4. Add an extractor name and optionally a description.

   This name is used as the model name and as the name of the enrichment that is created when you publish the model. The name is displayed as the enrichment name in the Enrichments page where you and others can apply it to collections. It also is displayed as the model name in the JSON representation of documents where custom entities are found. The name is stored with the capitalization and spacing that you specify.

5. Choose a collection with documents that are representative of your domain data.

6. Choose fields from the document to show in the document view where you will label documents from the collection.

   - **Document title** is shown in the page header as the document name. Choose a field that has a unique value per document, such as the file name, which is stored in the `extracted_metadata.filename` field.
   - **Document body** is where you label entity examples. Choose a field that contains the bulk of the document content, such as the `text` field.

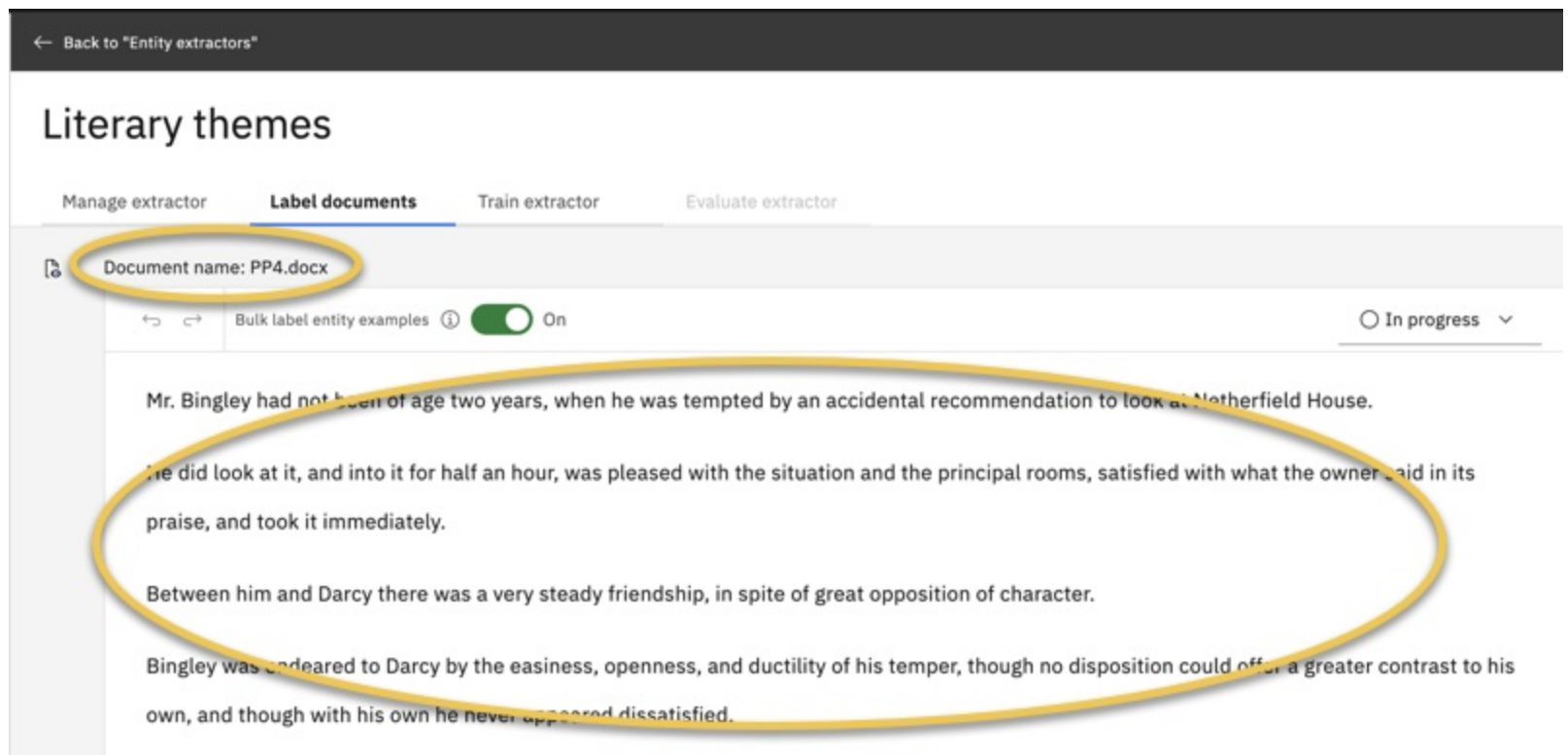


Figure 2. Label documents page

7. Click **Create**.

A document from the collection that you selected is displayed in the *Label documents* view. You will label occurrences of the entity types that you want Discovery to recognize from this and other documents in the collection.

> **Tip:** If no text is displayed in the body of the page, start over now by creating a new entity extractor. This time, when you select a value for the *Document body* field, be sure to choose a field from your processed documents that contains text.

Defining entity types

Define entity types by completing the following steps:

1. Click **Add an entity type**.

2. Add the entity type name and an optional description.

   Use a naming convention that works for your data. The built-in Entities enrichment uses initial capitals and no spaces, for example `EmailAddress`. To distinguish your entities from entities that are extracted by other enrichments, you might want to use a different convention.

3. Optional: Pick the color to use for highlighting text in the document that you want to label as an example of this entity type.

   You can click a color from the *Label color* palette, click the *Renew color* icon to tab from one color to the next. To use a custom color, specify its hexadecimal color code (#fff0f7).

4. Click **Create**.

5. Repeat this process to add all of the entity types that you want the extractor to recognize.

   If you aren't sure what to add for entity types, it might help to review the documents in the collection first. By reviewing the content, you can get a feel for which terms have significant meaning and look for logical ways to group such terms.

Label significant terms

From the *Label documents* view, find terms of significance in the documents from your collection and label them to indicate their entity types.

Before you begin labeling documents, decide whether you want to keep bulk labeling enabled. The bulk label feature is a great way to speed up the process of labeling your documents. When enabled, every term that you label is labeled automatically everywhere it occurs in the document. Otherwise, you must label each occurrence of the term one at a time.

If you decide that you don't want to bulk label examples, set the **Bulk label entity examples** switch to **Off**. For more information, see Labeling examples in bulk.

Labeling tips

Review these tips before you begin:

- The document collection that you label must contain a representative set of documents. The documents must have many and varied examples of the entity types that you want the entity extractor to recognize. If the collection you selected when you started to create the entity extractor does not meet the requirement, stop now and start over with a different document collection.
- Define entity types that are clearly distinct from one another.
- Aim to label at least 40 examples of each entity type.
- Label every valid example of an entity type. Do not skip any occurrences. To speed up the process, use the bulk label feature.

Labeling entity examples

Label terms in the document that represent examples of the entity types you defined. When you are done with one document, switch the document status from *In progress* to *Complete*, and then move on to the next document.

To label entity examples, complete the following steps:

1. Review the text of the document. Look for entity examples to label.

   The following table shows some examples.

Entity type	Examples to label in the document
color	white, green, purple
car	convertible, SUV, sedan
auto_model	Explorer, Civic, Sorrento
auto_manufacturer	Ford, Honda, Kia
clothing	shirt, blouse, skort
instruments	bonds, stocks, ETFs, munis

Entity types and examples

   If an entity type that you want to identify is not created yet, add the entity type. From the *Entity types* panel, click **Create new**. For more information about adding entity types, see Defining entity types.

2. First, click the entity type from the *Entity types* panel.

3. In the document body, select the word or phrase that represents the entity example.

   The term is selected and a color label is applied to the term. The first two characters of the entity type name are shown in uppercase superscript within the label boundary. Both the 2-character ID and the label color help you to associate the example with the entity type it represents.

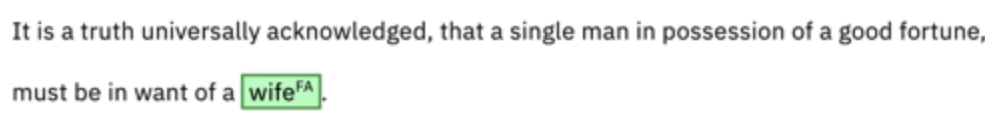


Figure 3. A label is applied to an entity example

   The example text is also added to the *Entity types* panel. If you click the chevron to view details, you can see that the example is listed. The example text is saved in lowercase, regardless of the capitalization that is used in the original text.

4. If bulk labeling is enabled, a notification is displayed to show the number of occurrences of the term that were found and labeled in the current document.

5. If you want to label occurrences of the term in all of the documents in the collection, click **Apply to all documents** .

   When you enable this option, occurrences of the term are labeled in all of the documents in the collection, including documents that you already reviewed and marked complete.

   You are asked to confirm the action because it cannot be undone. If you don't want to have to confirm the action every time you choose to apply bulk labeling to all documents, select **Do not ask for confirmation again** . Click **Run**.

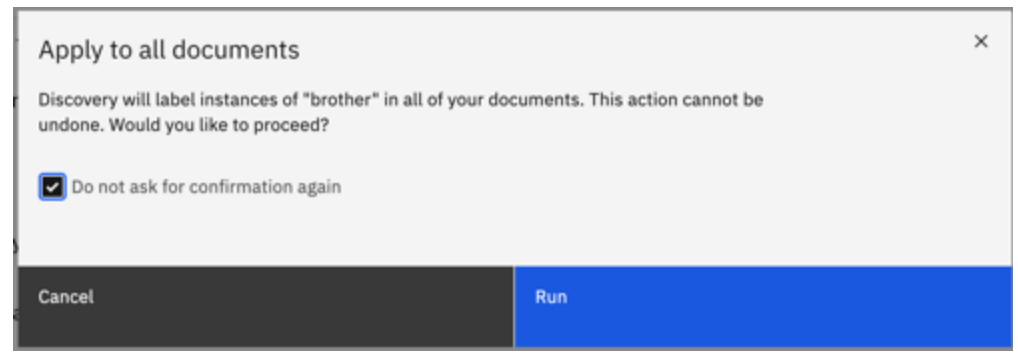


Figure 4. Bulk labeling configuration confirmation

For more information, see Labeling examples in bulk.

6. Scroll through the document to label every valid example of every entity type that you want your extractor to recognize.

   Note: You can search for terms that you want to label as entity examples. For more information, see Searching for examples by using keywords.

   Important: The machine learning model learns as much from the terms that you don't label as the terms that you do.

   If you miss labeling a valid example, the model learns that when the term is used in that context, it is not a valid mention of the entity type. In some cases, an omission is appropriate. For example, some terms have different meanings in different contexts. You don't want to label the term when it is used in the wrong context. However, if the term is used in the right context and you don't label it, you are teaching the model to ignore it. You decrease the model's effectiveness when your training data is inconsistent.

   After you label many examples, entity example suggestions are displayed. You can accept or reject entity example suggestions.

Figure 5. Decide whether to accept a suggestion

   Accepting example suggestions is another way to speed up the labeling process. For more information, see Entity example suggestions. After you accept a suggestion, you can bulk label the term.

7. If you make a mistake and label the wrong word or a word was labeled incorrectly by the bulk labeling process, you can delete the label.

   Hover over the labeled word until the **Delete this example** option is displayed, and then click it. You can choose to delete only this mention or all of the mentions in the document. Make a choice, and then click **Delete**.

8. After you label all of the entity examples in the current document, change the document status from *In progress* to **Complete**.

   Another document from the collection is displayed.

9. Label examples of your entity types in each document in the collection.

   At any time during the labeling process, you can click **Save entity extractor** to save your work.

10. If you don't have enough examples in the current set of documents, you can add more documents.

    From the *Document list* panel, click **Add documents**. The option is available only when more documents are available in the collection. You can add up to 20 documents. If bulk labeling for all documents is enabled, labels are applied to the newly-added documents automatically.

11. After you label examples in as many documents in the collection as you want, click **Save entity extractor**, and then open the *Train extractor* page.

Searching for examples by using keywords

Using the search feature, you can find entity examples in a document and label them easily. You can also use search to find labeled examples and unlabeled examples and correct any labeling inconsistencies.

To search by using keywords, complete the following steps:

1. On the *Label documents* view, click the *Find* icon.
2. In the **Find** field, specify a keyword to search in the document.

   The search results from the document are displayed when you enter the keyword.

   To browse the search results, you can click the *Next result* and *Previous result* icons. To choose a label for an unlabeled example in the result, click the *Edit label* icon and select a label. You can also remove a label from an already labeled example in the result by clicking the *Edit label* icon.

3. To filter the search results, click the *Show filter options* icon.

   The following table describes the filter options.

Option	Description
All	To find all examples in a document that match the keyword.
Labeled text	To find existing labeled examples in a document that match the keyword.
Unlabeled text	To find unlabeled examples in a document that match the keyword.
Match case	To find examples that match both the keyword and its case.
Whole words	To find examples that match the word boundaries of the keyword. For example, if you specify *york* as the keyword, *yorktown* is not matched when this option is selected.

Filter options in find

For the unlabeled examples in the results, you can accept or reject a label suggestion.

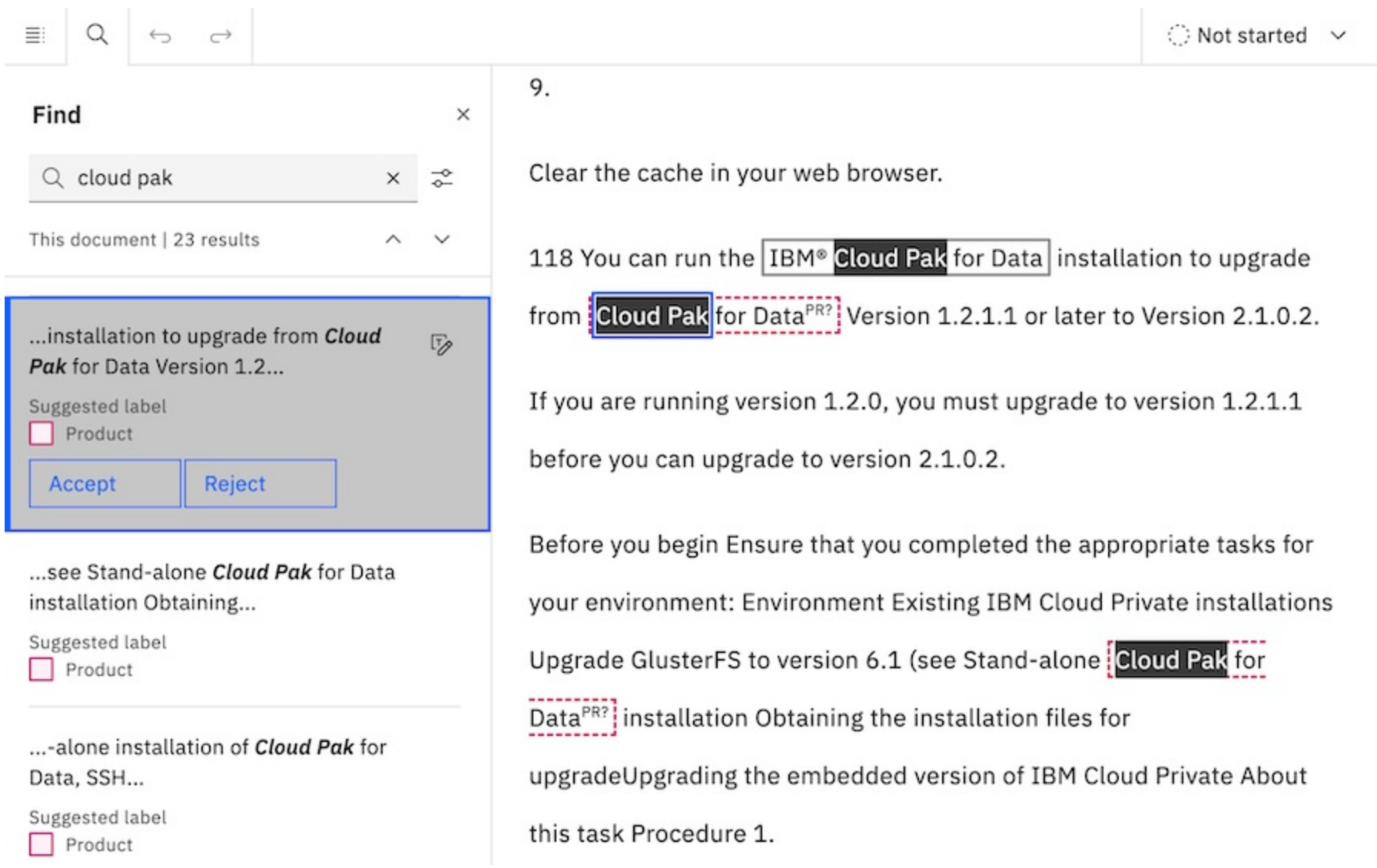


Figure 6. A label is suggested for an entity example

To resolve any overlapping examples, click **Review suggestions** and choose an entity example suggestion from the *Overlapping entity example suggestions* dialog box.

