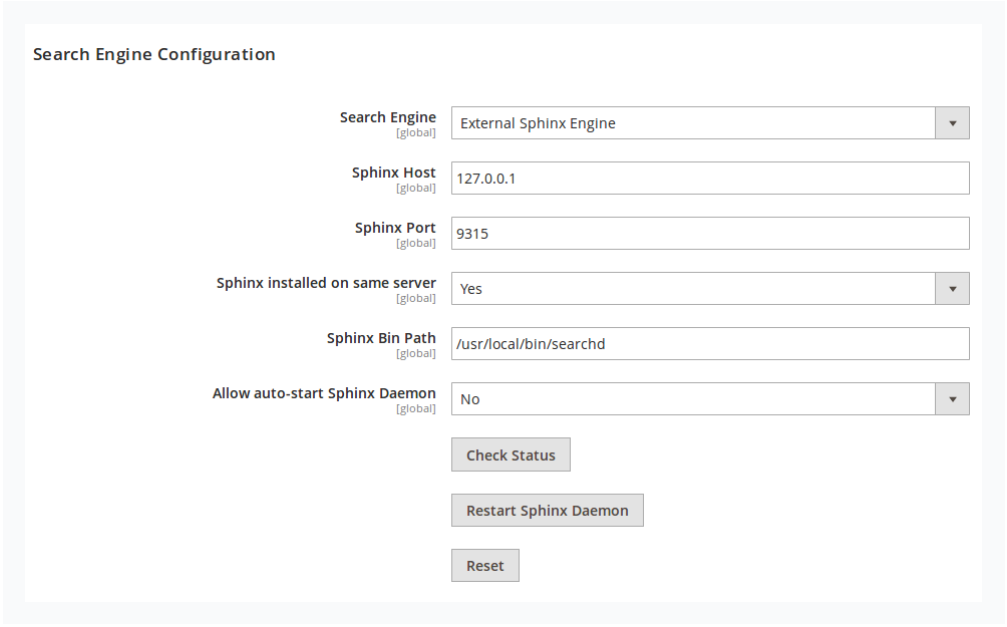


> **Note**
>
> To start, please make sure that you have installed Sphinx Search Engine. To do this, please follow our installation guide

External Sphinx Engine also triggers additional options for configuring and managing Sphinx Daemon:

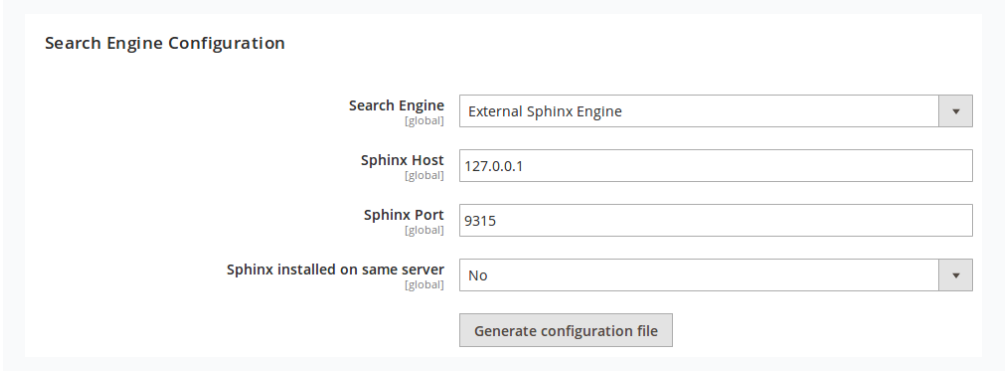
- **Sphinx Host** - sphinx daemon host (localhost by default).
- **Sphinx Port** - sphinx daemon port (any free port, like 9811, 9812).
- **Sphinx installed on the same server** - triggers the appearance of additional features of Sphinx Daemon. Can have two different modes:

For Sphinx installed on the same server with your Magento store :



- **Yes** - defines that Sphinx works on the same server as store and database. Triggers the following additional options and buttons, which allows the managing daemon:
  - **Sphinx Bin Path** - defines the name and location of sphinx daemon. By default, it's **searchd**.
  - **Allow auto-start Sphinx Daemon** - sets auto-starting daemon with Magento's store. Useful when you have an unexpected server power-off (for example, for maintenance purposes).
  - **Check Status** - button that allows viewing current daemon status
  - **Restart Sphinx Daemon** - button which allows restarting daemon directly from the Magento Configuration pane.
  - **Reset** - button that allows resetting the daemon current search task.

For Sphinx installed on the dedicated (remote) server :

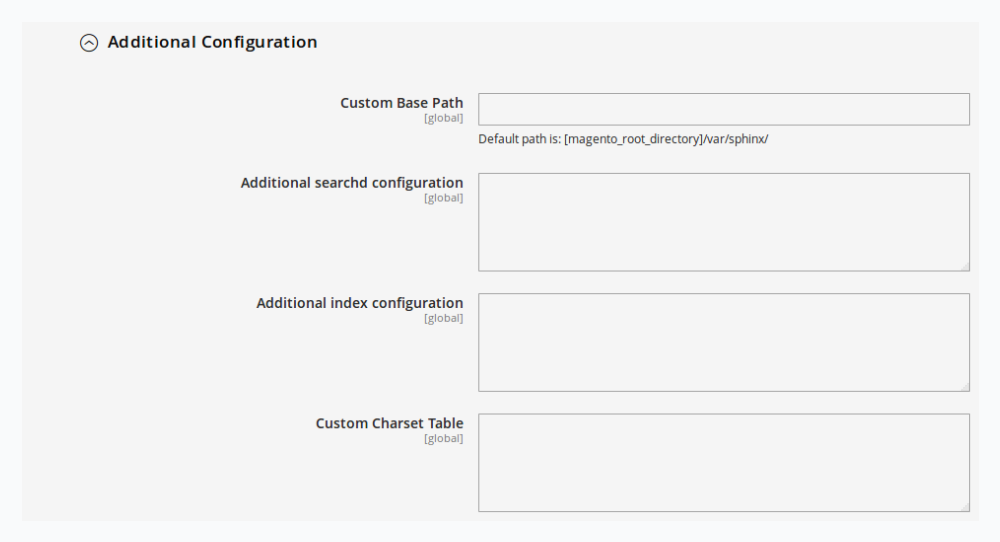


- **No** - defines that Sphinx works on separate or dedicated server.
  - **Generate configuration file** - button that allows you to generate Sphinx config file to copy to

your remote (dedicated) server.

> **Note**
>
> If you already installed **Sphinx Search engine**, please learn more about the connection with Sphinx Engine.

The **Search Engine Configuration** section contains an **Additional Configuration** subsection, visible for **External Sphinx engine** only. It allows you to tune up the Sphinx configuration file, and contains the following settings:

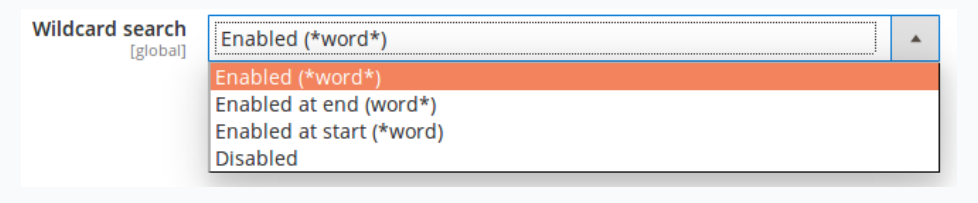


- **Custom Base Path** - defines a custom path to Sphinx, if it was not installed to the default `[magento_root_directory]/var/sphinx/` location.
- **Additional searchd configuration** - defines additional parameters to `searchd` Search Daemon. Read more about it here.
- **Additional index configuration** - allows you to add settings to the Sphinx index configuration. Read more about it here.
- **Custom Charset Table** - allows for adding character sets to the Sphinx configuration file. Read more about it here.

