AWS Kendra

Friday, November 22, 2024

9:31 AM

Amazon Kendra is an intelligent search service powered by machine learning. It enables organizations to build powerful and accurate search capabilities into their applications, allowing users to search unstructured and structured data using natural language queries.

Key features of Amazon Kendra include:

- \*\*Natural Language Understanding\*\*: Users can search using natural language questions, and Kendra understands the context and intent to provide accurate answers.

- \*\*Connectors\*\*: Kendra provides connectors to various data sources such as file systems, databases, and applications like SharePoint, Salesforce, and more.

- \*\*Relevance Tuning\*\*: Kendra allows you to fine-tune the relevance of search results based on specific business needs.

- \*\*Faceted Search\*\*: Users can refine search results using facets like document type, author, date, and more.

- \*\*Security\*\*: Kendra integrates with AWS Identity and Access Management (IAM) and other security mechanisms to ensure that search results respect data access controls.

Amazon Kendra is useful for a wide range of applications, including enterprise search, customer support, knowledge management, and more. It helps improve productivity by making it easier for users to find the information they need quickly and accurately.

This code demonstrates how to perform the following operations:

* 1. NLP and Querying Kendra: Basic querying and retrieving results.
  2. Faceted Search: Using facets in your Kendra query.
  3. Relevance Tuning: Adjusting how results are scored.
  4. Connectors: Creating a data source connector (e.g., S3 bucket).
  5. Security Configuration: Setting up access control

import com.amazonaws.auth.profile.ProfileCredentialsProvider;

import com.amazonaws.services.kendra.AmazonKendra;

import com.amazonaws.services.kendra.AmazonKendraClientBuilder;

import com.amazonaws.services.kendra.model.\*;

import java.util.\*;

public class KendraExample {

private static final String KENDRA\_INDEX\_ID = "your-kendra-index-id";

private static final String S3\_BUCKET\_NAME = "your-source-bucket-name";

private static final String ROLE\_ARN = "your-role-arn";

public static void main(String[] args) {

AmazonKendra kendraClient = AmazonKendraClientBuilder.standard()

.withCredentials(new ProfileCredentialsProvider())

.build();

String queryText = "What is Amazon Kendra?";

// Perform a basic query

queryKendra(kendraClient, queryText);

// Perform a faceted search query

queryKendraWithFacets(kendraClient, queryText);

// Relevance tuning by adjusting result scores

tuneRelevance(kendraClient, KENDRA\_INDEX\_ID);

// Create a data source connector, e.g., S3 bucket

createDataSource(kendraClient, KENDRA\_INDEX\_ID, S3\_BUCKET\_NAME, ROLE\_ARN);

// Set up security access control

configureIndexPermissions(kendraClient, KENDRA\_INDEX\_ID);

}

private static void queryKendra(AmazonKendra kendraClient, String queryText) {

QueryRequest queryRequest = new QueryRequest()

.withIndexId(KENDRA\_INDEX\_ID)

.withQueryText(queryText);

QueryResult queryResult = kendraClient.query(queryRequest);

System.out.println("Query results:");

queryResult.getResultItems().forEach(item -> {

System.out.println("Title: " + item.getDocumentTitle().getText());

System.out.println("Excerpt: " + item.getDocumentExcerpt().getText());

});

}

private static void queryKendraWithFacets(AmazonKendra kendraClient, String queryText) {

QueryRequest queryRequest = new QueryRequest()

.withIndexId(KENDRA\_INDEX\_ID)

.withQueryText(queryText)

.withFacetFields(Arrays.asList("DocumentType"));

QueryResult queryResult = kendraClient.query(queryRequest);

System.out.println("Faceted Query results:");

queryResult.getFacetResults().forEach(facetResult -> {

System.out.println("Facet key: " + facetResult.getFacetKey());

facetResult.getDocumentAttributeValueCountPairs().forEach(pair -> {

System.out.println("Value: " + pair.getDocumentAttributeValue().getStringValue() + ", Count: " + pair.getCount());

});

});

}

private static void tuneRelevance(AmazonKendra kendraClient, String indexId) {

RelevanceLevel relevanceLevel = new RelevanceLevel()

.withName(RelevanceLevelName.IMPORTANT)

.withSensitivity(RelevanceLevelSensitivity.HIGH);

DocumentRelevanceConfiguration relevanceConfiguration = new DocumentRelevanceConfiguration()

.withRelevanceLevel(relevanceLevel);

UpdateRelevanceLevelRequest updateRequest = new UpdateRelevanceLevelRequest()