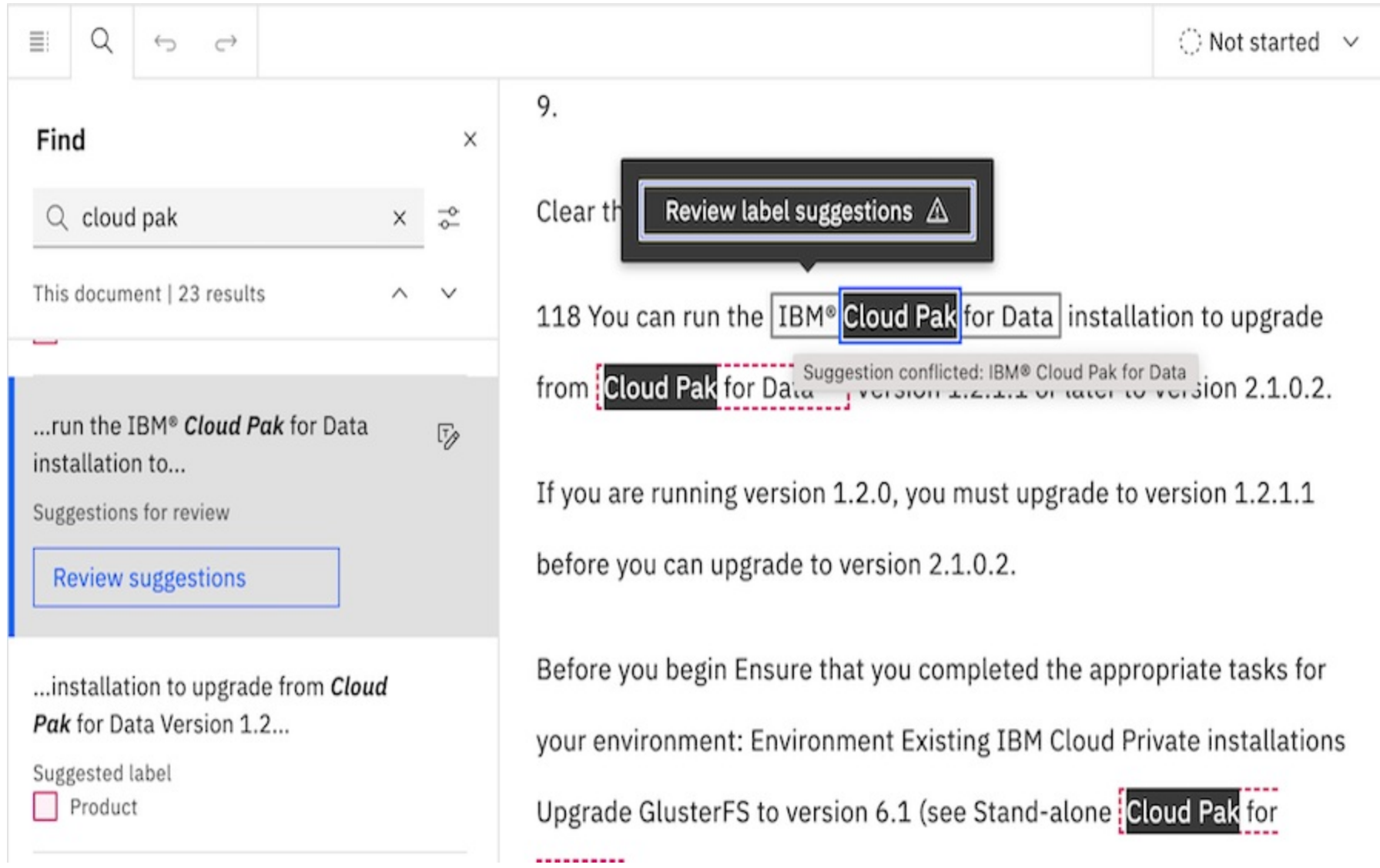


Figure 7. A suggestion conflict for an entity example

Labeling examples in bulk

For most entity examples, enabling the bulk label feature is helpful. You might want to skip it if a term has more than one meaning in different contexts. In that case, evaluate each occurrence individually. Remember, if you enable the bulk label feature, you can check the accuracy of the labels that were added automatically and make corrections when necessary as you review the document.

After you enable the bulk label feature, a notification is displayed that indicates how many occurrences of an entity example were found in the current document. From the current page, the labeling tool cannot access other documents to report how many occurrences exist in other documents from the collection. However, the mention count is shown in the *Entity types* panel. When you first open other documents, you can check the mention counts to see how many mentions were labeled automatically.

Did the bulk label feature miss an occurrence?

Occurrences of the term are not labeled if they occur in the same phrase in which the term is already labeled. For example, the first occurrence of the term `husband` is not labeled when the bulk label feature is switched on for the second occurrence of the term in the following sentence.

Figure 8. Treatment of overlapping labels

Entity example suggestions

After you label enough examples, suggested entity type examples are displayed. The system learns from the types of examples you label, and applies what it learns to identify potential new examples. For example, after you label `red`, `orange`, `yellow`, `green`, and `blue` as examples of the `color` entity type, the *Example suggestions* panel might show `indigo` and `violet` as suggested examples for you to label. Suggestions are not displayed until after you label many examples of an entity type.

The following example shows suggestions that are made for family member mentions.

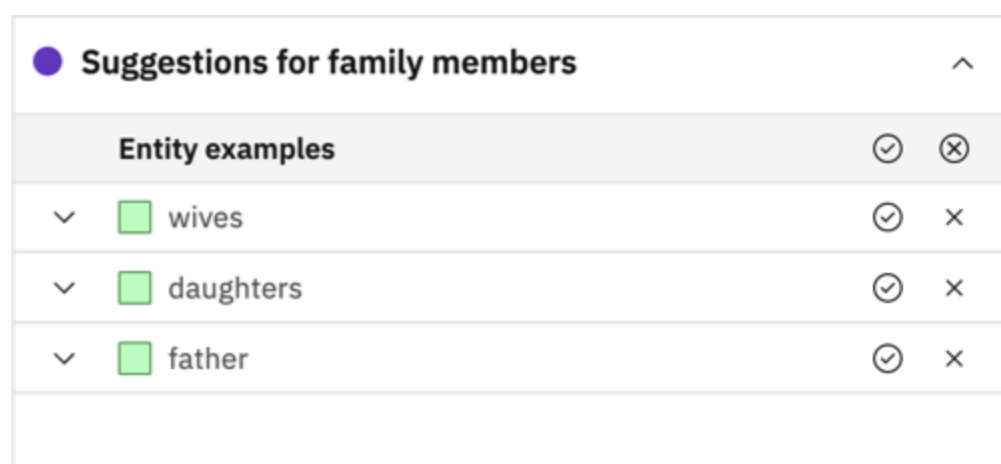


Figure 9. Entity example suggestions

You might notice that a term that you chose to bulk label is not labeled, but is displayed as a suggestion instead. A term is skipped in the following situations:

- The term might occur in different noun phrases in different sections of the document. For example, the term `father` might occur in the noun phrases `the kindest *father*` and `to her *father*`. When a word is included in a noun phrase with adjectives, the meaning can change. Therefore, such terms sometimes are suggested rather than labeled automatically.
- A word might be a valid example on its own and as part of a multiple-word mention. For example, a mention of `IBM` might refer to the company *International Business Machines, Corp.* or might be used as part of a product name, such as *IBM Cloud Pak for Data* . However, a word or phrase can be part of only one example. Example labels cannot overlap one another. Therefore, you must choose which example suggestion is the most accurate. In this example, where the term *IBM* is used as part of a product name, it is more accurate to label the full phrase as an example of the `Product` entity type.
- The service might recognize that a term is a possible example of more than one entity type. For example, the word `top` might mean *the best* or might mean *shirt*.

To investigate a suggestion further, click it to see the word in context within the document. Seeing the term in context helps you to decide whether the occurrence is a valid entity example for you to label.

Exporting labeled data for an entity extractor

You can export the labeled data for an entity extractor from Discovery. You can use the exported labeled data for training or building large language models (LLMs) on a service such as Watson Studio and Natural Language Processing (NLP).

To export the labeled data, complete the following steps:

1. From the *Improvement tools* panel of the *Improve and customize* page, expand *Teach domain concepts*, and then click *Extract entities*.
2. For the entity extractor from which you want to export labeled data, click the *Actions* icon, and then select **Download labeled data**.

   A compressed file is downloaded with labeled data. The compressed file contains the following JSON files.

   - `labeled_data.json`: Includes the text and labels. The data format is based on the input data format for entity extraction in Watson Natural Language Processing. For more information, see Input data format.
   - `metadata.json`: Includes metadata for the workspace and labeled data.

Importing a Knowledge Studio corpus

Note: For installed deployments, the import capability was added with the 4.6.2 release.

You can import a corpus of documents that were annotated in IBM Watson® Knowledge Studio to use as the training data for an entity extractor in Discovery.

Entity types that were defined in Knowledge Studio are shown as new entity types in Discovery. You can continue to annotate the imported documents when you customize the entity extractor model.

Entity subtypes and relations from the Knowledge Studio machine learning model are not represented, neither are any custom dictionaries that are associated with the model.

Before you can import a corpus, you must export the document set from Knowledge Studio as a .zip file. Follow the appropriate steps for exporting based on your Knowledge Studio deployment type:

- IBM Cloud
- Cloud Pak for Data

Although you must download both the document set and type system to include annotations in documents that you upload to another Knowledge Studio workspace, the same is not true in this use case. You import only the document set to Discovery. Any annotations in the documents are recreated in Discovery. The Knowledge Studio type system is not needed.

To import a Knowledge Studio corpus, complete the following steps:

1. Open the project where you want to import the corpus.
2. From the *Improvement tools* panel of the *Improve and customize* page, expand **Teach domain concepts**, and then click **Extract entities**.
3. Click the arrow that is associated with the *New* button. and then click **Import a Knowledge Studio corpus**.
4. Add an extractor name and optionally a description.

   This name is used as the model name and as the name of the enrichment that is created when you publish the model. The name is displayed as the enrichment name in the Enrichments page where you and others can apply it to collections. It also is displayed as the model name in the JSON representation of documents where custom entities are found. The name is stored with the capitalization and spacing that you specify.
5. Click **Upload**, and then browse to find and select the .zip file that you exported from Knowledge Studio. Click **Create**.

The annotated documents that you upload are stored with the entity extractor workspace, not as a new collection in the project. You can continue to annotate the documents.

Give Discovery some time to import and process the machine learning model corpus. After the entity extractor is created, the extractor is opened to the *Label documents* page.

Training the extractor

After you label documents, review the training data that will be used to train the entity extractor model.

To train the extractor, complete the following step:

1. Decide whether you want to apply an advanced option. Most models do not require changes to these options.

   The following customizations are available from the *Review and finish* page:

   - Include documents that were not reviewed by a person in the training set.

     Typically, only documents that a person labeled, reviewed, and explicitly marked complete can be candidates for inclusion in the training set. However, if you want to allow documents that were not marked complete to be included in the training set, you can do so.

   - Change the ratio of documents that are included in the document sets that comprise your training data.

     The documents from your collection are split at random into the following sets:

     - Training set: The documents that you label and that are used to train the entity extractor machine learning model. The goal of the training set is to teach the machine learning model about correct labels.
     - Test set: The documents that are used to test the trained model. After you run a test, you can review the results, closely analyze areas where the model got something wrong, and find ways to improve the model's performance.
     - Blind set: Documents that are set aside and used to test the model periodically after several iterations of testing and improvement are completed. The documents in the blind set are intentionally roped off. As you test the model with documents from the test set and analyze the results, you become familiar with the underlying test documents. Because the test documents are used iteratively to improve the model, they can start to influence the model training indirectly. That's why the blind set of documents is so important. The blind set gives you a way to generate an unbiased evaluation of the model periodically.

     The default split applies a ratio (70%-23%-7%) that is commonly used for machine learning training.

2. Click **Train extractor**.

When you train the extractor, Discovery uses documents from the training set to build a machine learning model. After the model is generated, it runs a test against the documents from the test set automatically. The results of the test are displayed for you to review.

Troubleshoot training issues

Learn about possible error messages and how to address them.

The training data is too large

Your training data contains large text document or many entity types and resources that are needed to process the data is greater than the resources that are available to your service instance. This error can occur even when your workspace doesn't exceed the documented entity extractor limits. To resolve the issue, you can try one of the following approaches:

- Remove one or more entity types to decrease the size of your training data.
- Remove extra large documents from the training data. For example, if one of the labeled documents is extra large, change its status from *Completed* to *In progress* to omit it from the training data.
- Reduce the number of documents that are included in the training set. The default split ratio (70%-23%-7%) for the training data uses 70% of the documents in the training set. You can change the percentage of documents that are used in the training set to a smaller number. For example, you might change the split ratio to 60%-33%-7%.
- IBM Cloud Pak for Data Increase the capacity of your deployed service instance by scaling up service pods.

Evaluating the extractor

To review metrics from the test run of the entity extractor model that you created, click the **Evalute extractor** tab.

The following table describes the available evaluation metrics.

Metric	Description

Confusion matrix	A table that provides a detailed numeric breakdown of annotated document sets. Use it to compare entity type mentions that are labeled by the machine learning model to entity type mentions that are labeled in the training data.
F1 Score	Measures whether the optimal balance between precision and recall is reached. The F1 score can be interpreted as a weighted average of the precision and recall values. An F1 score reaches its best value at 1 and worst value at 0. Overall scores are lower if the model doesn't have enough training data to learn from.
Precision	Measures how many of the overall extracted mentions are classified as the correct entity type. A false positive is when an entity label is incorrect, but was predicted to be correct (Predicted = Positive, Actual = Negative). False positives typically mean low precision.
Recall	Measures how often entity type mentions that should be extracted are extracted. A false negative is when an entity type label is correct, but was predicted to be incorrect (Predicted = Negative, Actual = Positive). False negatives typically mean low recall.

Metrics details

1. Review the metrics that are provided about the extractor model test run to determine whether more training is needed.
2. Explore the test results in more detail by clicking **Review training results in test set** .

   Documents from the test set are displayed with the predicted labels shown in one panel and the ground truth shown in the other.
   - Predicted labels are the examples that the entity extractor identified and labeled as entity types.
   - The *ground truth* has examples that a person labeled or that were bulk labeled and reviewed by a person. Labels in the ground truth are considered the correct labels.

   The performance of the model is rated based on how closely the predicted labels match the ground truth.

Improving the extractor

The following table shows suggested fixes for common problems.

Problem	Action to remedy the problem
Low overall scores	You might not have enough documents with labeled examples in your training set. Label more examples in more of your documents.
Low recall	Label more documents with new examples of the entity types that the extractor missed.
Low precision	Look for entity types that are commonly confused. Find and label more examples of each entity type to help the entity extractor distinguish between the entity types.

Improvement actions

Adding documents to the training data

To add more documents, complete the following steps:

1. Open the *Label documents* tab.
2. From the *Document list* panel, choose **Add documents**.

   This button is disabled if no other documents are available to add to the entity extractor from the current collection. To add more documents to the collection, go to the *Activity* page for the collection, and then click the *Upload data* tile to browse for and add more files.

Tip: You cannot choose the documents from the collection to show in the *Document list* for labeling purposes. If there are specific types of documents that you want to label, consider adding representative documents to a collection that you can use to create the entity extractor.

There are limits to the number of documents that can be included in the training data. If your training data includes documents with a combination of sections that are labeled and others that are not, the system might sample some examples from unlabeled sentences. Subsampling helps to balance the number of positive and negative examples that are used for training. Balancing the examples in the training set improves the training performance.

Publishing the entity extractor as an enrichment

When you think the entity extractor is ready, publish the entity extractor. How do you know when it's ready? If the score doesn't change after several test runs in which you make improvements, the model is ready. You can return to update and retrain the model after you publish it.

1. From the *Evaluate extractor* page, click **Publish extractor**.
2. Click **Apply to data**.

3. Choose a collection, and then select the document field where you want the entity extractor enrichment to be applied.
4. Click **Apply**.

Exporting the entity extractor

> **Note:** For installed deployments, the export capability was added with the 4.6.2 release.

An entity extractor model that you create and deploy in one project is available as an enrichment that can be applied to a collection from any project in the same service instance.

If you want to use the entity extractor model in a project from another service instance, you can export the entity extractor. To use it elsewhere, follow the steps to create a machine learning model from Use imported ML models to find custom terms. You cannot continue to edit an entity extractor that you import into another project.

The entity extractor that you want to export must be fully trained.

To export an entity extractor, complete the following steps:

1. Open the project with the entity extractor that you want to export.
2. From the *Improvement tools* panel of the *Improve and customize* page, expand *Teach domain concepts*, and then click **Extract entities**.
3. From the *Entity extractors* list, find the entity extractor that you want to export.
4. Click the *Actions* icon for your extractor, and then choose **Download model** to save the model to your system.

> **Note:** The *Download model* option is not available unless the model is trained.

The entity extractor model is saved as a .ent file. You can import it into a project in another service instance as a machine learning model, and then apply it to your collections. For more information about importing the model, see Use imported ML models to find custom terms.

Applying an entity extractor enrichment

When you publish the extractor, you specify the field where you want the extractor to be applied. If you decide to apply the enrichment to different or more fields later, you can follow these steps to do so.

1. From the navigation panel, click **Manage collections**.
2. Click to open the collection where you want to apply the enrichment.
3. Click **Enrichments**.
4. Find the entity extractor name in the list, and then choose a field to apply the enrichment to.
5. Click **Apply changes and reprocess**.

For more information about how to remove an entity extractor enrichment from a collection, see Managing enrichments.

Entity extractor output

When the enrichment recognizes one of your custom entities in a document, an entry is added to the `enriched_text.entities` section of the JSON representation of the document. The section contains occurrences of entities that are recognized by your custom model along with entities that are recognized by the built-in Entities enrichment. The built-in enrichment uses the Watson NLP service to identity entites that are part of what it calls the *Natural Language Understanding* type system. For more information about the built-in Entities enrichment, see Entities.

The following JSON output is produced by a custom model that is named *literature* that recognizes family member mentions.

```
        {
          "model_name": "natural_language_understanding",
          "mentions": [
            {
              "confidence": 0.4220163,
              "location": {
                "end": 12418,
                "begin": 12411
              },
              "text": "evening"
            }
          ],
          "text": "evening",
          "type": "Time"
        },
        {
          "model_name": "literature",
          "mentions": [
            {
              "confidence": 1,
              "location": {
                "end": 443,
                "begin": 437
              },
              "text": "sister"
            },
```


Figure 10. JSON representation of a custom entity mention

Monitoring performance over time

You can retrain your entity extractor model at any time. Each time that you train the model, review the performance metric scores to determine whether your most recent changes increase or decrease the model's scores.