Elastic Search Engine Elasticsearch is a distributed RESTful search and analytics engine capable of solving a growing number of use cases, written on Java so it can be run virtually anywhere. It is best used for stores with more than 50k of products and/or support of Layered Navigation. Read more about its key features.
You can configure a connection using native magento interface at **Stores -> Configuration -> Catalog -> Catalog Search**
The extension also features two buttons that allows you to check the actual Elastic Search status and reset indexes:

- **Check Status** - button that allows for viewing current Elastic status
- **Reset Indexes** - button that removes Elastic search indexes. To restore indexes, just run a full search reindex.

Search Logic Configuration



- **Wildcard search** - allows the customer to search the product by part of a word, marking unknown parts with asterisk (*). There are four different wildcard modes available:

  - **Enabled (*word*)** - fully enables wildcards.
  - **Enabled at the end (*word*)** - partially enables wildcards, allowing you to search by the first part of a keyword.
  - **Enabled at the start (*word*)** - partially enables wildcards, allowing you to search by the last part of a keyword.
  - **Disabled** - totally disables wildcards.

  > 💬 Note
  > Wildcards enabling slightly reduces the relevance of searches and increases the number of search results.

- **Wildcard Exceptions** - the list of keywords (characters) for which wildcard search can not apply.
- **Replace words in search query** - two-column list of auto-replace. When evaluating, the search extension will seek keywords from **Find word** columns, and automatically replace them with one from the **Replace With** column. Column **Find word** can contain more than one keyword, separated by a comma.
- **Not' words** - words from this list invert search–e.g., the appearance of these words in a search are automatically treated as "exclude results with this word".
- **Long Tail Expressions** - allows you to receive the correct search results for words that contain dashes or any other non-alphabetic symbols. Read more in the Long Tail Configuration section.
- **Match mode** - overrides the default Magento mode of search with one of the following options:
  - **AND** - this mode is **default**. Elements (e.g. products, pages) matched only when all requested keywords are found in respective attributes.
  - **OR** - defines that elements matched only when at least one of the requested keywords is found.

Search Features

- **Enable redirect from 404 to search results** - if this option is enabled, the customer will be redirected to the store search results of the broken URL text instead of the "404 Not Found" page.
- **Redirect if Single Result** - if the search query results only have one match, the customer will be immediately taken to the corrresponding product page.
- **Enable Google Sitelinks Search** - if the option is enabled, the extension shows the Sitelink Search Box on the Google search results page. After enabling this option, the search box will be shown only after Google reindexing.
- **Enable search terms highlighting** - if this option is enabled, search query word(s) will be highlighted in the search results.
- **Display Search Recommendations** - if this option is enabled, related search terms will be displayed on the search result page.
- **Omit tabs when the number of results fewer than** - indicates when an extension should display search indexes as tabs.

Multi-store Search Result

This option is useful when you have store-dependent elements in your store. For example, you have products which are visible only in specific storeviews and you wish to allow customers to search simultaneously in all your stores.

- **Enable Multi-Store Search Results** - if you enable this option, search results will be displayed in tabs. Each tab has a number of results for a storeview and corresponds to one of your storeviews. This option triggers an additional sub-option:

  - **Stores** - allows you to select which storeviews should be included in a multi-store search.

  > 💬 Note
  > Multi-store search results work only on the search results page (when visitors click on the tab

Multi-store search results work only on the search results page (when visitors click on the tab, it redirects them to the selected storeview).

Autocomplete always returns results from the current store only. It can not display search results from another storeviews.

Sphinx Engine Installation

If you want to use our extension with Sphinx Engine, you first need to install it.

> 💬 Note
>
> Warning: The minimum allowed sphinx engine version is **2.2.x, 3.x**

Installation includes a set of actions that should be performed under **root** privileges. Actions can differ depending on the selected platform.

After installation, **run full reindex of Sphinx Engine indexes** to fully enable your new search engine.

Installation on Ubuntu

Execute the following command from console/SSH under root privileges:

```
sudo apt-get install sphinxsearch
```

Proceed to Connection with Sphinx Engine

Installation on CentOS

Execute the following command from console/SSH under root privileges:

```
sudo yum install sphinxsearch
```

Proceed to Connection with Sphinx Engine

Installation with Stemmer (with language specific morphology)

In order to install Sphinx with Stemmer, run the following commands from console/SSH under root privileges:

```
wget http://sphinxsearch.com/files/sphinx-2.2.11-release.tar.gz
tar xvf sphinx-2.2.11-release.tar.gz
cd sphinx-2.2.11-release
wget http://snowball.tartarus.org/dist/libstemmer_c.tgz
tar xvf libstemmer_c.tgz
./configure --with-libstemmer
make
make install
```

Learn more about Libstemmer morphology

Libstemmer controls which characters are accepted as valid and which are not, and how the accepted characters should be transformed (eg. should the case be removed or not).
If charset table is configured for special language chars, then search phrases "kottkvarn" and "köttkvarn" will return the same result.
Libstemmer supports morphology of the following languages:

- Danish
- Dutch

- English
- Finnish

- French
- German
- Hungarian
- Italian
- Norwegian
- Polish
- Portuguese
- Romanian
- Russian
- Spanish
- Swedish
- Turkish

To configure morphology and charset tables for additional languages, perform the following actions:

- Open the file `/vendor/mirasvit/module-search-sphinx/src/SearchSphinx/etc/conf/index.conf`.
- Configure `morphology` variable. Standard language codes you can find here. Use two-letter codes from **639-1** column.

> **Tip**
>
> For example, to add German to the list of supported languages, you need to set in `index.conf`:
>
> `morphology = stem_enru, libstemmer_de`
>
> Read more about morphology here.

- Add charsets to the `charset_table` variable. List of codes for different charset tables can be found here

> **Tip**
>
> For example, to add English and Russian characters, variable should look as shown below:
>
> ```
> charset_table = 0..9, A..Z->a..z, _, a..z, \
>                 U+410..U+42F->U+430..U+44F, U+430..U+44F, U+401->U+451,
> U+451
> ```
>
> Read more about character tables here.

Proceed to Connection with Sphinx Engine

Manual installation

In some cases, automatic installation either does not start, or even is not possible. In those cases, you can manually install Sphinx:

```
wget http://sphinxsearch.com/files/sphinx-2.2.11-release.tar.gz
tar xvf sphinx-2.2.11-release.tar.gz
cd sphinx-2.2.11-release
./configure
make
make install
```

Configuration file will be generated automatically.

Proceed to Connection with Sphinx Engine

Connection with Sphinx Engine

After the Sphinx Engine is installed, you need to connect it to our Search Extension.

Settings are diffenent depending on where you had installed Sphinx - on the same server as a store, or on a dedicated separate server.