.withIndexId(indexId)

.withRelevanceConfiguration(relevanceConfiguration);

kendraClient.updateRelevanceLevel(updateRequest);

System.out.println("Relevance tuning adjusted successfully");

}

private static void createDataSource(AmazonKendra kendraClient, String indexId, String bucketName, String roleArn) {

S3DataSourceConfiguration s3DataConfig = new S3DataSourceConfiguration()

.withBucketName(bucketName);

DataSourceConfiguration dataSourceConfig = new DataSourceConfiguration()

.withS3Configuration(s3DataConfig);

CreateDataSourceRequest createDataSourceRequest = new CreateDataSourceRequest()

.withName("S3DataSource")

.withIndexId(indexId)

.withType(DataSourceType.S3)

.withConfiguration(dataSourceConfig)

.withRoleArn(roleArn);

CreateDataSourceResult createDataSourceResult = kendraClient.createDataSource(createDataSourceRequest);

System.out.println("Data Source created successfully: " + createDataSourceResult.getId());

}

private static void configureIndexPermissions(AmazonKendra kendraClient, String indexId) {

// Specify roles/groups who can access the Kendra index

Principal principal = new Principal()

.withName("example-user")

.withType(PrincipalType.USER);

AccessControlConfiguration accessControlConfig = new AccessControlConfiguration()

.withName("example-access-config")

.withType(PrincipalType.USER)

.withRolePrincipal(principal);

UpdateIndexRequest updateIndexRequest = new UpdateIndexRequest()

.withId(indexId)

.withAccessControlConfigurations(Arrays.asList(accessControlConfig));

UpdateIndexResult updateIndexResult = kendraClient.updateIndex(updateIndexRequest);

System.out.println("Index security configured successfully: " + updateIndexResult);

}

}

* 1. NLP and Querying Kendra: Basic querying and retrieving results.
  2. Faceted Search: Using facets in your Kendra query.
  3. Relevance Tuning: Adjusting how results are scored.
  4. Connectors: Creating a data source connector (e.g., S3 bucket).
  5. Security Configuration: Setting up access control.