1. To compare one test run against another, click **View score history**.

   The history view shows the last 5 training runs.

   > **Tip:** To retain the score information for more than the most recent 5 training runs, you can export the metrics in comma-separated value format, and track the scores in a separate application. Click the tabular representation icon, and then click **Download as CSV**.

If a subsequent training run results in lower scores, don't publish that version of the model.

Deleting an entity extractor

You can delete an entity extractor if it is not in use, meaning the enrichment that is published from the entity extractor is not applied to a collection.

You might want to delete an entity extractor if you hit the limit for the maximum number of extractors that are allowed for your plan, for example.

Remember, limits are defined per service instance, not per project. If you cannot create new entity extractors, but do not have the maximum number of extractors in the current project, check other projects in the same service instance. There might be entity extractors that aren't being used in other projects that can be deleted.

1. Remove the entity extractor enrichment that was published from the entity extractor that you want to delete from any collections where it is being used.

   For more information, see Deleting enrichments.

2. From the *Improvement tools* panel of the *Improve and customize* page, expand **Teach domain concepts**, and then click **Extract entities**.

3. Find the entity extractor that you want to delete, click the *Actions* icon, and then select **Delete**.

Entity extractor limits

The number of entity extractors that you can create per service instance depends on your Discovery plan type.

Plan	Entity extractors per service instance [1]	Maximum entity types per extractor	Maximum documents in training data
Cloud Pak for Data	Unlimited	18	1,000
Premium	10	18	1,000

Enterprise	10	18	1,000
Plus (including Trial)	3	12	200

Entity extractor plan limits

1. This number reflects the number of published entity extractor enrichments for the service instance (including from imported entity extractor models) whether they are applied to a collection or not. ↩

Classify sentences

IBM Cloud

Sentence classification is a beta feature that is available in managed deployments only. Also, the feature is available for English-language documents only.

Use sentence classification to classify sentences in your documents that are of significant business interest.

Sentence classification uses a machine learning model that classifies sentences based on user-defined sentence classes. You can label example sentences in your documents to define the sentence classes. To speed up the labeling process, the system automatically prepares a suggestion model in the background and provides you with more sentence labeling suggestions.

Before you begin

Find or create a collection with documents having various sentence examples that you want Discovery to learn about. To teach the sentence classifier, you must label examples of sentence classes. You can only label examples if your collection contains valid examples. Try to find documents that have many and varying sentences that function as examples of every sentence class that you want to define.

Adding a sentence classifier

To add a sentence classifier, complete the following steps:

1. Open the project where you want to create the sentence classifier.

   The project must have at least one collection with documents that are representative of data that you want to classify.

2. From the *Improvement tools* panel of the *Improve and customize* page, expand **Teach domain concepts**, and then click **Sentence classifiers**.

3. Click **New**.

4. Add a sentence classifier name and optionally a description.

   This name is used as the model name and as the name of the enrichment that is created when you publish the model. The name is displayed as the enrichment name in the *Enrichments* page where you and others can apply it to collections. It is also displayed as the model name in the JSON representation of documents where sentence classes are found. The name is stored with the capitalization and spacing that you specify.

5. Choose a collection with documents that are representative of data that you want to classify.

6. Choose fields from the document to show in the document view where you will label documents from the collection.
   - **Document title** is shown in the page header as the document name. Choose a field that has a unique value per document, such as the file name, which is stored in the `extracted_metadata.filename` field.
   - **Document body** is where you label sentence examples in the content. Choose a field that contains the bulk of the document content, such as the `text` field.

7. Click **Create**.

A document from the collection that you selected is displayed in the *Label documents* view. You will label occurrences of the sentence classes that you want Discovery to recognize from this and other documents in the collection.

Tip: If no text is displayed in the body of the page, start over now by creating a new sentence classifier. This time, when you select a value for the *Document body* field, be sure to choose a field from your processed documents that contains text.

Defining sentence classes

Define sentence classes by completing the following steps:

1. Click **Add a sentence class**.

2. Add the sentence class name and an optional description.

3. Optional: Pick the color to use for the sentence class in the document.

You can click a color from the *Label color* palette, click the *Renew color* icon to tab from one color to the next. To use a custom color, specify its hexadecimal color code (#fff0f7).

4. Click **Create**.
5. Repeat this process to add all of the sentence classes that you want the classifier to classify.

   If you aren't sure what to add for sentence classes, it might help to review the documents in the collection first. By reviewing the content, you can get a feel for which sentences have significant meaning and look for logical ways to classify such sentences.

Labeling sentences

From the *Label documents* view, find sentences in the documents from your collection and label them to indicate their sentence classes. While you are labeling, the sentence classifier automatically trains a model in the background to present labeling suggestions.

The labeling suggestions speed up the process of labeling your documents. Instead of going through documents and spending time reading text, the labeling suggestions enable you to easily find relevant examples and examine them in context.

You can label sentences in the following ways:

- Manual labeling. For more information, see Manual labeling.
- Find-in-document labeling. For more information, see Find-in-document labeling.
- Smart labeling. For more information, see Smart labeling.

Note: Unlike entity extractor, there is no concept of document status such as *Not started*, *In progress*, and *Complete* for sentence classifier. In the case of entity extractor, you mark documents as *Complete* to indicate that they are used for training. In the case of sentence classifier, only labeled sentences are used for training and unlabeled data in each document is ignored.

Feature	What is used for training
Entity extractor	All the documents that are marked as Complete
Sentence classifier	All the labeled sentences (unlabeled sentences are not used)

Data used for training

Manual labeling

To label manually, complete the following steps:

1. Read the documents shown in the *Label documents* page to locate appropriate sentence examples to label.
2. Select a sentence example and click the *Edit labels* icon.
3. Select a sentence class from the list to label the example sentence as a positive label.

   To label the example sentence as a negative label, press Shift when selecting the sentence class in the list.

   If you don't find appropriate sentence examples, select a different document from the *Document list*.

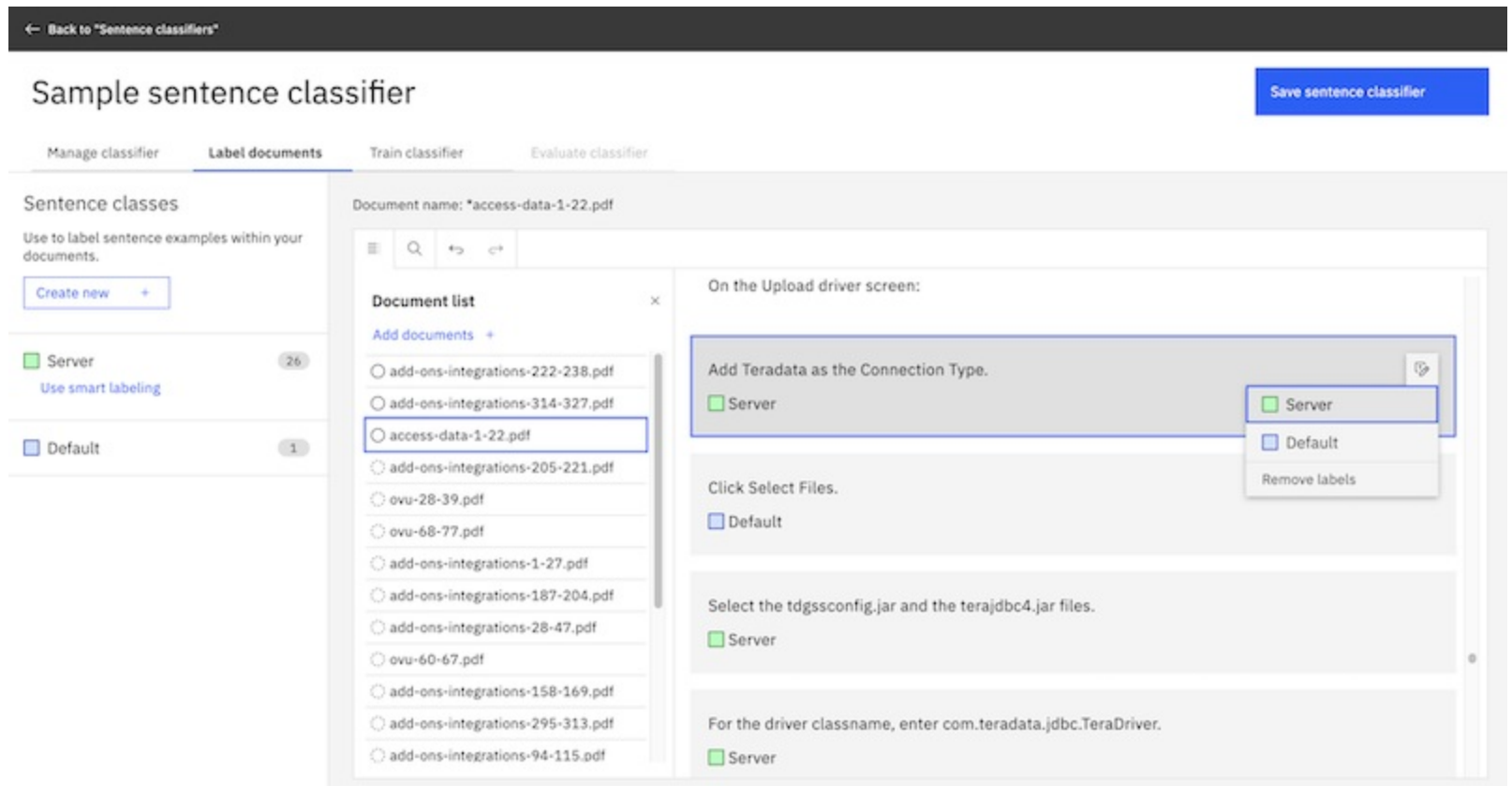


Figure 1. Labeled sentence examples

4. Repeat the steps to label example sentences as positive or negative in other documents of the collection.

While labeling, click **Save sentence classifier** to save your work. If you move to another page from the *Label documents* page, the system automatically saves your work.

Tip: Use shortcut keys for quick labeling operations. Press keys 1 to 5 corresponding to a sentence class shown in the list to label the example sentence as a positive label. Similarly, press keys 6 to 0 to add a negative label. You can press the Delete key or Backspace key to remove labels from the selected sentence.

Find-in-document labeling

Using the search feature, you can find sentence examples in a document and label them easily. You can also use search to find labeled examples and unlabeled examples and correct any labeling inconsistencies.

To find examples and label them in a document, complete the following steps:

1. On the *Label documents* view, click the *Find* icon.
2. In the **Find** field, specify the text to search in the document.

   The search results from the document are displayed when you enter the text.

   Note: To browse the search results, you can click the *Next result* and *Previous result* icons. To choose a label for an unlabeled example in the result, click the *Edit labels* icon and select a sentence class from the list. You can also remove labels from an already labeled example in the result by clicking the *Edit label* icon and selecting **Remove labels**.

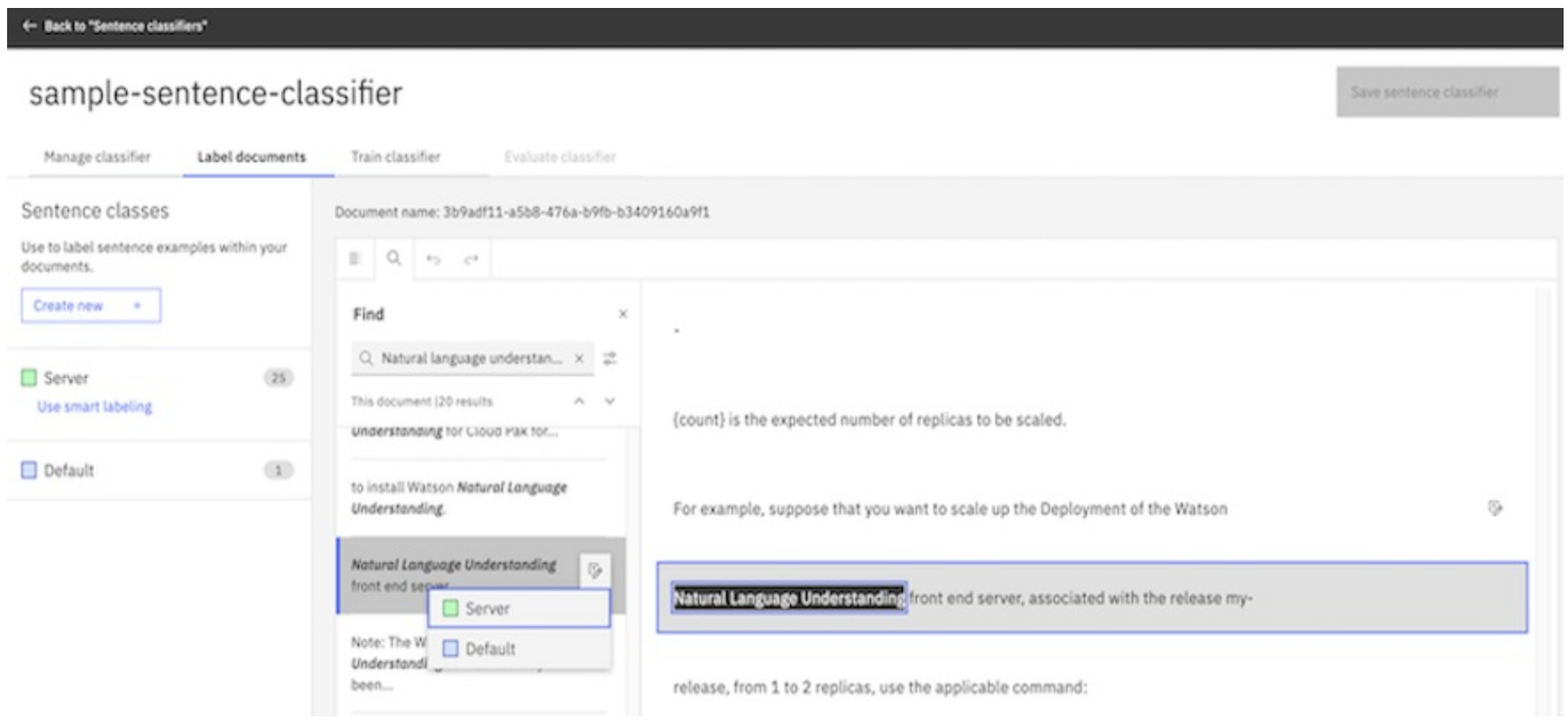


Figure 2. Search results

3. To filter the search results, click the *Show filter options* icon.

The following table describes the filter options.

Option	Description
All	To find all examples in a document that match the text.
Labeled text	To find existing labeled examples in a document that match the text.
Unlabeled text	To find unlabeled examples in a document that match the text.
Match case	To find examples that match both the text and its case.
Whole words	To find examples that match the word boundaries of the text. For example, if you specify *installing* in the text, *uninstalling* is not matched when this option is selected.

Filter options in find

1. Repeat the steps to label example sentences in other documents of the collection.

Smart labeling

The smart labeling feature uses active learning techniques to suggest sentence examples that you can label. Smart labeling speeds up the labeling process, but you must label at least 20 examples for each sentence class first so that the system can build a suggestion model.

To use smart labeling, complete the following steps:

1. Label a minimum of 20 positive examples for at least one sentence class.

   The system automatically starts preparing a suggestion model in the background.

   After a version of the suggestion model is ready, the system provides suggestions on which sentences to label next. Labeling the suggestions helps improve the sentence classifier model the most.

2. Click **Use smart labeling**.

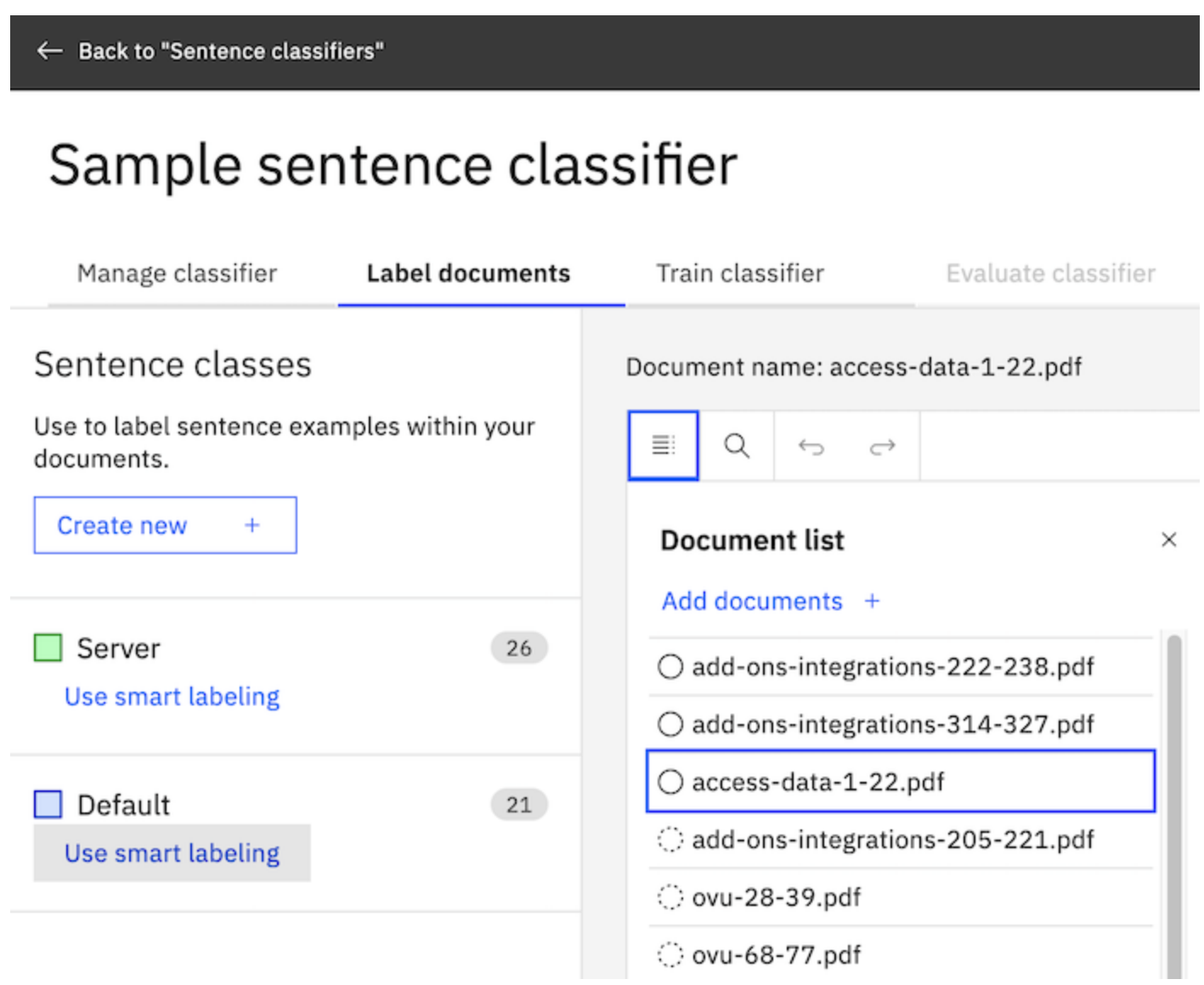


Figure 3. Smart labeling

   The Smart labeling pane is displayed. In this pane, you can label sentences for a specific sentence class.