Connect with Sphinx Engine on the same server

1. Go to **System -> Search Management -> Settings** and proceed to **Search Engine Configuration**.
2. In the field **Search Engine** select **External Sphinx Engine** option, and fill in the following fields.
   - **Sphinx Host** - sphinx daemon host. Should be set to **localhost** in this case.
   - **Sphinx Port** - sphinx daemon port (any free port).
   - **Sphinx Bin Path** - if "searchd" is not configured in shell paths on your server, here you need to enter the full path to "searchd" (ex. `/usr/local/bin/`).
3. Tune up your Search Daemon, using extended options in **Additional Configuration** subsection.

Connect with Sphinx Engine on another server

To establish a connection with sphinx engine on another server, you need to run sphinx engine with an auto-generated configuration file.

Go to **System / Search Management / Settings / Mirasvit Extensions / Sphinx Search.**

1. Select another server option and press the button "Generate configuration file".
2. Copy the configuration file to the sphinx server (same path to file is required)
3. Start sphinx daemon using command

```
searchd --config <path to config/sphinx.conf>
```

   - Open a port for connection between servers. You can use the below command to check if the port opened:

```
nc -zv sphinx_server_ip sphinx_server_port
```

4. Run the search reindex in **System / Search Management / Search Indexes**

Configure "Long-Tail" Search

This section describes the Long-Tail Search feature which will allow you to have the correct search results for words that contain dashes or other non-alphabetic symbols. You can also replace the most typical errors customers make in complex product names on the fly .

What is Long-Tail Search?

For example, we have a product `Canon PowerShot SX500 IS`. The customer can request `Canon PowerShot SX-500IS`, which a default search will not find, because it differs from the actual product label.

This is because Magento by default during reindex uses only correct product labels from the database, thus, ensuring the index will contain only them - making products with complex names "ineligible" for search.

This is where "Long-tail" search comes in. During the reindex and search, this feature recognizes keywords by pattern and replaces them either with empty space or some other characters, "correcting" customer's request in real time.

In the example above, the `SX500 IS` can be converted to the `SX500IS` and during the search, the `SX-500IS` is also be converted to the `SX500IS` by replacing the '-' symbol with empty char.

This way, the search will be able to find products by several combinations of spelling the product's name.

Configuring Long-Tail Search

Go to **System / Search Management / Settings / Mirasvit Extensions / Search**
In the section **Search Settings**, go to the option **Long tail**.
There you can set up regular expressions to receive required search results.

- **Match Expression** - the regular expression(s) that parses words for further replacing.

  Parsing is used for search index, during an indexing process, and goes for search phrases during a search. E.g. `/([a-zA-Z0-9]*[\-\/][a-zA-Z0-9]*[\-\/]*[a-zA-Z0-9]*)/`
- **Replace Expression** - the regular expression(s) for parsing characters to be replaced. Parsing goes in the results of "Match Expression". E.g. `/[\-\/]/`
- **Replace Char** - the character to replace values founded by "Replace Expression". E.g. `empty value`.

Configuring Long-Tail Search

Here are some of the most useful cases of long-tail search, implemented as corresponding rules.

Automatically remove '-' symbol from product names

Create a rule with the following parameters:

- **Match Expression** - `/[a-zA-Z0-9]*-[a-zA-Z0-9]*/`
  Matched text: `SX500-123`, `GLX-11A`, `GLZX-VXV`, `GLZ/123`, `GLZV 123`, `CNC-PWR1`
- **Replace Expression** - `/-/`
- **Replace Char** - empty
  Result text: `SX500123, GLX11A`, `GLZXVXV`, `GLZ/123`, `GLZV-123-123`, `CNCPWR1`

Automatically remove '-' and '/' symbols from product names

Create a rule with the following parameters:

- **Match Expression** - `/[a-zA-Z0-9]*[ \-\/][a-zA-Z0-9]*/`
  Matched text: `SX500-123`, `GLX-11A`, `GLZX-VXV`, `GLZ/123`, `GLZV 123`, `CNC-PWR1`
- **Replace Expression** - `/[ \-\/]/`
- **Replace Char** - empty
  Result text: `SX500123`, `GLX11A`, `GLZXVXV`, `GLZ123`, `GLZV123`, `CNCPWR1`

Automatically make solid all products names with separators

Create a rule with the following parameters:

- **Match Expression** - `/[a-zA-Z0-9]*[-\/][a-zA-Z0-9]*([-\/][a-zA-Z0-9]*)?/`
  Matched text: `SX500-123`, `GLX-11A`, `GLZX-VXV`, `GLZ/123`, `GLZV-123-123`, `CNC-PWR1`
- **Replace Expression** - `/[-\/]/`
- **Replace Char** - empty
  Result text: `SX500123`, `GLX11A`, `GLZXVXV`, `GLZ123`, `GLZV123123`, `CNCPWR1`

Automatically fix misspelled product's name

Create a rule with the following parameters:

- **Match Expression** - `/([a-zA-Z0-9]*[\- ][a-zA-Z0-9]*[\-][a-zA-Z0-9]*)/`
  Matched text: `VHC68B-80`, `VHC-68B-80`, `VHC68B80`

- **Replace Expression** - `/[\- ]/`

- **Replace Char** - empty
  Result text: `VHC68B80`

Moving Long-Tail Expressions from M1 to M2

Long-Tail expressions, which are used in Search Sphinx for M1 and M2 slightly differ.

In M1 Search Sphinx, you can enter one or more expressions to match, separated by '|' character. In M2, you can not.

Consider the following expression for Search Sphinx for M1:

> **Example**
>
> **Match Expression**:
> `/[a-zA-Z0-9][ -/][a-zA-Z0-9]([ -/][a-zA-Z0-9]*)?/|/[a-zA-Z]{1,3}[0-9]{1,3}/`
> **Replace Expression**: `/[ -/]|/([a-zA-Z]{1,3})([0-9]{1,3})/`
> **Replace Char**: `$1 $2`

It actually contains two separate regex to match:
`/[a-zA-Z0-9][ -/][a-zA-Z0-9]([ -/][a-zA-Z0-9]*)?/` and `/[a-zA-Z]{1,3}[0-9]{1,3}/` with respective separate expressions for replace.

You need either to reformat that expression, so it will match in single expression, or rewrite this rule as a set of two:

First rule

This rule will implement the first part of the original M1 expression.

- **Match Expression**: `/[a-zA-Z0-9][ -/][a-zA-Z0-9]([ -/][a-zA-Z0-9]*)?/`
- **Replace Expression**: `/[ -/]/`
- **Replace Char**: `$1 $2`

