AWS Cloudsearch

Xu Zhi Shu (xuzhizs2)

Overview

Having a search engine for a particular dataset is a feature relevant to many websites. Website owners have many options when choosing how to build and maintain the search engine - whether to do it all by themselves or to offload the task somewhere else. AWS Cloudsearch is a fully managed solution that provides customizable indexing and automatic scaling.

Configuring AWS Cloudsearch

Data format

AWS cloudsearch requires structured data to work, so the system must first be configured with what the data is expected to look like. Each item that is intended to show up in search results should be in the format of XML or JSON, with a unique ID, and one or more other fields. The user needs to manually map each field name to an AWS supported field type, or setup "dynamic field matching" to automatically detect the field type.

The following are basic field types supported by AWS Cloudsearch:

- `date`—contains a timestamp. Dates and times are specified in UTC (Coordinated Universal Time) according to IETF RFC3339: `yyyy-mm-ddTHH:mm:ss.SSSZ`. In UTC, for example, 5:00 PM August 23, 1970 is: `1970-08-23T17:00:00Z`. Note that you can also specify fractional seconds when specifying times in UTC. For example, `1967-01-31T23:20:50.650Z.`
- `date-array`—a date field that can contain multiple values.
- `double`—contains a double-precision 64-bit floating point value.
- `double-array`—a double field that can contain multiple values.
- `int`—contains a 64-bit signed integer value.
- `int-array`—an integer field that can contain multiple values.
- `latlon`—contains a location stored as a latitude and longitude value pair (`lat, lon`).

- `literal`—contains an identifier or other data that you want to be able to match exactly. Literal fields are case-sensitive.
- `literal-array`—a literal field that can contain multiple values.
- `text`—contains arbitrary alphanumeric data.
- `text-array`—a text field that can contain multiple values.

Additionally, each field could also be configured with the following properties:

- `HighlightEnabled`—You can get highlighting information for the search hits in any `HighlightEnabled` text field. Valid for: `text`, `text-array`.
- `FacetEnabled`—You can get facet information for any `FacetEnabled` field. Text fields cannot be used for faceting. Valid for: `int`, `int-array`, `date`, `date-array`, `double`, `double-array`, `latlon`, `literal`, `literal-array`.
- `ReturnEnabled`—You can retrieve the value of any `ReturnEnabled` field with your search results. Note that this increases the size of your index, which can increase the cost of running your domain. When possible, it's best to retrieve large amounts of data from an external source, rather than embedding it in your index. Since it can take some time to apply document updates across the domain, critical data such as pricing information should be retrieved from an external source using the returned document IDs. Valid for: `int`, `int-array`, `date`, `date-array`, `double`, `double-array`, `latlon`, `literal`, `literal-array`, `text`, `text-array`.
- `SearchEnabled`—You can search the contents of any `SearchEnabled` field. Text fields are always searchable. Valid for: `int`, `int-array`, `date`, `date-array`, `double`, `double-array`, `latlon`, `literal`, `literal-array`, `text`, `text-array`.
- `SortEnabled`—You can sort the search results alphabetically or numerically using any `SortEnabled` field. Array-type fields cannot be `SortEnabled`. Only

sort enabled numeric fields can be used in expressions. Valid for: `int`, `date`, `latlon`, `double`, `literal`, `text`.

Dynamic field matching

AWS Cloudsearch could be configured to automatically assign field types to fields in documents through regular expression style matching. For example the user could set a rule that matches "*_i" to integer type field, and "*_t" to a text type field, and then set the field names in the document according to the same rules.

Indexing options

Stemming

The user may configure Cloudsearch to combine multiple words similar in meaning. (ie. 'sleep' is the same as 'slept') There are several options:

- none - no stemming
- minimal - only perform basic stemming by removing plurals
- light - also performs stemming on common non/adjective combinations, and derived suffixes
- full - aggressively performs stemming

Additionally, the user could add a "stemming dictionary" that maps groups of words to stems. The "stemming dictionary" is applied after algorithmic stemming.

Stop words

The user may specify a "stop word dictionary". Words in the stop word dictionary will be ignored when indexing text and text-array fields. If the user does not specify a "stop word dictionary", AWS will use a default list of stop words.

Synonyms

The user could define a dictionary of synonyms. During indexing of a document, all the synonyms of a particular word will be added to the index. AWS does not have a synonym list by default because having many synonyms could drastically increase the index size, the number of matches, and the query time.

Searching for results

Search API

AWS Cloudsearch provides various APIs to let the user search by compound queries, or by specific field types.

Ranking documents relevance

This part is completely managed by AWS. The user does not get to change the ranking function.

Conclusion

Once the user obtains correctly formatted data, possibly from database output or a web crawler, AWS Cloudsearch seems relatively straight forward to setup. However, the major drawback of this solution is that the user does not have any visibility to the ranking function, nor the ability to to customize their own ranking function. Additionally, all the data that needs to be searched must be uploaded to AWS for indexing, which creates security and privacy concerns.

References

https://docs.aws.amazon.com/cloudsearch/latest/developerguide