3. Select an example from the suggested *Examples*, and click **Yes** to label the example sentence as a positive label. You can click **No** to label the example sentence as a negative label.

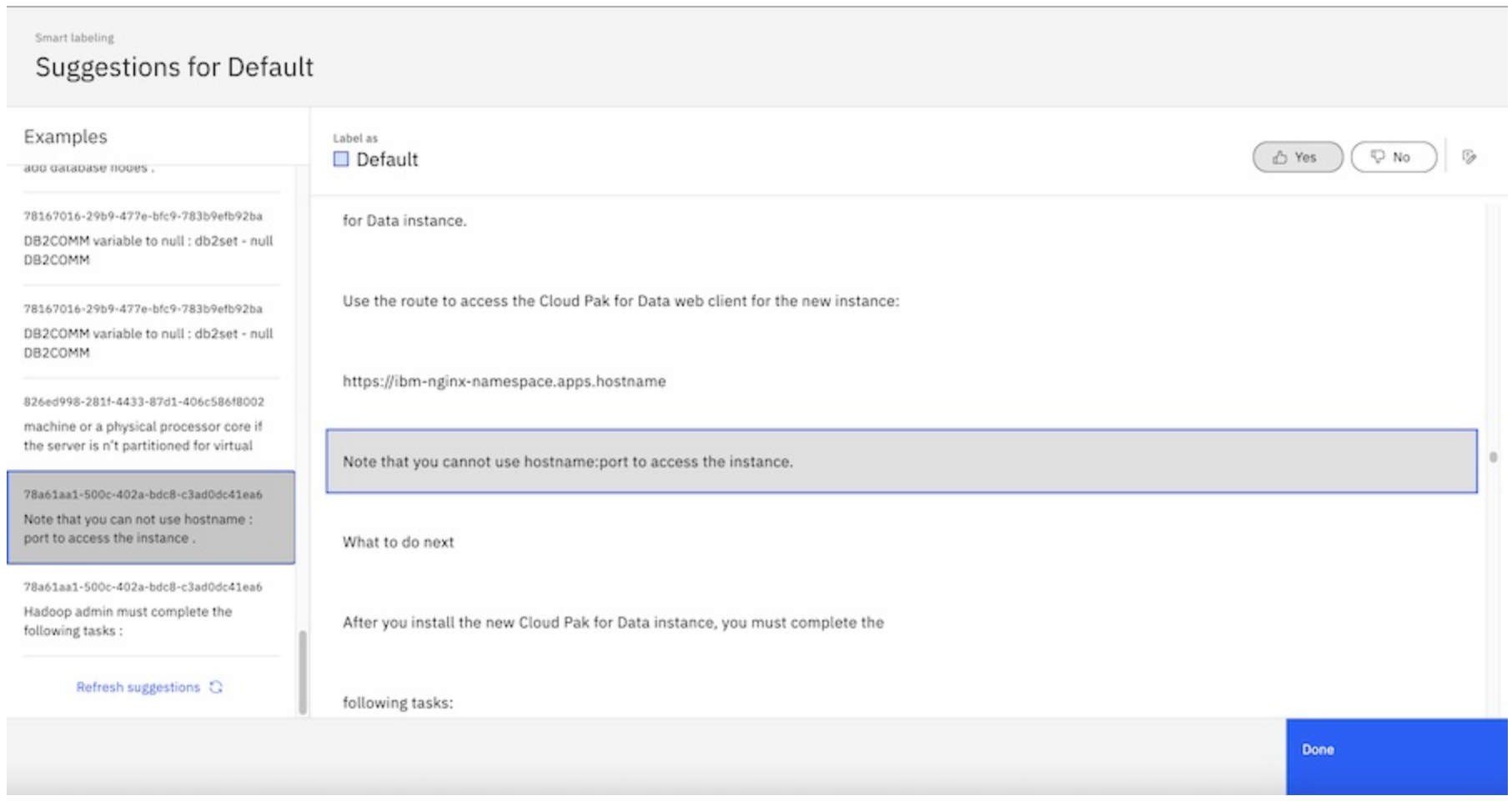


Figure 4. Smart labeling pane

4. Repeat the previous step to label other suggested examples.

   To refresh the suggested examples in the list, click **Refresh suggestions**. The list is refreshed with new suggestions and the existing labeled examples are not shown in the list.

5. Click **Done** after you have labeled the sentences.

   If you don't have enough sentence examples in the current set of documents, you can add more documents. This option is available only when there are more documents in the collection. For more information, see Adding documents to the training data.

6. After you label examples in as many documents in the collection as you want, click **Save sentence classifier**, and then train the classifier. For more information, see Training the classifier.

Smart labeling tips

Remember these tips for smart labeling:

- Use shortcut keys for quick labeling operations. You can press the Left Arrow key to choose Yes, or the Right Arrow key to choose No. Choosing Yes or No is equivalent to specifying a positive or negative label for the sentence class.
- To label an example sentence as a positive label, press keys 1 to 5, corresponding to a sentence class shown in the list. Similarly, press keys 6 to 0 to add a negative label. You can press the Delete key or Backspace key to remove labels from the selected sentence.
- If an example is irrelevant to the current sentence class, label the example as negative instead of leaving it unlabeled. Unlabeled data is ignored and not used for training, so specifying negative labels is important for improving the classification model.
- If you label 20 or more examples (either positive or negative labels) after the last suggestion model is trained, the system automatically starts building a new suggestion model in the background. You are notified when the new suggestions are ready for labeling.

Adding documents to the training data

To add more documents, complete the following steps:

1. Navigate to the *Label documents* view.
2. In the *Document list* panel, click **Add documents**.

This option is not available when there are no other documents in the collection to add to the sentence classifier workspace. To add more documents to the collection, navigate to the *Activity* page for the collection, and then click the **Upload data** tile to browse and add more documents.

Even if you add more documents from a collection, all the documents may or may not be used for training the model. Unlike entity extractor where all the completed documents are used for training, sentence classifier uses only labeled sentences for training and unlabeled data is ignored.

Note: You cannot choose the documents from the collection to show in the *Document list* panel for labeling. If there are specific types of documents that you want to label, consider creating a new collection that contains only those documents.

Training the classifier

After you label documents, you can review the training data in the *Train classifier* view. The training data is used to train the sentence classifier model.

To train the classifier, complete the following steps:

1. Navigate to the *Train classifier* view.
2. Review your labeling summary to check if you labeled enough to train the classifier.

   To train the classifier, each sentence class must have a minimum of 20 positive labels and two negative labels. Otherwise, the **Train classifier** button is disabled and you cannot start training. Sentence classes that have no positive labels or negative labels are ignored.
3. Review whether you want to apply advanced options for training. Most models do not require changes to the advanced options.

   Your sentences are split at random into sets. The training set is used to train the classifier. The test set is used to test the model after it is trained. The blind set has reserved sentences that you don't see during training. They are used to generate an unbiased evaluation of the model periodically. The default split uses the standard ratio for training. For more information, see Document sets for training.
4. Click **Train classifier**.

   When you train the classifier, Discovery uses sentences from the training set to build a machine learning model. The results of the test are displayed for you to review in the *Evaluate classifier* view.

Document sets for training

You can change the ratio of sentences that are included in the document sets that comprise your training data.

The sentences you labeled are split at random into the following sets:

- Training set: The sentences that you label and that are used to train the sentence classifier machine learning model. The goal of the training set is to teach the model about correct labels.
- Test set: The sentences that are used to test the trained model. After the model is generated, it runs a test against the documents from the test set automatically. You can analyze the results to determine areas where the model got something wrong, and find ways to improve the model's performance.
- Blind set: Sentences that are set aside and used to test the model periodically after several iterations of testing and improvement are completed. The sentences in the blind set are intentionally roped off. As you test the model with sentences from the test set and analyze the results, you become familiar with the underlying test sentences. Because the test sentences are used iteratively to improve the model, they can start to influence the model training indirectly. That's why you may want to have the blind set of sentences. The blind set gives you a way to generate an unbiased evaluation of the model periodically.

The default split ratio is 70% for the training set, 30% for the test set, and 0% for the blind set. You can have a blind set of sentences by increasing the ratio of the blind set. In this case, the numbers in the classifier score table of the *Evaluate classifier* view, such as **False positive**, **False negative**, and others do not match the number of sentences shown in the *Review training results* view. This is because sentences in the blind set are taken into account for evaluation, but they do not appear in the *Review training results* view.

Evaluating the classifier

To review metrics from the test run of the sentence classifier model that you created, click the **Evaluate classifier** tab.

The following table describes the available evaluation metrics.

Metric	Description
Confusion matrix	A table that provides a detailed numeric breakdown of labeled sentences. Use it to compare what are labeled by the machine learning model to what are labeled in the training data.
F1 Score	Measures whether the optimal balance between precision and recall is reached. The F1 score can be interpreted as a weighted average of the precision and recall values. An F1 score reaches its best value at 1 and worst value at 0. Overall scores are lower if the model doesn't have enough training data to learn from.
Precision	Measures how many of the overall sentences are classified as the correct sentence class. A false positive is when a sentence should not be classified, but was classified (Predicted = Positive, Actual = Negative). False positives typically mean low precision.
Recall	Measures how often sentences that should be classified are classified. A false negative is when a sentence should be classified, but was not classified (Predicted = Negative, Actual = Positive). False negatives typically mean low recall.

Metrics details

1. Review the metrics that are provided about the classifier model test run to determine whether more training is needed.
2. Explore the test results in more detail by clicking **Review training results in test set** .

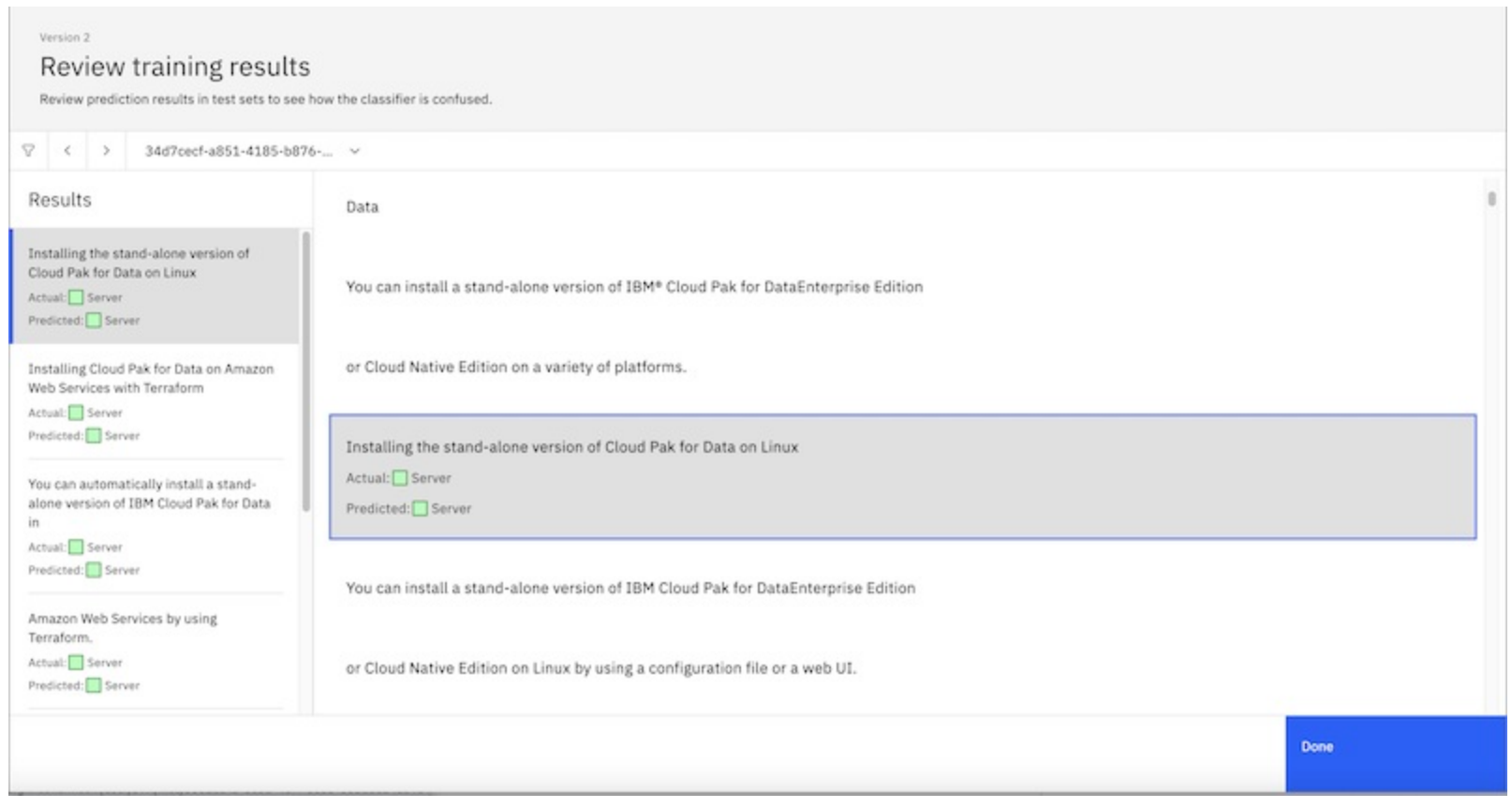


Figure 5. Review training results in test set

Sentences from the test set are displayed with the actual and predicted labels shown in the list on the left pane. If you click on a sentence in the list, it is shown in the document view on the right pane.

- Actual labels are the examples that a person labeled manually. They are considered as the correct labels.
- Predicted labels are the examples that the sentence classifier identified and labeled as sentence classes.

The performance of the model is rated based on how closely the predicted labels match the actual labels.

3. To filter the list, click the *Filter* icon and choose the **Sentence class** and **Prediction**, and then click **Apply**.

Reviewing training results from performance breakdown

To review training results from the performance breakdown, complete the following steps:

1. Click a number in the **Performance breakdown by sentence class** table.

   The *Review training results* dialog box is displayed.

2. Review the relevant sentences, which are based on the number that you clicked.

Improving the classifier

The following table shows suggested fixes for common problems.

Problem	Action to remedy the problem
Low overall scores	You might not have enough labeled sentences in your training set. Label more sentences in more of your documents.
Low recall	Label more documents with new examples of the sentence classes that the classifier missed to classify. Review false negative sentences to check if there are unique terms among them. If you see such unique terms, search for sentences that include such terms and add positive labels to them.
Low precision	Review false positive sentences carefully. You might have missed labeling some sentences. In particular, check sentences with a negative label. When a sentence has a negative label for a particular sentence class (for example, class A), check if a positive label is necessary for another sentence class (for example, class B). If the sentence actually belongs to class B, but you did not specify a positive class B label for it, then it can degrade the precision score. Also, if you find terms that commonly appear in the false positive sentences, then specify negative labels to sentences with such terms.

Improvement actions

Publishing the sentence classifier as an enrichment

When you think the sentence classifier is ready, publish the sentence classifier model. You can consider that the model is ready when the score doesn't change after several test runs in which you make improvements. You can return to update and retrain the model after you publish it.

1. Navigate to the *Evaluate classifier* view, and click **Publish classifier**.
2. Click **Publish**.

3. Click **Apply to data**.
4. Choose a collection, and then select the text field where you want the sentence classifier enrichment to be applied.
5. Click **Apply**.

Downloading the sentence classifier model

A sentence classifier model that you create and deploy in one project is available as an enrichment that can be applied to a collection from any project in the same service instance.

If you want to use the sentence classifier model in a project from another service instance, you can export or download the sentence classifier model. To use it elsewhere, follow the steps to create a machine learning model in Use imported ML models to find custom terms. You cannot continue to edit a sentence classifier that you import into another project.

The sentence classifier that you want to export must be fully trained.

To export a sentence classifier, complete the following steps:

1. Open the project with the sentence classifier that you want to export.
2. From the *Improvement tools* panel of the *Improve and customize* page, expand *Teach domain concepts*, and then click **Sentence classifiers**.
3. From the *Sentence classifiers* list, find the sentence classifier that you want to export.
4. Click the *Actions* icon for your classifier, and then select **Download model** to save the model to your system.

> **Note:** The *Download model* option is not available unless the model is trained.

The sentence classifier model is saved as a .sc file. You can import it into a project in another service instance as a machine learning model, and then apply it to your collections.

Downloading labeled data for a sentence classifier

You can download or export the labeled data for a sentence classifier from Discovery. You can use the exported labeled data for training or building large language models (LLMs) on a service such as Watson Studio and Natural Language Processing (NLP).