Second rule

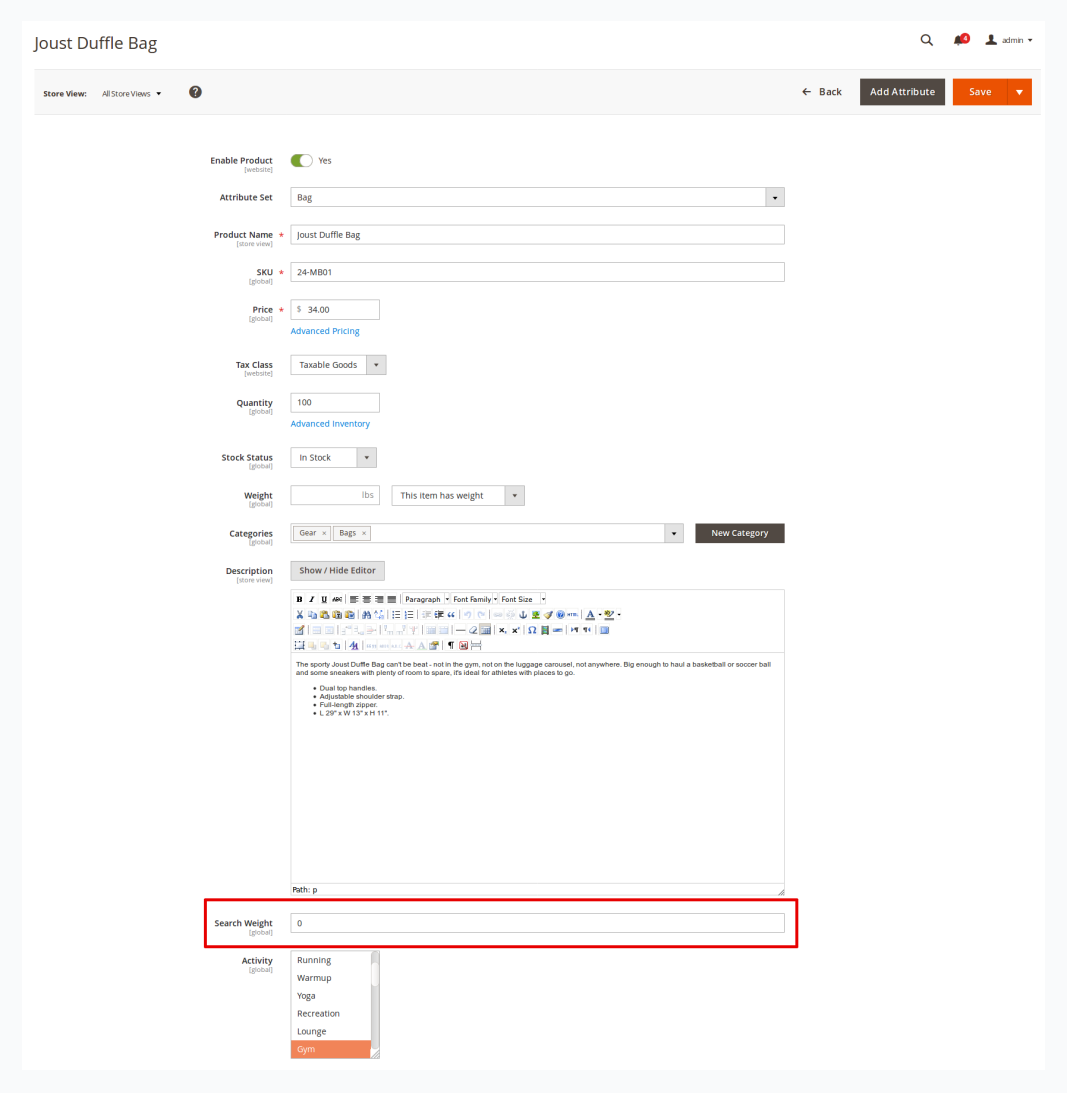
This rule will implement the second part of original M1 expression.

- **Match Expression**: `/[a-zA-Z]{1,3}[0-9]{1,3}/`
- **Replace Expression**: `/([a-zA-Z]{1,3})([0-9]{1,3})/`
- **Replace Char**: `$1 $2`

Customize Search Weight

Our extension arranges the relevance of products found using Global Settings. However, sometimes (for example, for promotional purposes) you need to forcibly move one or more specific products to the top, or vice versa, to the bottom of search results.

It can be done via special option **Search Weight**, added by our extension to the general settings of the Product Edit Pages.



This weight is the relative position in which the product will be placed on the search result page. It ranges from 100 (product or category will always appear at the top of the search results list) to -100 (product or category will always appear at the bottom of the search results list).

Configure Search Indexes

Search Indexes are the most important part of your search subsystem. The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. Without an index, the search engine would scan every document in your store, which would require considerable time and computing power.

This section covers all topics necessary for working with indexes, and consists of the following subsections:

- Managing Indexes
- Adding New Index
- Product Index
- Category Index
- CMS Page Index
- Attribute Index
- Wordpress Blog Index

Managing Indexes

Our search can combine all indexes existing in your configuration to boost the search and give your customers the most relevant results. It brings them all to a single grid, located at **System -> Search Management -> Search Indexes**, from which you can configure them.

Each index, added to this grid, displays the following properties:

- **Title** - title of the search index.
- **Type** - shows the index type (searchable content type - read more at **Adding New Index** subsection).
- **Position** - the position of the index in the search results. Search results will be organized in tabs according to this property.
- **Status** - indicates whether the current index is ready for searching. The **Disabled** value means that particular index will be excluded from the search.

Additional **Action** column provides common actions that can be performed directly from the grid, such as:

- **Edit** - edit index settings (default action).
- **Reindex** - run manual reindexing for selected index.
- **Delete** - remove index from Mirasvit Search extension. %

> 💬 Note
>
> This action will completely remove this index from your store, so if you wish for the index to be excluded from search - just change its status to **Disable**.

Adding and Configuring New Index

1. To add a new index to the Mirasvit Search extension, go to **System -> Search Management -> Search Indexes** and press **Add New Index**.

2. Index record creation is divided into two stages: setting common settings and specific, which depend on their type. Common settings are shown in the **General Settings** subsection:

    - **Title** - title of the search index. It will be used as a tab header at the search display page.
    - **Type** - shows the index type (searchable content type). Some values of this field will trigger specific options. Pick a link from type list below to know more:
        - **Magento Indexes**
            - Product
            - Category
            - CMS Page
            - Attribute
        - **Custom Search Indexes**
            - Wordpress Blog
        - **Mirasvit Extensions**
            - Blog MX
            - Knowledge Base
            - Gift Registry
        - **Magefun Blog Extension**
        - **Mageplaza Blog Extension**
        - **Ves Extensions**
            - Blog
            - Ves Brand
        - **Amasty**

        - Blog
        - FAQ
    - **Blackbird Content Manager**
- **Position** - the position of the index in the search results. The extension will sort tabs on the search results page based on the position.
- **Active** - sets whether the index should be activated.

3. Press **Save and Continue Edit** to proceed to the index configuration stage.

4. Add **Searchable Attributes** to the type-dependent option list, with rows corresponding to attributes where the extension should conduct a search. Each row consist of the following fields:
    - **Attribute** - attribute name. It is picked from properties of selected index type. For example, if **Product** type is selected - then attributes would be **Product Name**, **SKU**, **Price**, **Tax Class** and so on.
    - **Weight** - sort order, which defines the importance of each attribute for product relevancy. The maximum weight is 10 (highest priority), the minimum weight is 0(lowest priority). Each index type comes with a predefined set of searchable options that will be displayed after completing the first stage. There should be **at least one searchable attribute**, otherwise the search will not work properly.

5. **Properties** - type-dependent specific options section. Read more below, or pick a link from type values, described in **(2)** step.

6. Save the index and activate it to be included in the search.

During installation, three indexes will be automatically created and configured: **Product**, **Category** and **CMS Page**.

Product Index

The Product Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Specific options of this type will be shown on the **Properies** section of the Index edit page:

- **Search by Parent Categories Name** - include in the search all parent categories (useful when a store has a wide categories tree).
- **Search by child products** - include associated products from Bundled, Grouped and Configurable products in the search.
- **Search by Product ID** - enable search by product id (entity_id attribute, which is not listed as searchable by Magento).
- **Search by custom options** - enable search by custom options (defined in addition to existing options).

Category Index

Category Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's no specific options for this type of index.

