Highlighting Results

Overview

This is what we mean by "highlighting results." We searched for "nation." AIE highlighted "nation," "national," and "nationalities."

8. Kazakhstan

Asia

...single nation. the area was conquered by Russia in the 18th century, and Kazakhstan became a Soviet Republic... deported nationalities) skewed the ethnic mixture and enabled non-Kazakhs to outnumber natives. Independence... cohesive national identity; expanding the development of the country's vast energy resources and exporting...

If a field has **highlight.enabled** set to true in the AIE schema, and if the QueryRequest has highlighting turned on, then the matching terms and phrases will be highlighted in the search results.

It is also possible to attach a Teaser or ScopeTeaser field expression to a query, which can force highlighting on a field regardless of schema field settings and QueryRequest settings.

View incoming links. **>>**

Attivio Glossary , Field Properties , Query and Response - Concepts and Tools , Query Data and Content in Attivio , Simple Query Language

Index-based vs. Document-based Highlighting

AIE offers index-based highlighting, in which the data is tokenized during ingestion, and document-based highlighting, in which tokenization occurs during a query.

Index-based highlighting can be configured in the Schema for tokenized string or text fields that are indexed, such as the **title** and **text** fields. (Highlighting is not supported for untokenized datatypes such as dates.)

To enable index-based highlighting for a field, you must configure the schema field in one of two ways:

1. Set the **highlight.enabled** field property to "true", or
2. Set **highlight.enabled** to "false", but explicitly set **highlight.method** to "offsets".

If a field has **highlight.enabled** set in the schema, then AIE will always highlight the field in every query. If **highlight.enabled** is "false", but **highlight.method** is set to "offsets", then index-based highlighting of this field can be enabled on demand by using the **Teaser** field expression in a query.

To enable document-based highlighting, set **highlight.method** to "document". This streamlines the ingestion process and slims the index, but slows down querying when highlighting is requested.

The Teaser field expression can be used on any field expression, regardless of the **highlight.method** used during ingestion. If the field has been prepared for highlighting (**highlight.method** is "offsets") the Teaser uses index-based highlighting. If the underlying expression has not been prepared for highlighting (**highlight.method** "document"), Teaser will tokenize the value at query time and will try its best to apply highlighting to the result. You can apply this approach to non-string fields/expressions with varying success.

View incoming links. **>>**

Attivio Glossary , Field Properties , Query and Response - Concepts and Tools , Query Data and Content in Attivio , Simple Query Language

Schema Field Properties

See Field Properties for the list of AIE Schema field properties that influence result highlighting.

Example Highlighted Field

This is an example of configuring the **title** field to support result highlighting:

```
<schema name="default">
  <fields default-search-field="content">
    <field name="title" type="string" indexed="true" stored="true" sort="true" facet="false">
      <properties>
        <property name="highlight.enabled" value="true" />
        <property name="highlight.fragment" value="false" />

        <!-- specify fields that will have query terms extracted from query for highlighting this field -->
        <property name="highlight.whitelist" value="content,title,*_s"/>

        <!-- ... Rest of properties -->
      </properties>
    </field>
  </fields>
</schema>
```

Note that setting **highlight.enabled** to "true" means that **highlight.method** will default to "offsets", and therefore does not need to be specified in the example.

Highlighting Methods

There are two primary methods for performing highlighting of documents at query time: index-based highlighting (with two sub-types), and document-based highlighting. Each method has its performance and disk space characteristics. Different methods will also produce different highlighted results.

Index-Based Highlighting

⚠ Note that index-based highlighting requires that the schema field have the **indexed** attribute set to "true".

Index-based highlighting prepares data for highlighting during ingestion, in order to accelerate highlighting performance in queries. This approach requires extra disk space to store special data structures (known as **term vectors**) for each document. This method also allows for better highlighting of phrases, as well as proper highlighting of stems, entities, synonyms, and other inflected forms generated at index time.

The required disk space/performance can be tuned by storing the term vector data in one of two different ways.

Option 1. highlight.method = "index"

Setting the **highlight.method** to **index** requires AIE to build term vectors for a field and store them in the index, enabling very fast result highlighting during a query. This requires the most disk space of the three highlighting options, but it also produces the fastest queries.

Option 2. highlight.method = "offsets"

Setting the **highlight.method** to "offsets" requires AIE to build term vectors for a field and store them in the index, just like the **index** option, but in this case the term vectors are compressed to take up less disk space in the index. Query-time highlighting performance will be slower as a result, especially for wildcard queries.

Query performance of this method can be improved by setting the **index.termVector** field property to "true" for the field. This will require more disk space (but less than is required for the **index** method), and it will accelerate the performance of highlighting wildcard/expansion queries.

Document-Based Highlighting

Document-based highlighting does not require indexing the field, because it tokenizes stored fields at query time instead of during ingestion. As a result, highlighting large documents at query time (especially using slow tokenizers) can be very CPU-intensive.

This method can be enabled by setting the **highlight.method** property of the field to "document".