To export the labeled data, complete the following steps:

1. From the *Improvement tools* panel of the *Improve and customize* page, expand *Teach domain concepts*, and then click **Sentence classifiers**.
2. For the sentence classifier from which you want to export labeled data, click the *Actions* icon, and then select **Download labeled data**.

   A compressed file is downloaded with labeled data. The compressed file contains the following JSON files.

   - `labeled_data.json`: Includes the text and labels. The data format is based on the input data format for text classification in Watson Natural Language Processing. For more information, see Input data format.
   - `metadata.json`: Includes metadata for the workspace and labeled data.

Converting `labeled_data.json` to CSV

Enter the following command to convert `labeled_data.json` to CSV:

```
$ cat labeled_data.json | jq -r '.[] | [.text, .labels[]] | @csv'
```

The labeled data is converted to the following format:

```
    "sentence1", class-label1, class-label2
    "sentence2", class-label3
    "sentence3", ...
    ...
```

Applying a sentence classifier enrichment

When you publish the sentence classifier, you specify the field where you want the sentence classifier to be applied. If you decide to apply the enrichment to different or more fields later, you can follow these steps to do so.

1. From the navigation panel, click **Manage collections**.
2. Click to open the collection where you want to apply the enrichment.
3. Click **Enrichments**.
4. Find the sentence classifier name in the list, and then choose a field to apply the enrichment to.

You can choose a field that contains either text or html.

5. Click **Apply changes and reprocess**.

For more information about how to apply a sentence classifier enrichment to a collection, see Managing enrichments

Sentence classifier output

When the enrichment classifies one of your sentences in a document, an entry is added to the `enriched_text.element_classes` section of the JSON representation of the document. The section contains sentences classified by your classifier model along with their sentence classes.

> **Note:** A sentence classifier does not classify sentences with confidence scores that are lower than 0.5.

The following JSON output is an example result of sentence classification.


```
"root" : { 6 items
  "document_id" : "c92d1065b39d8d16bec913ef38fc8688"
  "result_metadata" : { 1 item
    "collection_id" : "d8295889-a628-04fd-0000-018b0ca3e837"
  }
  "enriched_text" : [ 1 item
    0 : { 2 items
      "element_classes" :  2598 items
        [ 100 items
          0 : { 4 items
            "model_name" : "My Sentence Classifier 5"
            "classes" : [ 1 item
              0 : { 2 items
                "confidence" : 0.9571497440338135
                "class_name" : "Revenue"
              }
            ]
            "location" : { 2 items
              "end" : 1403
              "begin" : 1297
            }
            "text" :
            "2022 Performance For the year, IBM generated $60.5 billion
            in revenue and $9.3 billion of free cash flow."
          }
```


Figure 6. JSON representation

Monitoring performance over time

You can retrain your sentence classifier model at any time. Each time that you train the model, review the performance metric scores to determine whether your most recent changes increase or decrease the model's scores.

1. To compare one test run against another, click **View score history** in the *Evaluate classifier* view.

   The history view shows the last 5 training runs.

   > **Tip:** To retain the score information for more than the most recent 5 training runs, you can export the metrics in comma-separated value format, and track the scores in a separate application. Click the tabular representation icon, and then click **Download as CSV**.

If a subsequent training run results in lower scores, don't publish that version of the model.

Deleting a sentence classifier

You can delete a sentence classifier if it is not in use, such as when the enrichment that is published from the sentence classifier is not applied to a collection.

You might want to delete a sentence classifier if you reach the limit for the maximum number of sentence classifiers that are allowed for your plan, for example.

There are two different limits, the maximum number of sentence classifier workspaces and the maximum number of sentence classifier enrichments. You create a sentence classifier workspace when you navigate to the *Improvement tools* panel, expand *Teach domain concepts*, click **Sentence classifiers**, and then click the **New** button. You create a sentence classifier enrichment when you publish your trained sentence classifier or upload a sentence classifier

model. For information about the limits, see Sentence classifier limits.

Remember, limits are defined per service instance, not per project. If you do not have the maximum number of sentence classifiers in the current project, but you cannot create new sentence classifier workspaces or publish your trained sentence classifier, then check other projects in the same service instance. There might be sentence classifier workspaces or enrichments that aren't being used in other projects which can be deleted.

Removing sentence classifier enrichments

Remove the sentence classifier enrichment, which was published from the sentence classifier that you want to delete, from any collections where it is being used. For more information, see Deleting enrichments.

Removing a sentence classifier enrichment does not remove its workspace.

Removing sentence classifier workspaces

1. From the *Improvement tools* panel of the *Improve and customize* page, expand *Teach domain concepts*, and then click **Sentence classifiers**.
2. Find the sentence classifier workspace that you want to delete, click the *Actions* icon, and then select **Delete**.

Removing a sentence classifier workspace does not remove the enrichment published from the workspace.

Using the APIs for sentence classifier

The sentence classifier API is beta functionality.

You can use the APIs for applying sentence classifier enrichment to your documents. Using the APIs, you can create a sentence classifier enrichment, and also manage the enrichment such as updating and deleting it.

For using the sentence classifier APIs, do the following things:

1. Create a sentence classifier enrichment using the `create an enrichment` method in the API.

   For more information about creating the enrichment, see Create an enrichment in the API reference.

   You must specify the labeled data to train the sentence classifier model when you create the sentence classifier enrichment. The labeled data must be in the following CSV format:

   ```
   "sentence1", class-label1, class-label2
   "sentence2", class-label3
   "sentence3", ...
   ...
   ```

   Each row is a sentence followed by a comma-separated list of zero or more sentence class labels that are associated with the sentence.

   > **Note:** As a best practice, each sentence class label in the CSV file should be represented by at least 100 sentences to achieve sentence classification of reasonable quality. Sentences that are associated with a sentence class label are considered as positive examples of that sentence class. Sentences that are not associated with a sentence class label are considered as negative examples of that sentence class.

   After the sentence classifier enrichment is successfully created, navigate to the *Manage collections* page, choose the collection, and then open the **Enrichments** tab. You can find the sentence classifier enrichment in the list of available enrichments.

   After the sentence classifier enrichment **Status** is ready, you can apply the sentence classifier enrichment to the documents in your collection.

2. Apply the sentence classifier enrichment that you have created to the fields (text or html) in your documents to classify sentences. For more information about applying an enrichment and managing the enrichment using APIs, see Using the API to manage enrichments.

Sentence classifier limits

The number of sentence classifiers that you can create per service instance depends on your Discovery plan type.

Plan	Sentence classifier workspaces per service instance	Sentence classifier enrichments per service instance	Maximum sentence classes per classifier	Maximum documents in training data
Premium	10	20	5	1,000
Enterprise	10	20	5	1,000

Plus (including Trial)	3	5	3	200

Sentence classifier limits

Use imported ML models to find custom terms

Use custom Machine Learning models that use rules or context to recognize and tag entities.

Add Machine Learning models that you created with IBM tools that you can use to define your own type system.

The type of models you can add depend on your deployment:

- IBM Cloud Pak for Data You can add models that were created with Watson Explorer Content Analytics Studio models, or with an instance of IBM Watson® Knowledge Studio that is hosted on IBM Cloud Pak® for Data or IBM Cloud. Starting with the 4.6.2 release, you can also add custom entity extractor models that were created in and exported from another instance of Discovery.
- IBM Cloud You can add models that were created with a IBM Watson® Knowledge Studio instance that is hosted in IBM Cloud only.

> **Tip:** To use a Knowledge Studio model that was built with Knowledge Studio on IBM Cloud Pak for Data, migrate the ground truth to a IBM Cloud instance of Knowledge Studio. and then retrain the model.

The following types of models are supported:

- Rule-based models created in Knowledge Studio that find entities in documents based on rules that you define. (File format: .pear)
- Machine learning models created in Knowledge Studio that understand the linguistic nuances, meaning, and relationships specific to your industry (file format: .zip)
- Custom entity extractors that are created in and exported from Discovery. (File format: .ent)
- Sentence classifiers that are created in and exported from Discovery. (File format: .sc)
- IBM Cloud Pak for Data Custom UIMA text analysis models created in Watson Explorer Content Analytics Studio. (File format: .pear)

From installed deployments, support for importing entity extractor models was added with the 4.6.2 release.

> **Important:** Discovery cannot identify entity subtypes that are defined by a Knowledge Studio model.

To add a Machine Learning model, complete the following steps:

1. Create the model and export it from the tool you use to create it.

   For more information, see the following documentation:

   - Knowledge Studio for IBM Cloud Pak® for Data
     - Creating a rule-based model
     - Creating a machine learning model
   - Knowledge Studio for IBM Cloud
     - Creating a rule-based model
     - Creating a machine learning model
   - Watson Explorer Content Analytics Studio

     You must export the model from Watson Explorer Content Analytics Studio as a UIMA PEAR file. For more information, see: Creating Custom PEAR Files for use with Lexical Analysis Streams.
   - Discovery entity extractor
2. From the *Teach domain concepts* section of the *Improvement tools* panel, and then click **Import machine learning models**.
3. Specify a name for the model, and then choose the language that was used to define the model.
4. Click **Upload** to browse for the file that you exported earlier.
5. Click **Create**.
6. Choose the collection and field where you want to apply the enrichments from the model, and then click **Apply**.

> **Note:** If the model is too large to upload from the product user interface, you can use the Create an enrichment method of the API to import the file.

Rule-based model example

For example, when a machine learning model is applied as an enrichment to a field, it extracts all entity types in that field that were specified in a Knowledge Studio rule-based model. If the model recognizes entity types such as `person`, `surname`, and `job title` they are recognized in your documents and tagged.

In the output, the information that is extracted by the Machine Learning enrichment in the `enriched_{field_name}` array, within the `entities` array. In this example, the field that is selected for enrichment is `text`.


```
{
   "enriched_text":  [
      {
         "entities":  [
            {
               "path":  ".wksrule.entities.PERSON",
               "text":  "George Washington",
               "type":  "PERSON"
            },
            {
               "path":  ".wksrule.entities.GIVENNAME",
               "text":  "George",
               "type":  "GIVENNAME"
            },
            {
               "path":  ".wksrule.entities.SURNAME",
               "text":  "Washington",
               "type":  "SURNAME"
            },
            {
               "path":  ".wksrule.entities.POSITION",
               "text":  "politician",
               "type":  "POSITION"
            },
            {
               "path":  ".wksrule.entities.POSITION",
               "text":  "soldier",
               "type":  "POSITION"
            },
            {
               "path":  ".wksrule.entities.JOBTITLE",
               "text":  "President of the United States",
               "type":  "JOBTITLE"
            }
         ],
         "text":  [
            "George Washington (February 22, 1732, December 14, 1799) was an American politician and soldier who served as the first President of the United States from 1789 to 1797 and was one of the Founding Fathers of the United States."
         ]
      }
   ]
}
```


As a result, if someone uses the API to submit a Discovery Query Language query to look for occurrences of the `enriched_{field_name}.entities.type:jobtitle` enrichment, any passages that discuss a person's job title are returned.

Machine learning model example

In this example, a Machine learning model extracts entity types such as `person`, `oranization`, and `date`, and information about relationships between the entities. When this ML model is applied as an enrichment to a field, it uses machine learning to understand the linguistic nuances, meaning, and relationships that are mentioned in the document.

In the output, the information that is extracted by the Machine Learning enrichment in the `enriched_{field_name}` array, within the `entities` and the `relations` arrays. In this example, the field that is selected for enrichment is `text`.


```
{
   "enriched_text":  [
      {
         "entities":  [
            {
               "count":  1,
               "text":  "Democratic Party",
               "type":  "ORGANIZATION"
            },
            {
               "count":  1,
```

```
                "text":  "March 15, 1767",
                "type":  "DATE"
            },
            {
                "count":  1,
                "text":  "President",
                "type":  "POSITION"
            },
            {
                "count":  1,
                "text":  "Andrew Jackson",
                "type":  "PERSON"
            }
        ],
        "relations":  [
            {
                "sentence":  "Andrew Jackson (March 15, 1767, June 8, 1845) was an American soldier and statesman who served as the seventh
President of the United States from 1829 to 1837 and was the founder of the Democratic Party."
            }
        ]
    }
  ]
}
```


Machine learning model limits

The number of Machine Learning (ML) models you can create per service instance depends on your Discovery plan type.

Plan	ML models per service instance
Cloud Pak for Data	Unlimited
Premium	10
Enterprise	10
Plus (includes Trial)	3

ML model plan limits

For each Knowledge Studio machine learning model, the maximum number of entities that can be detected is 50.

Advanced rules models

Add an advanced rules model to apply a text extraction model that was created and exported from the Advanced Rule editor of IBM Watson® Knowledge Studio to your collection.

Your model must be created with the appropriate Knowledge Studio deployment:

- IBM Cloud Pak for Data You can add models that were created and exported from the following places:
    - IBM Watson® Knowledge Studio that was built with a IBM Cloud Pak® for Data deployment earlier than the 4.5 release.
    - IBM Watson® Knowledge Studio that is hosted on IBM Cloud
    - NLP Editor that is built by contributors to the Center for Open-source Data & AI Technologies
- IBM Cloud You can add models that were created with a IBM Watson® Knowledge Studio instance that is hosted on IBM Cloud only.

Removal from Knowledge Studio

Support for building models with the beta Advanced Rules Editor in Knowledge Studio ended. Any rules models that were exported from Knowledge Studio prior to the end of support date can continue to be used in Discovery.

End of support dates differ based on the deployment type:

- IBM Cloud 30 June 2022
- IBM Cloud Pak for Data IBM Cloud Pak for Data release 4.5.1 on 3 August 2022.

IBM Cloud As an alternative to using a model that is generated by the Knowledge Studio Advanced Rules Editor, you can define a rule by adding a Patterns enrichment.

Adding an existing model

To add an advanced rule model, complete the following steps:

1. Create the model and export the ZIP file that contains the model resources.

   For more information about how to export the model, see the instructions for your model source:

   - Knowledge Studio for IBM Cloud Pak® for Data
   - Knowledge Studio for IBM Cloud
   - Open source NLP Editor

2. From the *Teach domain concepts* section of the *Improvement tools* panel, choose **Advanced rules model**.
3. Click **Upload**.
4. Specify a name for the model, and then choose the language that was used to define the model.
5. Specify a name for the result field, which is the field in the index where the output of this enrichment will be stored.
6. Click **Upload** to browse for the ZIP file that you exported earlier.
7. Click **Create**.
8. Choose the collection and field where you want to apply the enrichments from the model, and then click **Apply**.

Output format for advanced rules

Knowledge Studio uses the Annotation Query Language (AQL) to define the rules in an advanced rules model. Each model is defined by one or more views. Each view is a relational data structure that contains multiple data records. Each record is composed of values in columns that are defined by the view's schema. To facilitate representing these models, which are custom and therefore have various schemas, a uniform JSON output schema is used.

- Each JSON object represents an Annotation Query Language (AQL) view.
- The name-and-value pairs in the JSON objects represent the names and values of the attributes in the view.
- The tuples in an AQL view are represented as an array of JSON objects, with one object for each tuple in the view.

The following table describes how AQL data types are represented in JSON syntax.

AQL data type	JSON syntax	JSON example
Integer	number	`5`
Float	number	`4.13`
Boolean	boolean	`true`
Text	string	`"some string"`
Span	object with the form `{"text": String, "location": {"begin": Integer, "end": Integer}}`	`{ "text": "Jane", location": {"begin": 5, "end": 9} }`
Special case: null value	null	`null`
List of Integer	array of number values	`[ 1, 2, 3, 4, 5]`
List of Float	array of number values	`[ 4.13, 4.5 ]`
List of Boolean	array of boolean values	`[ true, true, false]`
List of Text	array of string values	`[ "some string", "another string" ]`

List of Span	array of objects with the form `{"text":String, "location": {"begin": Integer, "end": Integer}}`	`[{ "text":"Jane", "location": {"begin": 5, "end": 9} }, { "text":"...", "location": {"begin": 15, "end": 40} }]`
Special case: empty List	array with 0 elements	`[ ]`

Advanced rules model JSON output schema

Advanced rules model limits

The number of advanced rules models that you can define per service instance depends on your Discovery plan type.

Plan	Advanced rules models per service instance
Cloud Pak for Data	Unlimited
Premium	3
Enterprise	3
Plus (includes Trial)	1

Advanced rules model plan limits

Identify terms by pattern

Use patterns to find terms

Recognize terms that are mentioned in sentences that match a syntactic pattern that you teach Discovery to recognize.

IBM Cloud Patterns is a beta feature that is available in managed deployments only. The feature is available for English-language documents only.

Add a Patterns resource to teach Discovery to recognize patterns in your data. The Patterns feature uses pattern induction, which generates extraction patterns from examples that you provide as training data. After you specify a few examples, Discovery suggests more rules that you can review and accept to complete the pattern.

Patterns produces a model by using a human-in-the-loop process. You aren't asked to build a large set of training data up front. Instead, you provide a few examples, and then participate in an interactive process to define the training data. You passively accept or reject smart suggestions that are proposed by the system.

Pattern recognition works best on text with consistent structure in casing, length, text, or numeric values. Examples of patterns you can teach Discovery to identify in your documents:

- Standards numbers, such as `ISO 45001`, `ISO 22000`.
- Currency references, such as `$50.5 million`, `$29 million`.
- Date references, such as `8 September 2019`, `12 June 2020`.

If you need to identify specific terms or text, such as product names, add a dictionary.

For more information, read the following blog posts:

- Extracting Text Patterns with User Highlights with Pattern Induction
- Pattern Induction: Best Practices for Extracting Text Patterns

To define a pattern, complete the following steps:

1. From the *Teach domain concepts* section of the *Improvement tools* panel, choose **Patterns**.
2. Click **New**.
3. Pick how you want to choose documents.
    - Allow Discovery to choose 10 random documents for you.
    - Choose the documents yourself (up to 20 can be selected).