CMS Index

CMS Page Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

There's only one specific option for this type on the **Properies** section of the Index edit page:

- **Ignored CMS Pages** - defines, on whose CMS pages search should not be performed. You can select zero or more pages here via the checkbox drop-down list.

Attribute Index

Unlike other indexes, this one can be created only for a specific attribute which should be displayed as a separate section in the Search Results.

This attribute should be previously enabled for Advanced Search. It can be done at the **Stores -> Attributes -> Product** grid. Pick up the desired attribute, then jump to the **Storefront Properties** subsection and make it available for search by setting to **Yes** two options: **Use in Search** and **Visible in Advanced Search**.

> 💬 Note
>
> Attribute index can work only with attributes that can be **indexed**, e.g. they belong to a selectable type.
>
> To see type of Product Attribute, visit the **Stores -> Attributes -> Product** grid, pick up the attribute record, and see the **Catalog Input Type for Store Owner** field. Selectable types are **Multiple Select** and **Dropdown**. Only attributes of this type can be indexed.
>
> If you wish to use attributes like **Author**, or something similar, you have to make them selectable first, and then make them available for searching as above.

After saving the product, you can configure Attribute Index for this attribute at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

Wordpress Blog Search Index

Wordpress Blog Index can be created at the **System -> Search Management -> Search Indexes** grid. Read more about it here: adding new index.

- **Database Connection Name** - connection name of the Wordpress database.
  - If WordPress is installed on the same database, the correct value is `default`.
  - If WordPress is installed on a separate database, you need to create a new connection in the file `app/etc/env.php`.

    > 🏢 Example
    >
    > A typical database connection should look similar to this:

```
'db' => array(
'table_prefix' => '',
'connection' => array(
    'default' => array(
        'host' => 'localhost',
        'dbname' => 'store',
        'username' => 'root',
        'password' => 'password',
        'active' => '1',
    ),
 ),
),
```

- **Table Prefix** - the prefix for the Wordpress tables (wp_ by default).
- **Url Template** - the full URL for your posts with dynamical variables.

  Typical base urls should look like the example below:

```
http://example.com/blog/{post_name}.html
http://example.com/blog/?p={ID}
http://example.com/{category_slug}/{post_name}.html
```

Implementing Custom Search Index

Sometimes it's necessary to have a specific type of Index which is either not included to our search extension, or belongs to some third-party extension. In this case, custom index can be implemented using the following instructions:

1. Clone the example module from the repository https://github.com/mirasvit/module-search-extended.git
2. Go to `app/code/Mirasvit/SearchExtended/Index/` and rename the subpath `Magento/Review/` to the required one. We suggest to use the following structure - `([provider]/[module]/[entity])`
3. Change class names in file `app/code/Mirasvit/SearchExtended/Index/[provider]/[module]/[entity]/Index.php`
   - Set your values to `getName()`, `getPrimaryKey()` and `getIdentifier()` methods
   - Configure the attributes you want to get in `getAttributes()` method
   - Change `buildSearchCollection()` and `getIndexableDocuments()` methods
4. Change class names in file `app/code/Mirasvit/SearchExtended/Index/[provider]/[module]/[entity]/InstantProvider.php`
   - Set your values to `map()` and `mapItem()` methods
5. Change registration for new index in file `app/code/Mirasvit/SearchExtended/etc/di.xml`
6. Adjust layout file `app/code/Mirasvit/SearchExtended/view/frontend/layout/catalogsearch_result_index.xml`
   - Rename the template name/path and adjust it `/app/code/Mirasvit/SearchExtended/view/frontend/templates/index/magento/review/review.pht`
7. Adjust layout file `app/code/Mirasvit/SearchExtended/view/frontend/layout/default.xml`
   - Rename the template name/path and adjust it `/app/code/Mirasvit/SearchExtended/view/frontend/templates/magento_review.phtml`
8. Enable the module and Clear magento cache

If everything was performed correctly, you should be able to add index of your custom type like any regular index.

Configuring Autocomplete & Suggest

All Autocomplete settings are located in the **System / Settings / Mirasvit Extensions / Search Autocomplete** section.

It consists of the following sections:

- General Configuration
- Hot searches

General Configuration

This section is broken down into smaller subsections, and contains the following options:

- **Minimum number of characters to search** - specifies the minimum number of characters which customers need to enter to trigger autocomplete.
- **The delay to start finding** - specifies the delay between triggering autocomplete (by option above) and the beginning of the actual search.

> 💬 Note
>
> Our extension begins searching for possible autocompletion and offers suggestions only when both of the following conditions have been met:
>
> - the customer has entered the minimal required number of characters;
> - no actions took place during the specified delay period.

- **Fast Mode** - This option allows you to greatly improve search speed by excluding Magento 2 from the autocomplete search at the initialization stage.

> 💬 Note
>
> - This option is available for Sphinx search and Elastic engines only;
> - The disadvantages include an increased indexing time of the search index, and the lack of some search capabilities, such as wildcard exceptions, individual search weight, Popular suggestions, Products in categories, number of found search results, etc.

- **Searchable content** - list of search indexes where the search is performed, and results are displayed as autocomplete options. Indices are either taken from standard Magento or if the extension is installed as part of **Advanced Search Sphinx Pro** or **Sphinx Search Ultimate** - from corresponding **Indexes** grid.

> 💬 Note
>
> The grid contains the following settings:
>
> - **Index** - the name of the index which can be included to autocomplete.
> - **Is Enabled** - includes current index to autocomplete
> - **Max Number of results** - the maximum number of results which should be displayed in the autocomplete drop-down widget.
>   You can drag and drop rows in this list to define the order in which results from different indices will be displayed in autocomplete drop-down.

- **Enable TypeAhead** - enables the auto-suggestion feature. Our extension collects information about the most popular search queries and their results, groups them, and stores them separately. When autocomplete is triggered and the **TypeAhead** option enabled, our application automatically searches for the typed term and displays the suggestion of the most relevant query found.

> 📢 Tip
> Use the Right Arrow button to quickly turn auto-suggestion to full autocomplete query and save autocomplete time.

- **Product Settings** - defines the content and appearance of autocomplete individual product information cells.
- **Show Price** - displays price or price range of products. Will show only final_price of the product if Fast mode is enabled.
- **Show Thumbnail** - displays a small thumbnail of the product image.
- **Show Rating** - displays a number of reviews and approval rating (so-called star rating).
- **Show Description** - displays a short excerpt from product's description.
- **Show SKU** - displays the SKU of the product.
- **Show "Add to cart"** - displays a shortcut button for quick purchasing products.
- **Optimize autocomplete view for small screen size** - allows for optimization of autocomplete layout to small screen sizes. **Note:** this may require additional style fixing in the **Appearance** section.
- **Appearance** - contains only one field, which defines custom appearance of the autocomplete widget.
- **Additional CSS Styles** - custom CSS styles, which should be applied either to the entire drop-down, or to individual product cells. It is an extremely powerful tool which allows you to match our Autocomplete extension to almost any theme.

> **Example**
>
> To customize individual product cell in autocomplete drop-down, use the following expression:
>
> ```
> .searchautocomplete__item-magento_catalog_product
> {
>     // Your extended styles
> }
> ```
>
> It will be added to our stylesheet.

Hot Searches

Hot searches are the most popular queries which have been requested by customers. If a current customer request includes such a query, autocomplete can highlight them and bring them to the top of the drop-down.