Enabling Highlighting At Query Time

Highlighting at query time is enabled by calling QueryRequest.setHighlight(true) . This will result in all requested fields with **highlight.enabled** set to "true" in the schema being highlighted on every query.

```
SearchClient client = ...; // Initialize the client

// Enable Highlighting
QueryRequest query = new QueryRequest("foo");
query.setHighlight(true);

QueryResponse response = client.search(query);

// "foo" will be highlighted in configured fields
for (SearchDocument doc : response.getDocuments()) {
  System.err.println(doc);
}
```

Teaser and ScopeTeaser Field Expressions

The **Teaser** and **ScopeTeaser** Field Expressions let us request highlighting on individual fields at query time. They use the form of highlighting that is appropriate to the field. This makes it possible to highlight fields that are not configured for highlighting in the schema. It also lets us use different highlighting settings for fields that were configured for highlighting in the schema.

Using the Teaser or ScopeTeaser Field Expressions does not require turning on highlighting for the QueryRequest.

The following example is for **Teaser**. The **ScopeTeaser** field expression extends Teaser by tying the size of the result snippet to the position of scope tags in the text. See the Field Expressions page for more information.

```
SearchClient client = ...; // Initialize the client

// Search for all documents, but highlight "foo" in the returned fields
QueryRequest query = new QueryRequest("foo");

// Create the field expression for highlighting "random_s"
Teaser teaser = new Teaser("random_s", "random_s");
teaser.setFragment(true);
teaser.setNumFragments(1);

// Request all fields, and highlight "random_s"
query.addField("*");
query.addFieldExpression(teaser);

QueryResponse response = client.search(query);

// "foo" will be highlighted in configured fields
for (ResponseDocument doc : response.getDocuments()) {
  System.err.println(doc);
}
```

Highlight Scope and Mode

In search results, the highlighted text is set off by scope tags. The text of the scope tag (default "highlight") is set by the **QueryRequest. setHighlightScope()** method.

The tag be set by the following HTML REST parameter:

```
&highlight.scope=highlight
```

or in java code:

```
QueryRequest r;
r.setHighlightScope("highlight");
```

The tag can be returned in one of two modes, either as an XML tag or as an HTML tag.

```
XML: <highlight>term</highlight>

HTML: <span class="highlight">term</span>
```

These modes are set using the **QueryRequest.setHighlightMode()** method. The HTML REST examples are:

```
&highlight.mode=xml

&highlight.mode=html
```

In Java it looks like this:

```
QueryRequest r;
r.setHighlightMode( HighlightMode.XML );

QueryRequest r;
r.setHighlightMode( HighlightMode.HTML );
```

Highlighting Large Documents

Very large stored fields can have impacts on highlighting performance. In general, it is not recommended to index very large stored fields as this has impacts on disk usage, as well as memory usage during query retrieval.

If large stored fields will in general be highlighted, the following recommendations are suggested in order to reduce the CPU and memory burden.

Example Schema Configuration

This schema configuration has two text fields: **text** and **text.full**. **text** will be returned by default and will be highlighted if highlighting is enabled for the query. **text.full** must be explicitly requested. It is recommended that **text.full** is only requested for one document at a time. See query examples below.

```
<schema name="default">
  <fields default-search-field="content">
    <!-- The truncated text (used for highlighting) -->
    <field name="text" type="string" indexed="true" stored="true" sort="false" facet="false">
      <properties>
        <!-- Truncate the text field to only what will be highlighted -->
        <property name="store.maxChars" value="10240"/>

        <!-- Highlighting settings -->
        <property name="highlight.enabled" value="true"/>
        <!-- ... rest of highlighting settings for this field -->
      </properties>
    </field>

    <!-- The full, untruncated text. -->
    <field name="text.full" type="string" indexed="true" stored="true" sort="false" facet="false">
      <properties>
        <!-- Do not return this field when "*" fields are requested.
             This prevents this field from being cached in memory during regular searches. -->
        <property name="store.hidden" value="true"/>
      </properties>
    </field>
  </fields>
</schema>
```


This configuration requires that you copy the value of the **text** field to the **text.full** field in the ingestion workflow.

Query Example

Example that highlights the text field:

```
SearchClient client = ...; // Initialize the client

QueryRequest query = new QueryRequest("foo");
query.setHighlight(true);

QueryResponse response = client.search(query);

// Display the results ("text" field will be highlighted)
// NOTE: text.full will not be returned with results due to "store.hidden" property
for (ResponseDocument doc : response.getDocuments()) {
  System.out.println(doc);
}
```

Example that requests the "full text" for a specified document:

```
SearchClient client = ...; // Initialize the client

QueryRequest query = new QueryRequest(".id:documentid");

// Make sure the large stored field will not be cached in memory
query.setCacheable(false);

// Request the full text field
query.addField("text.full");

// ... (Request any other needed fields)

QueryResponse response = client.search(query);

if (response.getDocuments().size() > 0) {
  System.out.println(response.getDocuments().get(0));
}
```