      Each document must be under 5,000 characters in length. Documents that exceed the limit are truncated to 5,000 characters.

4. Click **Next**.
5. Start selecting example words or phrases that fit the pattern you want to define.

   For example, if you have a collection of articles that discuss `ISO` standards, you might start highlighting the numbers of the standards in each document.

   If you annotate something, and then change your mind, hover over the selection, and then click `x` to delete it.
6. Continue selecting examples.

   After you identify enough examples, Discovery shows a list of suggested examples for you to review and determine to be valid or not valid examples. Suggested examples are taken from the field that is configured to be used in search results. If the source of result content is configured to be passages, the `text` field is used. For more information, see Changing the result content.
7. Choose **Yes** or **No** for each suggestion.

   Click the **Preview document** icon if you want to see the example in context before you make a choice.
8. Continue highlighting examples and validating suggestions until a message is displayed to inform you that you identified enough examples.
9. Click the **Review examples** tab to review the lists of examples that were identified by you and Discovery.
10. If the examples are correct, click **Save pattern**.

    Note: If Discovery cannot discern a consistent and valid pattern based on the information you provided, the **Save pattern** button is never enabled. A pattern might not be created if you provide contradictory examples, for example. To start over, click the **Reset** button. The documents are returned to their original state and any examples that you identified previously are removed.
11. To apply the pattern immediately, choose the collection and field where you want to apply the enrichments from the model, and then click **Apply**.

When Discovery finds text in a document that matches a pattern that you defined, it is annotated in the `enriched_{fieldname}.entities` field. You can find it by checking the `enriched_{fieldname}.entities.model_name` field for your pattern name.

Downloading a pattern

To download a pattern, complete the following step:

1. In the **Patterns view**, click the download icon.

   A pattern model is downloaded as a ZIP file.

You can import the downloaded ZIP file as the source for an advanced rules model resource. For more information, see Advanced rules models.

Pattern limits

The number of patterns that you can define per service instance depends on your Discovery plan type.

Plan	Patterns per service instance
Premium	100
Enterprise	100
Plus (includes Trial)	20

Pattern plan limits

Use regular expressions to find terms

Define regular expressions that capture patterns of significance, such as that `AB10045` is the syntax that is used for your order numbers.

Define regular expressions that can identify and extract information from fields in your collection.

For example, this regular expression finds occurrences of credit card numbers of a specific format and length.

```
4[0-9]{15}
```

The following regular expression finds occurrences of a US social security number.

```
(?!666|000|9\d{2})\d{3}-(?!00)\d{2}-(?!0{4})\d{4}
```

To add a regular expression, complete the following steps:

1. From the *Teach domain concepts* section of the *Improvement tools* panel, choose **Regular expression**.
2. Click **Upload**.
3. Optional: Specify a facet path to categorize any text that matches the regular expression. The text can be filtered by this facet later.

   If you use a hierarchy of categories, add a period between category names in the facet path to represent the hierarchy. For example, if you are adding a regex that can recognize phone numbers, you might have a facet path such as `international.europe`.
4. Add the regular expression.
   - Use a Java™ regular expression.

     For more information, see the Java documentation. Another useful resource is Regex 101.
   - Keep the regular expression as short and understandable as possible.
   - The best regular expressions resolve to a match or non-match quickly.
   - Use common patterns. For example, use `a(b|c|d)` instead of `(ab|ac|ad)`.
   - The regular expression engine might fail if it backtracks because it can't make a negative match toward the end of the string and then attempts too many permutations. To prevent backtracks, consider using a possessive quantifier, such as `(a+b*)++c`.
5. Click **Create**.
6. Choose the collection and field to search for occurrences of text that match this regular expression pattern.

In the output, the information that is extracted by the regular expression enrichment can be found under `enriched_{field_name}`, within the `entities` array.

In this example, the **Facet Path** is `regex.cccardnumber`, and the field that is selected for enrichment is `text`.

```
{
   "enriched_text":  [
      {
         "entities":  [
            {
               "path":  ".regex.cccardnumber",
               "type":  "cccardnumber",
               "text":  "4000000000000000"
            }
         ]
      }
   ],
   "text":  [
      "He has 2 phones, 090-1234-5678 and 080-1234-5678. His credit card number is 4000000000000000."
   ]
}
```

When you submit test queries from the *Improve and customize* page, you can add a facet that is based on the `enriched_text.entities.model_name` field. As a result, the `cccardnumber` regular expression enrichment that you created is displayed as a facet value by which documents can be filtered. For more information about creating facets, see Adding facets.

Regular expression limits

The number of regular expressions that you can define per service instance depends on your Discovery plan type.

Plan	Regular expressions per service instance
Cloud Pak for Data	Unlimited
Premium	100
Enterprise	100
Plus (includes Trial)	20

Regular expression plan details

Classify text

Define categories by which text in your documents can be classified.

This topic describes how to classify text. If you want to classify documents, use the Content Mining application. For more information, see Classifier types.

Add a text classifier to assign text from documents in your collection into categories. Discovery uses the labels and text examples that you provide to predict the categories of text in your collection.

To create a text classifier, complete the following steps:

1. Create a CSV file that contains example text followed by its category label per line.

   The CSV file must be in UTF-8 encoding format and must meet the following requirements:

   - The format must be `text,label`. The `text` is the example text, and the `label` is the category name.

     Add complete sentences as text entries. Do not include any blank lines in the CSV file.

     You can add more `label` columns if you need to apply more than one label to the sentence in the `text` column. For example, `text,label,label`.

   - The file must have at least two columns with no headers.
   - Add 10 or more entries for each category that you want to define. The minimum number of entries that are required per category is 3. The more examples that you provide for each category, the better the classifier can predict the categories of other content in your collection.

The following example is a CSV file that defines two categories, named `facility_temperature` and `catering`. The example text consists of feedback from conference attendees.

```
The rooms were too cold.,facility_temperature
Breakfast did not include gluten-free options.,catering
The rooms were too warm.,facility_temperature
I was very comfortable in the session rooms.,facility_temperature
The awards dinner was delicious.,catering
Coffee ran out during one of the breaks.,catering
The temperature was not comfortable.,facility_temperature
I was very happy with the selection at lunch.,catering
It was nice that you provided tea and coffee. Tea drinkers are often ignored.,catering
Can you turn up the air conditioning? I was very warm.,facility_temperature
My teeth were chattering because I was so cold.,facility_temperature
The speaker left the room to find someone to adjust the temperature.,facility_temperature
Would you consider an all-vegan menu next year?,catering
I would like lemonade and iced tea to be served during the breaks.,catering
The lunch staff was excellent.,catering
Appreciated the fresh blueberry muffins at breakfast.,catering
The hotel staff adjusted the temperature in my session room as soon as I asked. Excellent
service!,facility_temperature
Every meal was delicious and there was something for everyone.,catering
The seats under the skylights were not comfortable. Too hot.,facility_temperature
I was comfortable everywhere in the conference center. I never needed my emergency
sweater.,facility_temperature
```

2. From the *Teach domain concepts* section of the *Improvement tools* panel, and then click **Text classifiers**.
3. Click **Upload**.
4. Specify a name for the classifier, and then choose the language that was used in the CSV file.
5. Click **Upload** to browse for the CSV file that you created earlier.
6. Click **Create**.

   A classifier enrichment is created based on the training data that you provided.

7. Choose the collection and field where you want to apply the text classifier enrichment, and then click **Apply**.

The following example shows how an enrichment that is created with the sample CSV file as its training data might classify text in a document. In the output, the classifier enrichment applies the `facility_temperature` label to the document text. The `label` is stored in the `enriched_{field_name}` array, within the `classes` array.

```
{
   "enriched_text":  [
      {
         "classes":  [
            {
               "confidence":  0.999692440032959,
```

```
                "label":  "facility_temperature"
            }
          ]
       }
    ],
    "text":  [
       "I think more attendees would stay awake in the sessions if the rooms were colder."
    ]
}
```

Classifier types

The classifier that you add from the Discovery user interface is a *text classifier*. A text classifier can classify documents based on words and phrases that are extracted from the body text with their part of speech information taken into account.

You can create another classifier type, a *document classifier*, only from the deployed Content Mining application. A document classifier can classify documents based on words and phrases that are extracted from the body text fields with information from their part of speech and the other enrichments that are applied to the body text taken into account. The information from the other non-body fields are also used.

You can apply a document classifier to a collection in a project type other than a Content Mining project. To do so, you must create the classifier in the deployed Content Mining application and export it. You can then import the classifier and apply it to your collection as an enrichment. For more information, see Creating and applying a document classifier.

The text classifier uses Part of Speech information regardless of whether the Part of Speech enrichment is applied to the project.

Text classifiers that you add to one project can be used by other projects, including Content Mining projects.

Tip: A text classifier does not classify the target text field with confidence scores that are lower than 0.5. You cannot change the confidence threshold that is used by the text classifier. If you expected certain types of passages to be classified that weren't, you can add passages with similar characteristics to your training data and train another classifier.

Text classifier limits

The number of text classifiers and labels that you can create per service instance depends on your Discovery plan type.

Limit	Plus	Enterprise	Premium	Cloud Pak for Data
Number of text classifiers per service instance	5	20	20	Unlimited
Number of labeled data rows	2,000	20,000	20,000	20,000
Maximum size in MB of training data after enrichment	16	1,024	1,024	1,024
Number of labels	100	1,000	1,000	1,000

Text classifier plan limits

Detect sentiment

Use the built-in Watson Natural Language Processing (NLP) sentiment enrichment to analyze the sentiment that is expressed in text and indicate whether the text is `positive`, `neutral`, or `negative`.

To understand the sentiment of an entire document, apply this enrichment to a field that contains as much of the text from the document as possible, such as the `text` field.

To analyze sentiment in text from multiple fields at one time and capture the overall sentiment of the document, use the Content Mining application. For more information, see Detecting phrases that express sentiment.

Adding the enrichment

To add the sentiment enrichment, complete the following steps:

1. Open your project and go to the *Manage collections* page.
2. Click to open the collection that you want to enrich.
3. Open the **Enrichments** tab.

4. Scroll to find and select the Sentiment enrichment.
5. Choose one or more fields to apply the enrichment to.

   You can apply enrichments to the `text` and `html` fields, and to custom fields that were added from uploaded JSON or CSV files or from the Smart Document Understanding (SDU) tool.
6. Click **Apply changes and reprocess**.

Enrichments that you enable are applied to the documents in random order. For information about how to remove an enrichment, see Managing enrichments.

Example

Input

```
"It is powerful and easy to use and integrate with third party applications."
```

Response

In the JSON output:

- `score` = Sentiment score from `-1` (negative) to `1` (positive)
- `label` = `positive`, `negative`, or `neutral`
- `mixed` = Indicates that the document expresses a combination of different sentiments

```
{
   "sentiment":  {
      "score":  0.9255063900060722,
      "mixed":  false,
      "label":  "positive"
   }
}
```

Read contracts

The *Contracts* enrichment identifies contract-related elements in a document.

To use the Contracts enrichment, create a *Document Retrieval* project type and select the *Apply contracts enrichment* option. When you make this selection, a *Document Retrieval for Contracts* project is created.

Note: Only users of installed deployments (IBM Cloud Pak for Data) or Premium or Enterprise plan managed deployments can create a *Document Retrieval for Contract* project type.

To see the elements that are identified by the *Contracts* enrichment, complete the following steps:

1. From the *Improve and Customize* page, submit a search query.

   You can submit any keyword you want or pick one of the suggested keyword search terms.
2. Click **View passage in document** for one of the search results that are displayed for the document that you want to review.
3. Do one of the following things:

   IBM Cloud

   1. Click **Open advanced view** to see the *Contract Data* page.

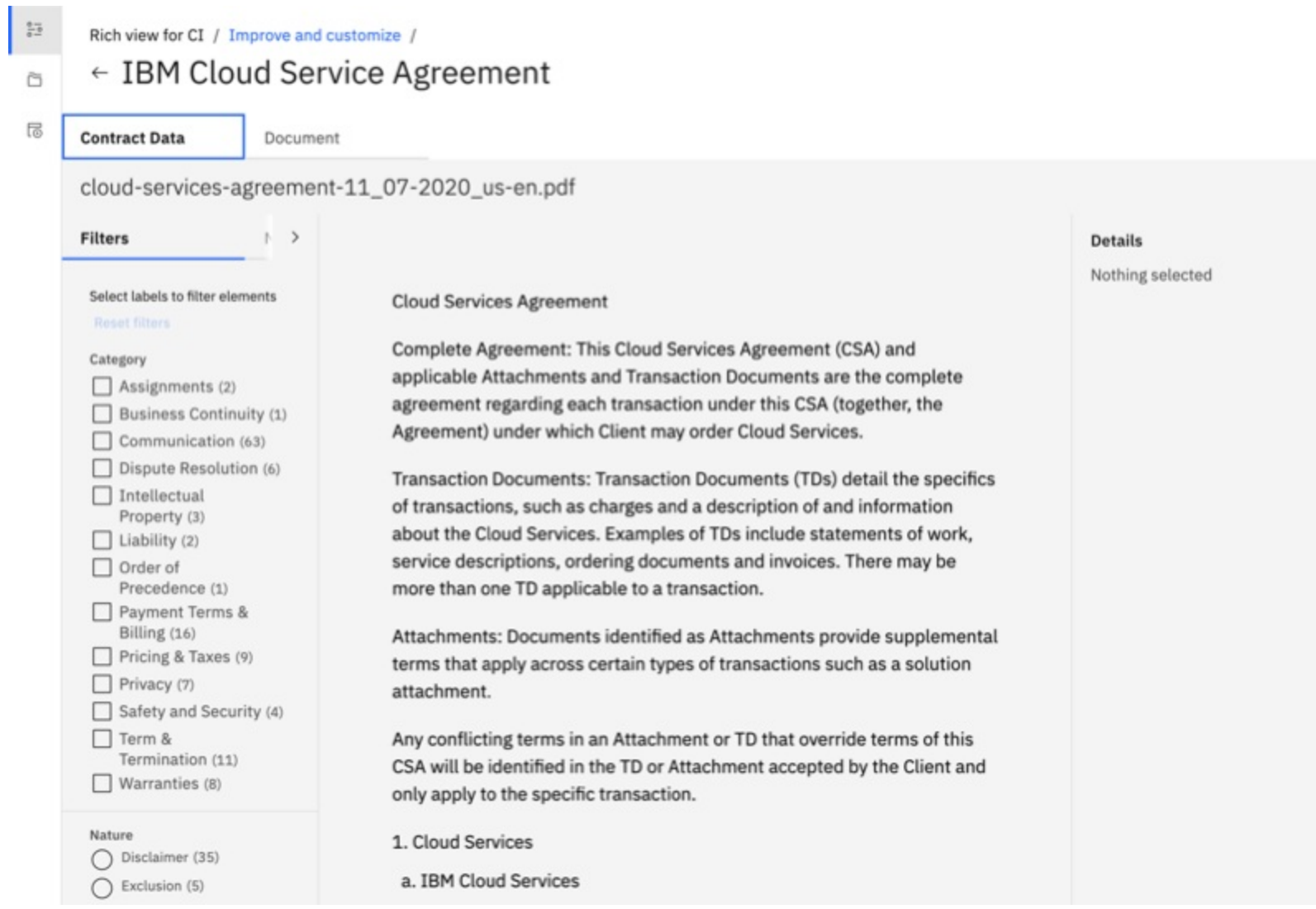


Figure 1. Contract Data view

IBM Cloud Pak for Data

1. Click **Contract Data**.

A list of the elements that the *Contracts* enrichment identified in the document is displayed.

Contract schema information

Contract enrichments are applied to the `html` field of documents that are added to the project.

After a document is processed by the Contracts enrichment, the service generates JSON output in the following schema:

```
{
   "elements": [
      {
         "location": {
            "begin": int,
            "end": int
         },
         "text": string,
         "types": [
            {
               "label": { "nature": string, "party": string },
               "provenance_ids": [string, string, ...]
                  ...
               ]
            }
            ...
      ],
      "categories": [
         {
            "label": string,
            "provenance_ids": [string, string, ...]
         }
         ...
      ],
      "attributes": [
         {
            "type": string,
            "text": string,
            "location": { "begin": int, "end": int }
          }
      ]
   }
   ...
],
"effective_dates": [
```

```
    {
      "confidence_level": string,
      "text": string,
      "text_normalized": string,
      "provenance_ids": [ string, string, ... ],
      "location": { "begin": int, "end": int }
    },
    ...
  ],
  "contract_amounts": [
    {
      "confidence_level": string,
      "text": string,
      "text_normalized": string,
      "interpretation": {
        "value": string,
        "numeric_value": number,
        "unit": string,
      },
      "provenance_ids": [ string, string, ... ],
      "location": { "begin": int, "end": int }
    },
    ...
  ],
  "termination_dates": [
    {
      "confidence_level": string,
      "text": string,
      "text_normalized": string,
      "provenance_ids": [ string, string, ... ],
      "location": { "begin": int, "end": int }
    },
    ...
  ],
  "contract_types": [
    {
      "confidence_level": string,
      "text": string,
      "provenance_ids": [ string, string, ... ],
      "location": { "begin": int, "end": int }
    },
    ...
  ],
  "contract_terms": [
    {
      "confidence_level": string,
      "text": string,
      "text_normalized": string,
      "interpretation": {
        "value": string,
        "numeric_value": number,
        "unit": string,
      },
      "provenance_ids": [ string, string, ... ],
      "location": { "begin": int, "end": int }
    },
    ...
  ],
  "payment_terms": [
    {
      "confidence_level": string,
      "text": string,
      "text_normalized": string,
      "interpretation": {
        "value": string,
        "numeric_value": number,
        "unit": string,
      },
      "provenance_ids": [ string, string, ... ],
      "location": { "begin": int, "end": int }
    },
    ...
  ],
  "contract_currencies": [
    {
      "confidence_level" : string,
      "text" : string,
      "text_normalized" : string,
```

```
        "provenance_ids": [string, string ..],
        "location": { "begin": int, "end": int }
      },
    ...
],
    "tables": [],
    "document_structure": {
      "section_titles": [
        {
          "text": string,
          "location": {
            "begin": int,
            "end": int
          },
          "level": int,
          "element_locations": [
            {
              "begin": int,
              "end": int
            },
            ...
          ]
        },
        ...
      ],
    "leading_sentences": [
      {
        "text": string,
        "location": {
          "begin": int,
          "end": int
        },
        "element_locations": [
          {
            "begin": int,
            "end": int
          },
          ...
        ],
    "paragraphs": [
      {
        "location": {
            "begin": int,
            "end": int
         }
      },
      ...
    ]
      },
      ...
    ]
},
"parties": [
  {
    "party": string,
    "role": string,
    "importance": string,
    "addresses": [
      {
        "text": string,
        "location": {
          "begin": int,
          "end": int
        }
      },
      ...
    ],
    "contacts": [
      {
        "name": string,
        "role": string
      },
      ...
    ],
    "mentions": [
      {
        "text": string,
        "location": {
          "begin": int,
```

```
              "end": int
            }
          },
          ...
        ]
      },
      ...
    ]
}
```

Schema arrangement

The `contracts` schema is arranged as follows.