- **Search queries** - allows for overriding Hot Searches by adding special keywords (comma-separated), which should be counted as hot–this is particularly useful during promotional campaigns.
- **Ignored words** - allows for excluding certain keywords from Hot Searches. It is also a list of comma-separated words.
- **Max Number of queries** - the maximum allowed number of Hot Searches which should be displayed on autocomplete drop-down.

> **Tip**
>
> Currently, **Hot searches** are the most searched Search Terms in your Magento. You can clear some of the search terms which then won't appear in the **Hot searches** option: go to Marketing > SEO & Search > Search Terms, find the necessary term, and delete it. Otherwise, you will need to delete them in the database.

Configure Search Spell Correction

All configuration options are located in the **Store -> Configuration -> Mirasvit Extensions -> Search Spell Correction** section.

There are only two options for now. In both, our extension analyzes the customer's request and tries to find products whose names most closely resemble the original request.

- **Enable spell correction** - enables automatic spelling correction.

> **Example**
>
> Let's assume that your store has '*Samsung*' products in the catalog.
>
> When a customer accidentally misspells `Samsang phone`, the default Magento search will return nothing, since you have no such product.
>
> But with this option enabled, the customer will be notified about potential misspelling and will see the results for the corrected search phrase `Samsung phone`.

- **Enable fallback search** - enables searching for partial request satisfaction, when there are no results for the original request.

> **Example**
>
> Let's assume that customer puts the phrase `red samsung phone` into the search, but you have only `samsung phone` product.

If the store has no such product, the default Magento search also will return nothing.

But with this option enabled, the customer will be notified about the error, and will receive results by the correct search phrase `samsung phone`.

Search Results Validator

Validator - debugging tool. It is compatible only with the Elasticsearch engine.

It is located in the **System -> Search Management -> Validator** section.

To start validation, enter in the field **Search term** - wanted search term, and **Entity ID** which can be found in the Catalog - Products.

This tool helps you find out whether a particular product has been indexed or not. As a result, if the product has been indexed, you will get the output directly from the Elasticsearch index where it finds the matches and highlights all occurrences in the result.

Reports

With detailed search reports, you are able to check how relevant the search is to your customers.

From this information, you will be able to fine-tune your search configuration so that your customers will be led to the products they need.

You can find reports in **System / Search Management / Reports**

You can check the Search Report by:

Search Volume

- Total Searches / Popularity
- Unique Searches - number of unique searches (search phrases)
- Users - number of unique sessions with searches
- Engagement % - the percent of users that opened the product from the search page

You can group or filter it

- By exact date
- By hour
- By day
- By week
- By month
- By year

Search Terms

- Total Searches
- Popularity
- Engagement %

> 📢 Tip
> You can export Reports to CSV, Excel or XML formats

Manage Landing Pages

A landing search page is a special search result page with a static URL where customers are redirected by using some search expression.

Let's consider a large number of frequently requested (or just a promotional set) models of Samsung phones with a black coat. We can create a separate promotional page, say, `http://store.com/black-samsung-phone.html`, and bind it to the search phrase "black samsung phone". Then, when customer request a black Samsung phone, it will be immediately sent to your special page.

Also it supports the following logic: we create a separate promotional page, for example, `http://store.com/black-samsung-phone.html`, and bind it to the search phrase "black samsung phone". When a customer goes to this (specific) URL, the search results for "black samsung phone" will be immediately built on it.

All such pages can be managed from the **System -> Search Management -> Manage Landing Pages** grid.

Adding New Landing Page

- Go to **System / Search Management / Manage Landing Pages** and press **Add New** button.
- On the creation page, fill in the following fieds:
  - **Query Text** - the key phrase, which should bring customer to the landing page (ex. black samsung phone)
  - **URL Key** - relative path to the landing page. For example, if the URL key is `shoes/all`, then the full URL would be `https://example/shoes/all/`.
  - **Active** - activates or deactivated redirect to the landing page.
  - **Page Title** - overrides the title of that page with yours.
  - **Meta Keywords** - meta keywords that can be used by search crawlers.
  - **Meta Description** - meta description that can be used by search crawlers.
  - **Layout Update XML** - overrides XML layout of the landing page.
- Save and activate the landing page.

Manage Synonyms

The extension increases native Magento synonyms' functionality and provides additional ability for importing synonyms.

To import synonyms, perform the following steps:

- Place your custom YAML file into the `[magento_root]/var/sphinx/synonyms/` folder.
- Go to **System -> Search Management -> Manage Synonyms** and press the **Import Synonyms** button.
- The **Dictionary** field defines locale (language) to which synonyms are imported. All dictionaries should exist and have at least one record since imported data are appended to existing.
- **Store View** defines the store views where imported synonyms will be applied.
- Press **Import** to import and apply synonyms.

Manage Stopwords

Stopwords are words that have little lexical meaning or ambiguous meaning and are not useful during a search (ex. and, or, the, a, with, etc). Therefore, these words should be removed from search phrases to make them relevant.

You can either manually add stopwords, or import them from a YAML-formatted file.

Adding New Stopword

- Go to the **System -> Search Management -> Manage Stopwords** grid and press the **Add New Stopword** button.
- On the creation page, fill in the following fields:
  - **Stopword** - is the keyword which should be removed from search requests.
  - **Store View** - allows you to select where defined synonyms will be applied.
- Save record.

Importing Stopwords

Our extension uses the YAML file format for stopwords importing. It should resemble the following format:

```
[ID_1]:[Stopword_1]
[ID_2]:[Stopword_2]
[ID_3]:[Stopword_3]
```

The name of this file should be equal to your language code in capital letters. Codes can be found here, use column **639-1** for that.

> **Example**
>
> Let's create a stopwords file for English locale. The name of such a file would be `EN.yaml`, and its content should be:
>
> ```
> 1:"but"
> 2:"now"
> 3:"what"
> 4:"except"
> ```

To import stopwords from such a file, perform the following steps:

- Place your custom YAML file to the special `[magento_root]/var/sphinx/stopwords/` folder.
- Go to **System -> Search Management -> Manage Stopwords** and press the **Import Stopwords** button.
- **Dictionary** field defines locale (language), for which stopwords are imported. It is picked from the name of your YAML import files.
- **Store View** defines the storeview, where imported stopwords should be applied.
- Press **Import** to import and apply stopwords.

Score Boost Rules

Score Boost Rules are a powerful tool, which allows you to influence the relevancy of search results, depending on certain conditions.

When the Mirasvit Search extension builds search results, it groups them by indexes' position and their position in **System -> Search Management -> Settings -> Search Autocomplete -> Searchable Content**. Groups can include Products, Pages, Categories and so on - they can be defined in **System -> Search Management -> Search Indexes**.

However, inside these groups items are listed strictly by their relevance to search query, which is calculated for each item separately as **item rank**. The position in a search results list is dependent on this rank value.

Score Boost Rules allows you to increase or decrease this rank depending on item properties, which allow you to move certain products to the top or botton of a list, which can be extremely useful for promotion and marketing purposes.

Creating a new Rule

To create a new Score Boost Rule, navigate to **System -> Search Management -> Score Boost Rules** section and press **Add New Rule** button.