java

1import com.amazonaws.auth.profile.ProfileCredentialsProvider;  
2import com.amazonaws.services.kendra.AmazonKendra;  
3import com.amazonaws.services.kendra.AmazonKendraClientBuilder;  
4import com.amazonaws.services.kendra.model.\*;  
5import java.util.\*;  
6  
7public class KendraExample {  
8  
9 private static final String KENDRA\_INDEX\_ID = "your-kendra-index-id";  
10 private static final String S3\_BUCKET\_NAME = "your-source-bucket-name";  
11 private static final String ROLE\_ARN = "your-role-arn";  
12  
13 public static void main(String[] args) {  
14 AmazonKendra kendraClient = AmazonKendraClientBuilder.standard()  
15 .withCredentials(new ProfileCredentialsProvider())  
16 .build();  
17   
18 String queryText = "What is Amazon Kendra?";  
19  
20 // Perform a basic query  
21 queryKendra(kendraClient, queryText);  
22  
23 // Perform a faceted search query  
24 queryKendraWithFacets(kendraClient, queryText);  
25  
26 // Relevance tuning by adjusting result scores  
27 tuneRelevance(kendraClient, KENDRA\_INDEX\_ID);  
28  
29 // Create a data source connector, e.g., S3 bucket  
30 createDataSource(kendraClient, KENDRA\_INDEX\_ID, S3\_BUCKET\_NAME, ROLE\_ARN);  
31  
32 // Set up security access control  
33 configureIndexPermissions(kendraClient, KENDRA\_INDEX\_ID);  
34 }  
35  
36 private static void queryKendra(AmazonKendra kendraClient, String queryText) {  
37 QueryRequest queryRequest = new QueryRequest()  
38 .withIndexId(KENDRA\_INDEX\_ID)  
39 .withQueryText(queryText);  
40  
41 QueryResult queryResult = kendraClient.query(queryRequest);  
42  
43 System.out.println("Query results:");  
44 queryResult.getResultItems().forEach(item -> {  
45 System.out.println("Title: " + item.getDocumentTitle().getText());  
46 System.out.println("Excerpt: " + item.getDocumentExcerpt().getText());  
47 });  
48 }  
49  
50 private static void queryKendraWithFacets(AmazonKendra kendraClient, String queryText) {  
51 QueryRequest queryRequest = new QueryRequest()  
52 .withIndexId(KENDRA\_INDEX\_ID)  
53 .withQueryText(queryText)  
54 .withFacetFields(Arrays.asList("DocumentType"));  
55  
56 QueryResult queryResult = kendraClient.query(queryRequest);  
57  
58 System.out.println("Faceted Query results:");  
59 queryResult.getFacetResults().forEach(facetResult -> {  
60 System.out.println("Facet key: " + facetResult.getFacetKey());  
61 facetResult.getDocumentAttributeValueCountPairs().forEach(pair -> {  
62 System.out.println("Value: " + pair.getDocumentAttributeValue().getStringValue() + ", Count: " + pair.getCount());  
63 });  
64 });  
65 }  
66  
67 private static void tuneRelevance(AmazonKendra kendraClient, String indexId) {  
68 RelevanceLevel relevanceLevel = new RelevanceLevel()  
69 .withName(RelevanceLevelName.IMPORTANT)  
70 .withSensitivity(RelevanceLevelSensitivity.HIGH);  
71  
72 DocumentRelevanceConfiguration relevanceConfiguration = new DocumentRelevanceConfiguration()  
73 .withRelevanceLevel(relevanceLevel);  
74  
75 UpdateRelevanceLevelRequest updateRequest = new UpdateRelevanceLevelRequest()  
76 .withIndexId(indexId)  
77 .withRelevanceConfiguration(relevanceConfiguration);  
78   
79 kendraClient.updateRelevanceLevel(updateRequest);  
80  
81 System.out.println("Relevance tuning adjusted successfully");  
82 }  
83  
84 private static void createDataSource(AmazonKendra kendraClient, String indexId, String bucketName, String roleArn) {  
85 S3DataSourceConfiguration s3DataConfig = new S3DataSourceConfiguration()  
86 .withBucketName(bucketName);  
87  
88 DataSourceConfiguration dataSourceConfig = new DataSourceConfiguration()  
89 .withS3Configuration(s3DataConfig);  
90  
91 CreateDataSourceRequest createDataSourceRequest = new CreateDataSourceRequest()  
92 .withName("S3DataSource")  
93 .withIndexId(indexId)  
94 .withType(DataSourceType.S3)  
95 .withConfiguration(dataSourceConfig)  
96 .withRoleArn(roleArn);  
97   
98 CreateDataSourceResult createDataSourceResult = kendraClient.createDataSource(createDataSourceRequest);  
99  
100 System.out.println("Data Source created successfully: " + createDataSourceResult.getId());  
101 }  
102  
103 private static void configureIndexPermissions(AmazonKendra kendraClient, String indexId) {  
104 // Specify roles/groups who can access the Kendra index  
105 Principal principal = new Principal()  
106 .withName("example-user")  
107 .withType(PrincipalType.USER);  
108  
109 AccessControlConfiguration accessControlConfig = new AccessControlConfiguration()  
110 .withName("example-access-config")  
111 .withType(PrincipalType.USER)  
112 .withRolePrincipal(principal);  
113  
114 UpdateIndexRequest updateIndexRequest = new UpdateIndexRequest()  
115 .withId(indexId)  
116 .withAccessControlConfigurations(Arrays.asList(accessControlConfig));  
117  
118 UpdateIndexResult updateIndexResult = kendraClient.updateIndex(updateIndexRequest);  
119  
120 System.out.println("Index security configured successfully: " + updateIndexResult);  
121 }  
122}

Notes:

* 1. NLP and Querying Kendra:
     + The queryKendra method demonstrates using Kendra's NLP capabilities to perform a search query.
  2. Faceted Search:
     + The queryKendraWithFacets method demonstrates how to perform faceted search, which helps narrow down search results using specific facets like document types.
  3. Relevance Tuning:
     + The tuneRelevance method demonstrates adjusting the relevance score of search results for improved search experience.
  4. Connectors:
     + The createDataSource method demonstrates creating a data source connector (e.g., for an S3 bucket).
  5. Security Configuration:
     + The configureIndexPermissions method demonstrates configuring security access control for the Kendra index.

Prerequisites:

* 1. Ensure you replace placeholders like your-kendra-index-id, your-source-bucket-name, your-role-arn, etc., with actual values.
  2. Make sure you have the necessary IAM roles and policies set up to allow your application to access Kendra and other AWS resources.
  3. Add proper error handling and logging as needed for production use.

This code demonstrates essential operations with Amazon Kendra including NLP, faceted search, relevance tuning, connectors, and security configuration. Modify and expand as needed based on your specific requirements and AWS environment.