- `elements`: An array of the document elements detected by the service.
  - `location`: An object that identifies the location of the element. The object contains two index numbers, `begin` and `end`. The index numbers indicate the beginning and ending positions of the characters that constitute the element in HTML.
  - `text`: The text of the element.
  - `types`: An array that describes what the element is and whom it affects.
    - `label`: An object that defines the type by using a pair of the following elements:
      - `nature`: The type of action the sentence requires. Current values are `Definition`, `Disclaimer`, `Exclusion`, `Obligation`, and `Right`.
      - `party`: A string that identifies the party to whom the sentence applies.
    - `provenance_ids`: An array of one or more hashed values that you can send to IBM to provide feedback or receive support.
  - `categories`: An array that lists the functional categories into which the element falls; in other words, the subject matter of the element.
    - `label`: A string that lists the identified category. For a list of categories, see Categories.
    - `provenance_ids`: An array of one or more hashed values that you can send to IBM to provide feedback or receive support.
  - `attributes`: An array that identifies document attributes. Each object in the array consists of three elements:
    - `type`: The type of attribute. Possible values are `Currency`, `DateTime`, `Duration`, `Location`, `Number`, `Organization`, `Percentage`, and `Person` as described at Attributes.
    - `text`: The text that is associated with the attribute.
    - `location`: The location of the attribute as defined by its `begin` and `end` indexes.
- `effective_dates`: An array that identifies the date or dates on which the document becomes effective.
  - `confidence_level`: The confidence level of the identification of the effective date. Possible values include `High`, `Medium`, and `Low`.
  - `text`: An effective date, which is listed as a string.
  - `text_normalized`: The normalized form of the effective date, which is listed as a string. This element is optional; that is, the service output lists it only if normalized text exists.
  - `location`: The location of the date as defined by its `begin` and `end` indexes.
  - `provenance_ids`: An array that contains zero or more keys. Each key is a hashed value that you can send to IBM to provide feedback or receive support.
- `contract_amounts`: An array that monetary amounts that identify the total amount of the contract that needs to be paid from one party to another.
  - `confidence_level`: The confidence level of the identification of the contract amount. Possible values include `High`, `Medium`, and `Low`.
  - `text`: A contract amount, which is listed as a string.
  - `location`: The location of the amount or amounts as defined by its `begin` and `end` indexes.
  - `provenance_ids`: An array that contains zero or more keys. Each key is a hashed value that you can send to IBM to provide feedback or receive support.
- `termination_dates`: An array that identifies the date or dates on which the document is to be terminated.
  - `confidence_level`: The confidence level of the identification of the termination date. Possible values include `High`, `Medium`, and `Low`.
  - `text`: A termination date, which is listed as a string.
  - `text_normalized`: The normalized form of the termination date, which is listed as a string. This element is optional; that is, the service output lists it only if normalized text exists.
  - `location`: The location of the date as defined by its `begin` and `end` indexes.
  - `provenance_ids`: An array that contains zero or more keys. Each key is a hashed value that you can send to IBM to provide feedback or receive support.
- `contract_types`: An array that identifies the document's contract type or types.
  - `confidence_level`: The confidence level of the identification of the contract type. Possible values include `High`, `Medium`, and `Low`.

    - `text`: A contract type, which is listed as a string.
    - `provenance_ids`: An array that contains zero or more keys. Each key is a hashed value that you can send to IBM to provide feedback or receive support.
    - `location`: The location of the contract type as defined by its `begin` and `end` indexes.
- `contract_terms`: An array that identifies the duration or durations of the contract.
    - `confidence_level`: The confidence level of the identification of the contract terms. Possible values include `High`, `Medium`, and `Low`.
    - `text`: A contract term, which is listed as a string.
    - `provenance_ids`: An array that contains zero or more keys. Each key is a hashed value that you can send to IBM to provide feedback or receive support.
    - `location`: The location of the contract term as defined by its `begin` and `end` indexes.
- `payment_terms`: An array that identifies the document's payment duration or durations.
    - `confidence_level`: The confidence level of the identification of the payment term. Possible values include `High`, `Medium`, and `Low`.
    - `text`: A payment term, which is listed as a string.
    - `text_normalized`: The normalized text, if applicable.
    - `interpretation`: The details of the normalized text, if applicable.
        - `value`: A string that lists the value that was found in the normalized text.
        - `numeric_value`: An integer or double that expresses the numeric value of the `value` key.
        - `unit`: A string that lists the unit of the value that was found in the normalized text.

          The value of `unit` is the ISO-4217 currency code that is identified for the currency amount (for example, `USD` or `EUR`). If the service cannot disambiguate a currency symbol (for example, `$` or `£`), the ambiguous symbol itself is stored as the `unit` value.
    - `provenance_ids`: An array that contains zero or more keys. Each key is a hashed value that you can send to IBM to provide feedback or receive support.
    - `location`: The location of the contract term as defined by its `begin` and `end` indexes.
- `contract_currencies`: An array that identifies the document's contract currency or currencies.
    - `confidence_level`: The confidence level of the identification of the contract currency. Possible values include `High`, `Medium`, and `Low`.
    - `text`: A contract currency, which is listed as a string.
    - `text_normalized`: The normalized text, if applicable. It is listed as a string in ISO-4217 format
    - `provenance_ids`: An array that contains zero or more keys. Each key is a hashed value that you can send to IBM to provide feedback or receive support.
    - `location`: The location of the contract currency as defined by its `begin` and `end` indexes.
- `document_structure`: An object that describes the structure of the input document.
    - `section_titles`: An array that contains one object per section or subsection that is detected in the input document. Sections and subsections are not nested. Instead, they are flattened out and can be placed back in order by using the `begin` and `end` values of the element and the `level` value of the section.
        - `text`: A string that lists the section title, if detected.
        - `location`: The location of the title in the input document as defined by its `begin` and `end` indexes.
        - `level`: An integer that indicates the level at which the section is located in the input document. For example, represents a root-level section, represents a subsection within the level section.
        - `element_locations`: An array that specifies the `begin` and `end` values of the sentences in the section.
    - `leading_sentences`: An array that contains one object per leading sentence of a list or subsection, in parallel with the `section_titles` and `paragraph` arrays. The object details the leading sentences in the matching section or subsection. As in the `section_titles` array, the objects are not nested. Instead, they are flattened out and can be placed back in order by using the `begin` and `end` values of the element or any level markers in the input document.
        - `text`: A string that lists the leading sentence, if detected.
        - `location`: The location of the leading sentence in the input document as defined by its `begin` and `end` indexes.
        - `element_locations`: An array that specifies the `begin` and `end` values of the leading sentences in the section.
    - `paragraphs`: An array that contains one object per paragraph, in parallel with the `section_titles` and `leading_sentences` arrays. Each object lists the span (beginning and end location) of the corresponding paragraph.
        - `location`: The location of the paragraph in the input document as defined by its `begin` and `end` indexes.
- `parties`: An array that defines the parties that are identified by the service.
    - `party`: A string that provides the normalized form of the party's name.

- `role`: A string that identifies the role of the party.
- `importance`: A string that identifies the importance of the party. Possible values include `Primary` for a primary party and `Unknown` for a non-primary party.
- `addresses`: An array of objects that identify addresses.
  - `text`: A string that contains the address.
  - `location`: The location of the address as defined by its `begin` and `end` indexes.
- `contacts`: An array that defines the name and role of contacts that are identified in the input document.
  - `name`: A string that lists the name of an identified contact.
  - `role`: A string that lists the role of the identified contact.
- `mentions`: An array of objects that identify mentions of the party.
  - `text`: A string that lists the name of the party.
  - `location`: The location of the mention as defined by its `begin` and `end` indexes.

`location` object

The `location` object is included with most of element definitions. The object identifies the location of the text string or number that represents the element. The object contains two index numbers, `begin` and `end`. The index numbers indicate the beginning and ending positions of the characters in the mention.

For example, a `text` string with the value `Amount due` might have a corresponding `location` object that looks as follows:

```
{
   ...
   "location": {
      "begin": 2510,
      "end": 2519
   }
   ...
}
```

The `begin` index indicates that the string begins at character position `2510` in the transformed HTML, which is the location of the letter `A` in `Amount`. The `end` index indicates that the string ends at character position `2519`, which is the location of the letter `e` in `due`.

Document structure

The output of the `Contracts` enrichment includes a `document_structure` object that details the structural composition of the input document. The document structure information is represented in the following JSON sample. The object is located immediately after the root-level `tables` array.

```
"document_structure": {
   "section_titles": [
      {
         "text": string,
         "location": {
            "begin": int,
            "end": int
         },
         "level": int
         "element_locations": [
            {
               "begin": int,
               "end": int
            },
            ...
         ]
   },
   ...
],
"leading_sentences": [
   {
      "text": string,
      "location": {
         "begin": int,
         "end": int
      },
      "element_locations": [
         {
            "begin": int,
            "end": int
```

```
            },
            ...
         ]
      },
      ...
   ],
   "paragraphs": [
      {
         "location": {
            "begin": int,
            "end": int
         }
      },
      ...
   ]
}
```

Document structure elements

The elements of the `document_structure` object contain the following information:

- `document_structure`: An object that describes the structure of the input document.
  - `section_titles`: An array that contains one object per section or subsection that is detected in the input document. Sections and subsections are not nested. Instead, they are flattened out and can be placed back in order by using the `begin` and `end` values of the element and the `level` value of the section.
    - `text`: A string that lists the section title, if detected.
    - `location`: The location of the title in the input document as defined by its `begin` and `end` indexes.
    - `level`: An integer that indicates the level at which the section is located in the input document. For example, `1` represents a root-level section, `2` represents a subsection within the level `1` section, and so on.
    - `element_locations`: An array that contains objects that specify the `begin` and `end` values of the sentences in the section.
  - `leading_sentences`: An array that contains one object per leading sentence of a list or subsection, in parallel with the `section_titles` and `paragraph` arrays. The object details the leading sentences in the matching section or subsection. As in the `section_titles` array, the objects are not nested. Instead, they are flattened out and can be placed back in order by using the `begin` and `end` values of the element or any level markers in the input document.
    - `text`: A string that lists the leading sentence, if detected.
    - `location`: The location of the leading sentence in the input document as defined by its `begin` and `end` indexes.
    - `element_locations`: An array that contains objects that specify the `begin` and `end` values of the leading sentences in the section.
  - `paragraphs`: An array that contains one object per paragraph, in parallel with the `section_titles` and `leading_sentences` arrays. Each object lists the span (beginning and end location) of the corresponding paragraph.
    - `location`: The location of the paragraph in the input document as defined by its `begin` and `end` indexes.

Elements

The Contract enrichment generates an analysis of each identified element in the contract. The following sections describe each type of element that is generated.

Types

The `types` array includes a number of objects, each of which contains `nature` and `party` keys whose values identify a couplet for the element.

The following tables list the possible values of the `nature` and `party` keys.

`nature`	Description
`Definition`	This element adds clarity for a term, relationship, or similar. No action is required to fulfill the element, nor is any party affected.
`Disclaimer`	The `party` in the element is not obligated to fulfill the terms that are specified by the element but is not prohibited from doing so.
`Exclusion`	The `party` in the element will not fulfill the terms that are specified by the element.
`Obligation`	The `party` in the element is required to fulfill the terms specified by the element.
`Right`	The `party` in the element is guaranteed to receive the terms specified by the element.

Supported keys

Each `nature` key is paired with a `party` key, which contains either the name or the role of the party or parties that apply to the nature (examples include, but are not limited to, `Buyer`, `IBM`, or `All Parties`). For the `Definition` nature, the party is always `None`.

Parties

The `parties` array specifies the participants that are listed in the contract. Each `party` object is associated with other objects that provide details about the party, including:

- `role`: The party's role. Values are listed in the table that follows this list.
- `importance`: The importance of the party. Possible values are `Primary` for a primary party and `Unknown` for a non-primary party.
- `addresses`: An array that identifies addresses.
  - `text`: An address.
  - `location`: The location of the address as defined by its `begin` and `end` indexes.
- `contacts`: An array that defines the names and roles of contacts that are identified in the input document.
  - `name`: The name of a contact.
  - `role`: The role of the contact.
- `mentions`: n array of objects that identify mentions of the party.
  - `text`: A string that lists the name of the party.
  - `location`: The location of the mention as defined by its `begin` and `end` indexes.

The values of `role` that can be returned for contracts include, but are not limited to:

`role`	Description
`Buyer`	The party responsible for paying for the goods or services that are listed in the contract.
`End User`	The party who interacts with the provided goods or services, explicitly distinguished from the `Buyer`.
`None`	No party was identified for the element.
`Supplier`	The party responsible for providing the goods or services that are listed in the contract.

Supported role values

Categories

The `categories` array defines the subject matter of the sentence. Currently, supported categories include:

`categories`	Description
`Amendments`	Elements that specify changes to the contract after it was signed, or alterations to a standard contract. Includes discussions of the conditions for changing the terms of a contract.
`Asset Use`	Elements that refer to how one party may or may not use the assets of another party. This category specifically applies to one party having access to or using assets such as licenses, equipment, tools, or personnel of the other party while conducting their duties under the agreement, including permissions and restrictions thereon. This category does not extend to specifications of a party's obligations or rights regarding any purchased goods, services, licenses, and so on, as those goods are the party's own assets, rather than assets of another party.
`Assignments`	Elements that describe the transfer of rights, obligations, or both to a third party.
`Audits`	Elements referring to either the right of a party to examine or review compliance, or requirements that a party be available for inspection or compliance review. This category includes references to record keeping (primarily as it relates to the right of inspection) and the maintenance and retention of activity records that may be examined.
`Business Continuity`	Elements referring to the consequences if the entire business of one of the parties is sold.

`Communication`	Elements referring to requirements to communicate, respond, notify, or provide notice; contact information; or information regarding changes to the contract. Also includes references to details about communication methods, the act or process of exchanging information, and acceptable means of exchanging information between parties (and others who are not necessarily direct parties to the contract).
`Confidentiality`	Elements describing how parties can or cannot use information that is learned in the course of completing the contract and going forward. Also includes discussion of information that must be kept confidential, such as maintaining trade secrets or nondisclosure of business information.
`Deliverables`	Elements specifying the items, such as goods or services, that one party provides to another under the terms of the contract, usually in exchange for payment. Includes discussion of preparation of deliverables.
`Delivery`	Elements that specify the means or modes of transferring deliverables (things, as opposed to personal services) from one party to another. Includes discussions of characteristics of delivery, such as scheduling or location.
`Dispute Resolution`	Elements discussing provisions for settling any dispute (for example, regarding labor, invoices, or billing) arising between contracting parties. Provision examples may include settlement by a defined procedure such as an arbitration panel, a process for obtaining an injunction, waiving a right to trial, or prohibiting the pursuit of a class action. Also includes references to the contract's governing law or choice of law, such as a particular country or jurisdiction.
`Force Majeure`	Elements that refer to unexpected or disruptive events outside a party's control that would relieve the party from performing their contractual obligation.
`Indemnification`	Elements that specify the remediation of certain liabilities, when one party of the contract becomes responsible for compensating another party as a result of incurred loss or damages during the term of, or arising from the circumstances of the contract. Also includes references to any legal exemptions from loss or damages.
`Insurance`	Elements referring to insurance coverage or terms of coverage that is provided by one party to another party (including to third parties such as subcontractors or others). Includes various types of insurance including, but not limited to, medical insurance.
`Intellectual Property`	Elements that discuss the assignment of rights (such as copyrights, patents, and trade secrets) to parties to the contract. Includes references to patents, rights to apply for patents, trademarks, trade names, service marks, domain names, copyrights, and all applications and registration of such schematics, industrial models, inventions, authorship, know-how, trade secrets, computer software programs, and other intangible proprietary information. Also includes discussion of the consequences of violation of intellectual property rights.
`Liability`	Elements that describe the method for determining when and how fault attaches to any party. Examples may include, but are not limited to, statements regarding limitations of liability, third-party claims, and repairs, replacements, or reimbursements as required of the party at fault.
`Payment Terms & Billing`	Elements that detail how and when a party is to pay or get paid, and the items or fees the parties will be paying or billed for. Includes references to modes of payment or payment mechanisms.
`Pricing & Taxes`	Elements that refer to specific amounts or figures that are associated with individual deliverables that are exchanged (for example, how much something costs) as part of satisfying the terms of the contract. Includes references to specific figures or methods for calculating prices or tax amounts.
`Privacy`	Elements that are particularly concerned with the treatment of sensitive personal information, usually regarding its protection (for example, to satisfy regulations such as GDPR).
`Responsibilities`	Elements that discuss tasks ancillary to the contract that are in only one party's control and are focused on discussion of employee oversight.
`Safety and Security`	Elements referring to physical safety or cybersecurity protection for people, data, or systems. Examples include discussions of background checks, safety precautions, workplace security, secure access protocols, and product defects that might pose a danger.
`Scope of Work`	Elements that define what is in the contract versus is not in the contract; consequently, what is promised to be done. Examples include statements that define an order, or describe the goals or aims outlined in the contract.