You need to define the following properties to create a Rule/

- **Title** - internal title of the Rule
- **Active** - whether the Rule is active and should be applied to Search Results
- **Active (date)** - a time period, when the Rule should apply to Search Results. Leave this field empty to have the Rule always be applicable.
- **Store** - storeviews, where the current Rule should be applicable.
- **Score Factor** - score adjustment, which should be added or subtracted from the rating, generated by a search engine.
  - **Action** - the action that should be performed. It can have only two possible values.
    - **Increase By**
    - **Decrease By**
  - **Rank Adjustment** - the numerical value, which should be added or subtracted from rating.
  - **Metric** - defines, how Rank Adjustment shall be used for adjustment. It can have two possible values:
    - **Points** - in this case, Rank Adjustment is just added to the actual rating.
    - **Times** - in this case, the actual rating is multiplied by Rank Adjustment. It is used to rocket-jump products to the top (for example, promotional products).
  - **Parameter** - defines which rating shall be adjusted by the Rule.
    - **Initial Score** - the rating which was generated by search engine.
    - **Product Popularity** - the popularity rating, which is defined as the quantity of orders of products that meet conditions below.
    - **Product Rating** - product rating, that is defined as quantity of reviews for products, that meet conditions below

The conditions are broken into two parts:

- **Apply the rule only for the following products** - allows you to define which combination of products makes the Rule apply.

> 💬 Note
>
> To add additional conditions please go to **Stores - Attributes - Product**, select a necessary attribute, for example, SKU, open edition in the tab **Storefront Properties**, and set Yes for the **Use for Promo Rule Conditions**, clean Magento cache after saving.

- **Apply the rule only when the following conditions are met** - allows you to filter **Search Query**, to which the Rule shall apply.

Both of these conditions use the same pattern, as with other rules in Magento 2, and are enclosed into logical blocks `If ALL of these conditions are TRUE/FALSE` (the products meet conditions, when all of them apply) or `If ANY of these conditions are TRUE/FALSE` (product shall meet only one of defined conditions).

Here are few useful examples that demonstrate how the Score Boost Rules work.

Examples

Erin Recommends Promo

This example allows you to move products that were recommended by your editorial board (defined here by the custom attribute **Erin Recommends**), to the top of the search results.

Title: `Erin Recommends Promo` **Score Factor: Increase by** `10` **points Initial Score Apply the rule**

only for the following products:

- `Erin Recommends is Yes`

Analog Watches to the End

This rule drops all analog watches to the very bottom when a customer search includes the "watch" keyword.
Title: `Analog Watches to End` **Score Factor**: **Decrease by** `2` **times Initial Score Apply the rule only for the following products:**

- `Product Name contains analog` **Apply the rule only when the following conditions are met:**
- `Search Query contains watch`

New Products Promo

This example allows you to lift promotional products higher on the list than others, but not necessarily at the top.
Title: `New Products Promo` **Score Factor**: **increase by** `5` **points Initial Score Apply the rule only for the following products:**

- `New is Yes`

Translation

Our extension supports multilanguage stores. To translate it, we have used the same logic as default Magento.

All our translation files are in the "/i18n" folder of each sub-module.

i18n files should be located at:

- **vendor/mirasvit/module-name/src/ModuleName/i18n** - if installed via composer;
- **app/code/Mirasvit/ModuleName/i18n** - if installed manually.

Create a separate .csv file of your language for our extension. The names of all languages can be found with this command:
`php -f bin/magento info:language:list`

Override the strings in your dictionary file:

```
"Original line" , "Translated line"
```

To avoid rewriting the file with the updating of the module, replace the translation file to your theme directory at **app/design/frontend/THEME_VENDOR/theme_name/i18n/lg_LG.csv**.
Where **lg_LG.csv** is a translation file in another locale.

To see the changes, run these commands :
`php -f bin/magento setup:static-content:deploy`
`php -f bin/magento cache:flush`

FAQ

This section describes the most common problems that customers report, and how they can be resolved.

How to make the autocomplete dropdown scrollable and smaller for mobile devices

For this, navigate to the **Stores > Settings > Configuration > Mirasvit Extensions > Developer > CSS Settings** and add the css styles below to the **Additional CSS Styles** field:

```
@media screen and (max-width: 767px) {
    .mst-searchautocomplete__autocomplete {
        max-height: 200px;
        overflow-y: scroll;
    }
}
```

`max-width: 767px` - is the maximum width of the device for which these styles are applied.

How to make the autocomplete show a product price including or excluding tax

For this, navigate to the **Stores > Settings > Configuration > Sales > Tax > Price Display Settings** and switch the option **Display Product Prices In Catalog** to **Excluding Tax** - to display the price without taxes or any other option for displaying the price including tax.

Keep getting error onto each search page: The page you requested was not found, but we have searched for relevant content.

This issue is possible if a certain page contains resources (js, css, images) with a 404 error.

To prevent this issue, you can disable a 404 search at **System -> Search Management -> Settings -> Mirasvit Extensions -> Search.**

After enabling Mirasvit_SearchAutocomplete my CSS styles are missed on frontend

Please follow **System -> Search Management -> Settings -> Mirasvit Extensions -> Developer -> CSS Settings tab**, set **YES** at Enable preprocessed CSS, then flush cache and run static content deploy.

Does Mirasvit Search support REST API?

Since Native Magento provides search API, our search extension uses it as well. Please follow the official documentation here.

Singular/Plural search: how it works?

Our search extension supports singular and plural search out of the box for the EN, NL, RU locales. If you need to add support for other languages, use lang analyzer technology to customize your Elastic Search for the appropriate language.\ For the Sphinx, please use Sphinx engine version with Stemmer (with language specific morphology).

Troubleshooting

This section contains the most common problems that customer can encounter, and how they can be resolved.

> 💬 Note
>
> Please, make sure that you're using the latest version of the extension. Otherwise, please update it to the latest version.

Search is not possible by SKU

Take a moment to verify the following conditions are in order:

- SKU attribute is searchable. You can check it in **Stores -> Attributes -> Product grid -> SKU -> Storefront Properties -> Use in Search and Visible in Advanced Search** should be **Yes**.
- SKU is in the list of **Searchable** attributes in **Product Index**. You can check it in **System -> Search Management -> Search Indexes -> Product Index**.
- If SKU includes dashes or other non-alphabetic symbols, set up Long Tail Search expressions as described in our manual.
- Validate the search result. Go to **System -> Validator -> Validate Search Results**. In the **Search term** field, enter your SKU, in **Product ID** - ID of the product without a searchable SKU and press **Validate Search Results**.

After enabling fallback search and entering too many words, search fails

Possible cause: too small a max_execution_time PHP parameter, which is not enough to complete requests with large number of words.

Solution: there are two possible solutions.

1. Increase `max_execution_time` parameter either in `.htaccess`, or directly in `PHP.ini`.
2. Use default Magento settings to adjust search parameters. They are located at **Stores -> Configuration -> Catalog -> Catalog Search** and consist of two options:
   - **Minimal Query Length** - defines minimal quantity of words in search request (1 for default).
   - **Maximum Query Length** - defines maximal quantity of words in search request (128 by default).

Typically, it is enough to decrease the latter parameter until the search works correctly.

Autocomplete (and/or Search Results) is too slow

Possible Causes and **Solutions**

<details>
<summary>Search Engine Settings</summary>

How to Check:

- Navigate to **System -> Search Management -> Settings -> Search Engine Configuration**. If the option **Search Engine** is set to **Built-In Engine** or **MYSQL**, this is the probably cause.

How to Resolve:

- If you have Mirasvit MYSQL Search Module version below 1.0.23, upgrade it to latest version, as it contains significant improvements that speed up the built-in search engine.
- Enable the option **Fast Mode** in **Autocomplete settings** at **System -> Settings -> Mirasvit Extensions -> Search Autocomplete** section.
- If this **will not** help, consider a recommendation from next option.
</details>

<details>
<summary>Large Quantity of Products</summary>

How to Check:
</details>

- Navigate to **Catalog -> Products** section. If you have more than 14k records there, this is the probably cause.

How to Resolve:

- Replace **Build-In Engine** to **External Sphinx Engine** in option **Search Engine** at **System -> Search Management -> Settings -> Search Engine Configuration**.
  Search Sphinx shall be installed first, and then connected to your store.

High-load Crontasks

How to Check:

- Install **Cron Scheduler for M2**.
- Navigate to **System -> Cron Scheduler by KiwiCommerce -> All cron jobs** section and **System -> Cron Scheduler by KiwiCommerce -> Cron job schedule timeline**. Check there execution time and results of crontasks. If some of them are stuck or executed for too long period, this is the probably cause.

How to Resolve:

- Disable or reconfigure all crontasks, which are stuck or taking too much time.

Aheadworks blog search doesn`t return results

How to Resolve:

- Find and open the following file : Aheadworks/Blog/Block/Post.php
- Find `public function getSocialIconsHtml()`
- After condition you will see this row `$block = $this->getLayout()->createBlock`, place this code before
  `$socialIconsBlock = !empty($this->getSocialIconsBlock())?$this->getSocialIconsBlock():'Ahe`
- Replace `$this->getSocialIconsBlock()` after `createBlock` with `$socialIconsBlock`
- You should get your code to look like
  `$socialIconsBlock = !empty($this->getSocialIconsBlock())?$this->getSocialIconsBlock():'Ahe`

"unknown column" error during Sphinx reindex

If your Sphinx Search Engine installed on same server run the following steps:

- Click "Reset" in Search Engine Configuration (backend)
- Click "Restart Sphinx Daemon" in Search Engine Configuration (backend)
- Reindex Search indexes by running bin/magento indexer:reindex catalogsearch_fulltext (CLI) or in System / Search Management / Search Indexes (Backend)

If your Sphinx Search Engine installed on a remote server run the following steps:

- Click "Generate configuration file"
- Copy generated file to your remote server
- Run killall -9 searchd on your remote server
- Start sphinx daemon using the command searchd –config <path to config/sphinx.conf>
- Reindex Search indexes by running bin/magento indexer:reindex catalogsearch_fulltext (CLI) or in System / Search Management / Search Indexes (Backend)

Command line commands

Our Search extension also has a command-line interface which can be used from a console or SSH.

Here is the list of available commands. (for `bin/magento`):

- `mirasvit:search-sphinx:manage` - sphinx engine management
- `mirasvit:search:reindex` - reindex all search indexes. *This command is an alias for default command* `indexer:reindex catalogsearch_fulltext`
- `mirasvit:search:stopword` - manage stopwords
- `mirasvit:search:synonym` - manage synonyms

Managing Search Sphinx from console

Console management of Search Sphinx includes the following commands:

- `start` - starts Search Daemon
- `stop` - stops Search Daemon
- `restart` - kills Search Daemon process and starts it with clean data
- `reset` - cleans search temporary data
- `status` - displays current status of Search Sphinx engine.
- `ensure` - special command, which automatically checks status, and only if daemon does not work, will it start up.

To execute management command, run the following expression from console/SSH:

```
bin/magento mirasvit:search-sphinx:manage --[command]
```

Running Reindex from console

Console reindexing can be run either for a whole store (without mode options), or for a selected Index (see more at Managing Indexes), and for a selected Store.

```
mirasvit:search:reindex --[mode]=[argument]
```

Possible modes are:

- `INDEX` - allows for reindexing a particular index. `[argument]` value should be code if desired index.
- `STORE` - allows for reindexing all indices at particular store. `[argument]` value should be code if desired store. Store codes can be seen in **Stores -> All Stores** in Magento 2 backend.

> **Tip**
>
> Possible codes of indices you can get directly at console by executing command
>
> `mirasvit:search:reindex --index=default`
>
> This command will display all indexes with codes in square brackets.

Modes can be used simultaneously, e.g. if you need to reindex the Product index (code is *'catalogsearch_fullindex'*) on store with code *'german'*, you need the following command:

```
mirasvit:search:reindex --index=catalogsearch_fullindex --store=german
```

Managing Stopwords from console

Console stopword managing allows you not only to import, but also to remove stopwords. For that, you will need the following command:

```
mirasvit:search:stopword [arguments]
```

Possible arguments are:

- `--file` - defines, which YAML file will be used for stopwords importing.
- `--remove` - optional argument, that commands to remove stopwords instead of importing. Requires previous argument.
- `--store=[store_code]` - **required** argument, that forces import/remove action performing only on specific store. Store codes can be seen in **Stores -> All Stores** in Magento 2 backend.

Format of YAML file, which is used for managing stopwords, can be seen here.

Managing Synonyms from console

Console synonym managing allows you not only to import, but also to remove them. For that you will need the following command:

```
mirasvit:search:synonym [arguments]
```

Possible arguments are:

- `--file` - defines which YAML file will be used for synonyms importing.
- `--remove` - optional argument, that commands removing of synonyms instead of importing. Requires a previous argument.
- `--store=[store_code]` - **required** argument that forces import/remove action performing only on a specific store. Store codes can be seen in **Stores -> All Stores** in Magento 2 backend.

The format of YAML file, which is used for managing synonyms, can be seen here.

Extension Upgrading

To upgrade the extension follow these steps:

1. Back up your store's database and web directory.
2. Log in to the SSH console of your server and navigate to the root directory of the Magento 2 store.

   If extension was installed via:

   - **Composer**: run command to update the current extension with all dependencies.

     ```
     composer require mirasvit/module-search-ultimate:* --update-with-dependencies
     ```

     > 💬 Note
     >
     > In some cases, the command above is not applicable, as it's not possible to update just the current module, or you just need to upgrade all Mirasvit modules in a bundle. In this case, the command above will not be effective.
     >
     > Run instead `composer update mirasvit/*` command. It will update all Mirasvit modules, installed in your store.

   - **Direct file upload**: download new extension package from our store and copy contents to root Magento directory
3. Run command to install updates:

   ```
   php -f bin/magento setup:upgrade
   ```

4. Deploy static view files:

```
rm -rf pub/static/*
rm -rf var/view_preprocessed/*
php -f bin/magento setup:static-content:deploy
```

5. Clean the cache:

```
php -f bin/magento cache:clean
```

6. Rebuild the search index:

```
php -f bin/magento indexer:reindex catalogsearch_fulltext
```

Extension Disabling

Temporarily Disable

To temporarily disable the extension, please follow these steps:

1. Log in to the SSH console of your server and navigate to the root directory of the Magento 2 store.
2. Run command to disable extension:

```
php -f bin/magento module:disable Mirasvit_Search Mirasvit_SearchMysql
Mirasvit_SearchSphinx Mirasvit_SearchAutocomplete Mirasvit_Misspell
Mirasvit_SearchLanding Mirasvit_SearchReport Mirasvit_SearchElastic
```

Remove the Extension

To uninstall the extension, please follow these steps:

1. Log in to the SSH console of your server and navigate to the root directory of the Magento 2 store.
2. Disable extension.
3. Run command to remove the extension:

```
composer remove mirasvit/module-search-ultimate
```