`Subcontracts`	Elements referring to the hiring of third parties to perform certain duties under the contract, and the permissions, rights, restrictions, and consequences thereto and arising therefrom.
`Term & Termination`	Elements referring to duration of the contract, the schedule and terms of contract termination, and any consequences of termination, including any obligations that apply at or after termination.
`Warranties`	Elements that refer to ongoing promises and obligations that are made in the contract that are currently true and will continue to be true in the future. Also, elements that discuss the consequences of such promises or obligations that are broken, and the rights to remedy the situation (for example, but not limited to, seeking damages). This category does not apply to elements that are concerned with representation statements (statements of fact about the past or the present), or to elements that lay out assumptions about things that occurred in the past.

Supported categories

Attributes

The `attributes` array specifies any attributes that are identified in the sentence. Each object in the array includes three keys: `type` (the type of attribute from the following table), `text` (the applicable text), and `location` (the `begin` and `end` indexes of the attribute in the input document). Currently, supported attributes include:

`attributes`	Description
`Currency`	Monetary value and units.
`DateTime`	A date, time, date range, or time range.
`DefinedTerm`	A term that is defined in the input document.
`Duration`	A time duration.
`Location`	A geographical location or region.
`Number`	A digital or textual number that describes a quantity of countable things and is not classified as one of the other numerical `attribute` types.
`Organization`	An organization.
`Percentage`	A percentage.
`Person`	A person.

Supported attributes

Effective dates

The `effective_dates` array identifies the date or dates on which the document becomes effective.

`effective_dates`	Description
`confidence_level`	The confidence level of the identification of the effective date. Possible values include `High`, `Medium`, and `Low`.
`text`	An effective date, listed as a string.
`text_normalized`	The normalized text of the `text` if available, listed as a string.
`location`	The location of the date as defined by its `begin` and `end` indexes.
`provenance_ids`	An array of hashed values that you can send to IBM to provide feedback or receive support.

Effective date values

Contract amounts

The `contract_amounts` array identifies the monetary amounts specified in the document.

`contract_amounts`	Description
`confidence_level`	The confidence level of the identification of the contract amount. Possible values include `High`, `Medium`, and `Low`.
`text`	A contract amount, listed as a string.
`normalized_text`	The normalized text, if applicable.
`interpretation`	The details of the normalized text, if applicable.
`value`	A string that lists the value that was found in the normalized text.
`numeric_value`	An integer or float expressing the numeric value of the `value` key.
`unit`	A string that lists the unit of the value that was found in the normalized text.
`location`	The location of the contract amount as defined by its `begin` and `end` indexes.
`provenance_ids`	An array of hashed values that you can send to IBM to provide feedback or receive support.

Amount values

Termination dates

The `termination_dates` array identifies the document's termination dates.

`termination_dates`	Description
`confidence_level`	The confidence level of the identification of the termination date. Possible values include `High`, `Medium`, and `Low`.
`text`	The termination date, listed as a string.
`text_normalized`	The normalized text of the `text` if available, listed as a string.
`location`	The location of the termination date as defined by its `begin` and `end` indexes.
`provenance_ids`	An array of hashed values that you can send to IBM to provide feedback or receive support.

Termination datae values

Contract types

The `contract_types` array identifies the document's contract type or types as declared in the document.

`contract_types`	Description
`confidence_level`	The confidence level of the identification of the contract type. Possible values include `High`, `Medium`, and `Low`.
`text`	The contract type, listed as a string.
`location`	The location of the contract type as defined by its `begin` and `end` indexes.
`provenance_ids`	An array of hashed values that you can send to IBM to provide feedback or receive support.

Contract type values

Contract terms

The `contract_terms` array identifies the duration or durations of the contract as declared in the document.

`contract_terms`	Description
`confidence_level`	The confidence level of the identification of the contract term. Possible values include `High`, `Medium`, and `Low`.
`text`	The contract term, listed as a string.
`normalized_text`	The normalized text, if applicable.
`interpretation`	The details of the normalized text, if applicable.
`value`	A string that lists the value that was found in the normalized text.
`numeric_value`	An integer or float expressing the numeric value of the `value` key.
`unit`	A string that lists the unit of the value that was found in the normalized text.
`location`	The location of the contract term as defined by its `begin` and `end` indexes.
`provenance_ids`	An array of hashed values that you can send to IBM to provide feedback or receive support.

Contract term values

Payment terms

The `payment_terms` array identifies the payment duration or durations as declared in the document.

`payment_terms`	Description
`confidence_level`	The confidence level of the identification of the payment term. Possible values include `High`, `Medium`, and `Low`.
`text`	The payment term, listed as a string.
`normalized_text`	The normalized text, if applicable.
`interpretation`	The details of the normalized text, if applicable.
`value`	A string that lists the value that was found in the normalized text.
`numeric_value`	An integer or float expressing the numeric value of the `value` key.
`unit`	A string that lists the unit of the value that was found in the normalized text.
`location`	The location of the payment term as defined by its `begin` and `end` indexes.
`provenance_ids`	An array of hashed values that you can send to IBM to provide feedback or receive support.

Payment term values

Contract currencies

The `contract_currencies` array identifies the contract currency or currencies as declared in the document.

`contract_currencies`	Description
`confidence_level`	The confidence level of the identification of the contract currency. Possible values include `High`, `Medium`, and `Low`.
`text`	The contract currency, listed as a string.

`text_normalized`	The normalized text of the `text` if applicable, listed as a string in ISO-4217 format.
`location`	The location of the contract currency as defined by its `begin` and `end` indexes.
`provenance_ids`	An array of hashed values that you can send to IBM to provide feedback or receive support.

Contract currency values

Provenance

Each object in the `types` and `categories` arrays includes a `provenance_ids` array. The `provenance_ids` array has one or more keys. Each key is a hashed value that you can send to IBM to provide feedback or receive support.

Understand tables

Apply the *Table Understanding* enrichment to get detailed information about tables and table-related data within documents.

The following tasks generate an HTML field with table information and apply the Table Understanding enrichment to it for your collection automatically:

- If you use the Smart Document Understanding tool to define a user-trained or pretrained SDU model, the `Table Understanding` enrichment is applied to the `html` field that is generated for the collection.
- If you create a *Document Retrieval for Contracts* project type, a pretrained SDU model is applied to your collection automatically. As a result, the `Table Understanding` enrichment is applied to the `html` field that is generated for the collection.

  For more information, see Smart Document Understanding.

Before you begin

The documents in your collection must contain a field with HTML representations of your tables. This information often is stored in the `html` field. If your collection consists of CSV or JSON files, it might have a field other than the `html` field that contains table information in HTML format.

Applying the table understanding enrichment

You can apply the enrichment only to a field that contains an HTML representation of the table.

To apply the enrichment, complete the following steps:

1. From the navigation pane, open the **Manage collections** page, and then click a collection to open it.
2. Click the **Enrichments** tab.
3. Find the **Table Understanding** enrichment.
4. Select the `html` field from the field list.

   Choose the field that contains HTML representations of the tables.

After the enrichment is applied, you can get valid results when you submit queries that require Discovery to find information that is stored in tables.

A developer can query tables by using the API. For more information, see Query parameters.

For more information about how to apply the table understanding enrichment by using the API, see Applying enrichments by using the API.

Working with tabular data in Python

Use Text Extensions for Pandas, an open-source library from IBM, to read the tables that were parsed from documents in Discovery into pandas DataFrame objects. A pandas DataFrame is an object that represents two-dimensional tabular data in a form that can be transformed and manipulated for downstream analysis in Python.

For example, you can extract content from tables in many annual report documents and reconstruct it into a single table that includes multiyear data points of interest. For more information, read the Structured Information Extraction from Tables in PDF Documents with Pandas and IBM Watson blog post on Medium.com.

Output schema

The output schema from the `Table Understanding` enrichment is as follows.

```
{
   "tables": [
```

```
{
  "location" : {
    "begin" : int,
    "end" : int
  },
  "text": string,
  "section_title": {
    "text": string,
    "location": {
      "begin" : int,
      "end" : int
    }
  },
  "title": {
    "location": {
      "begin": int,
      "end": int,
    },
    "text": string
  },
  "table_headers" : [
    {
      "cell_id" : string,
      "location" : {
        "begin" : int,
        "end" : int
      },
      "text" : string,
      "row_index_begin" : int,
      "row_index_end" : int,
      "column_index_begin" : int,
      "column_index_end" : int
    },
    ...
  ],
  "column_headers" : [
    {
      "cell_id" : string,
      "location" : {
        "begin" : int,
        "end" : int
      },
      "text" : string,
      "text_normalized" : string,
      "row_index_begin" : int,
      "row_index_end" : int,
      "column_index_begin" : int,
      "column_index_end" : int
    },
    ...
  ],
  "row_headers" : [
    {
      "cell_id" : string,
      "location" : {
        "begin" : int,
        "end" : int
      },
      "text" : string,
      "text_normalized" : string,
      "row_index_begin" : int,
      "row_index_end" : int,
      "column_index_begin" : int,
      "column_index_end" : int
    },
    ...
  ],
  "body_cells" : [
    {
      "cell_id" : string,
      "location" : {
        "begin" : int,
        "end" : int
      },
      "text" : string,
      "row_index_begin" : int,
      "row_index_end" : int,
      "column_index_begin" : int,
```

```
            "column_index_end" : int,
            "row_header_ids": [ string ],
            "row_header_texts": [ string ],
            "row_header_texts_normalized": [ string ],
            "column_header_ids": [ string ],
            "column_header_texts": [ string ],
            "column_header_texts_normalized": [ string ],
            "attributes" : [
               {
                  "type" : string,
                  "text" : string,
                  "location" : {
                     "begin" : int,
                     "end" : int
                  }
               },
               ...
             ]
         },
         ...
      ],
      "key_value_pairs": [
         {
            "key": {
               "cell_id": string,
               "location": {
                  "begin": int,
                  "end": int
               },
               "text": string
            },
            "value": [{
               "cell_id": string,
               "location": {
                  "begin": int,
                  "end": int
               },
               "text": string
            },
            ...
            ]
         },
         ...
      ],
      "contexts": [
         {
            "text": string,
            "location": {
               "begin": int,
               "end": int
            }
         },
         ...
      ]
   }
```

Schema arrangement

The schema is arranged as follows.

- `tables`: An array that defines the tables that are identified in the input document.
  - `location`: The location of the current table as defined by its `begin` and `end` indexes in the input document.
  - `text`: The textual contents of the current table from the input document without associated markup content.
  - `section_title`: If identified, the location of a section title contained in the current table. Empty if no section title is identified.
    - `text`: The text of the identified section title.
    - `location`: The location of the section title in the input document as defined by its `begin` and `end` indexes.
  - `title`: If identified, the title or caption of the current table of the form `Table x.: ...`. Empty when no title is identified. When present, the `title` is excluded from the `contexts` array of the same table.
    - `location`: The location of the title in the input document as defined by its `begin` and `end` indexes.

        - `text`: The text of the identified table title or caption.
    - `table_headers`: An array of table-level cells applicable as headers to all the other cells of the current table. Each table header is defined as a collection of the following elements:
        - `cell_id`: The unique ID of the cell in the current table.
        - `location`: The location of the cell in the input document as defined by its `begin` and `end` indexes.
        - `text`: The textual contents of the cell from the input document without associated markup content.
        - `row_index_begin`: The `begin` index of the cell's `row` location in the current table.
        - `row_index_end`: The `end` index of the cell's `row` location in the current table.
        - `column_index_begin`: The `begin` index of the cell's `column` location in the current table.
        - `column_index_end`: The `end` index of the cell's `column` location in the current table.
    - `column_headers`: An array of column-level cells, each applicable as a header to other cells in the same column as itself, of the current table. Each column header is defined as a collection of the following items:
        - `cell_id`: The unique ID of the cell in the current table.
        - `location`: The location of the cell in the input document as defined by its `begin` and `end` indexes.
        - `text`: The textual contents of the cell from the input document without associated markup content.
        - `text_normalized`: Normalized column header text.
        - `row_index_begin`: The `begin` index of the cell's `row` location in the current table.
        - `row_index_end`: The `end` index of the cell's `row` location in the current table.
        - `column_index_begin`: The `begin` index of the cell's `column` location in the current table.
        - `column_index_end`: The `end` index of the cell's `column` location in the current table.
    - `row_headers`: An array of row-level cells, each applicable as a header to other cells in the same row as itself, of the current table. Each row header is defined as a collection of the following items:
        - `cell_id`: The unique ID of the cell in the current table.
        - `location`: The location of the cell in the input document as defined by its `begin` and `end` indexes.
        - `text`: The textual contents of the cell from the input document without associated markup content.
        - `text_normalized`: Normalized row header text.
        - `row_index_begin`: The `begin` index of the cell's `row` location in the current table.
        - `row_index_end`: The `end` index of the cell's `row` location in the current table.
        - `column_index_begin`: The `begin` index of the cell's `column` location in the current table.
        - `column_index_end`: The `end` index of the cell's `column` location in the current table.
    - `body_cells`: An array of cells that are not table header or column header or row header cells, of the current table with corresponding row and column header associations. Each body cell is defined as a collection of the following items:
        - `cell_id`: The unique ID of the cell in the current table.
        - `location`: The location of the cell in the input document as defined by its `begin` and `end` indexes.
        - `text`: The textual contents of the cell from the input document without associated markup content.
        - `row_index_begin`: The `begin` index of this cell's `row` location in the current table.
        - `row_index_end`: The `end` index of this cell's `row` location in the current table.
        - `column_index_begin`: The `begin` index of this cell's `column` location in the current table.
        - `column_index_end`: The `end` index of this cell's `column` location in the current table.
        - `row_header_ids`: An array of values, where each value is the cell ID value of a row header that is associated with this body cell.
        - `row_header_texts`: An array of values, where each value is the text from a row header for this body cell.
        - `row_header_texts_normalized`: An array of values, where each value is the normalized text from a row header for this body cell.
        - `column_header_ids`: An array of values, where each value is the cell ID value of a column header that is associated with this body cell.
        - `column_header_texts`: An array of values, where each value is the text from a column header for this body cell.
        - `column_header_texts_normalized`: An array of values, where each value is the normalized text from a column header for this body cell.
        - `attributes`: An array that identifies document attributes. Each object in the array consists of three elements:
            - `type`: The type of attribute. Possible values are `Address`, `Currency`, `DateTime`, `Duration`, `Location`, `Number`, `Organization`, `Percentage`, and `Person`.
            - `text`: The text that is associated with the attribute.

        - `location`: The location of the attribute as defined by its `begin` and `end` indexes.
- `key_value_pairs`: An array that specifies any key-value pairs in tables in the input document. For more information, see Understanding key-value pairs.
    - `key`: An object that specifies a key for a key-value pair.
        - `cell_id`: The unique ID of the key in the table.
        - `location`: The location of the key cell in the input document as defined by its `begin` and `end` indexes.
        - `text`: The text content of the table cell without HTML markup.
    - `value`: An array that specifies the value or values for a key-value pair.
        - `cell_id`: The unique ID of the value in the table.
        - `location`: The location of the value cell in the input document as defined by its `begin` and `end` indexes.
        - `text`: The text content of the table cell without HTML markup.
- `contexts`: A list of related material that precedes and follows åthe table, excluding its section title, which is provided in the `section_title` field. Related material includes related sentences; footnotes; and sentences from other parts of the document that refer to the table. The list is represented as an array. Each object in the array consists of the following elements:
    - `text`: The text contents of a related material from the input document, without HTML markup.
    - `location`: The location of the related material in the input document as defined by its `begin` and `end` indexes.

Notes on the table output schema

- Row and column index values per cell are zero-based and so begin with `0`.
- Multiple values in arrays of `row_header_ids` and `row_header_texts` elements indicate a possible hierarchy of row headers.
- Multiple values in arrays of `column_header_ids` and `column_header_texts` elements indicate a possible hierarchy of column headers.

Examples

The following table is an example table from an input document.

	Three months ended September 30,		Nine months ended September 30,	
	2005	2004	2005	2004
Statutory tax rate	35.0%	35.0%	35.0%	35.0%
IRS audit settlement	(97.9)%	(36.0)%	(58.4)%	(15.2)%
Dividends received deduction	(13.2)%	(3.3)%	(15.4)%	(4.7)%
Total effective tax rate	(76.1)%	(4.3)%	(38.8)%	15.1%

Figure 1. Table example

The table is composed as follows:

	Column header		**Column header**	
	Column header	**Column header**	**Column header**	**Column header**
Row header	Body cell	Body cell	Body cell	Body cell
Row header	Body cell	Body cell	Body cell	Body cell
Row header	Body cell	Body cell	Body cell	Body cell
Row header	Body cell	Body cell	Body cell	Body cell

Figure 2. Anatomy of the example table

The following syntax is used in the table:

- **Bold text** indicates a column header
- *Italic text* indicates a row header
- Unstyled text indicates a body cell

The output from service represents the example's first body cell (that is, the first cell in row 3 with a value of `35.0%`) as follows:

```
{
   "tables": [ {
      "location": {
         "begin": 872,
```

```
        "end": 5879
      },
      "text": "...",
      "section_title": {
        "text": "",
        "location": {
          "begin": 0,
          "end": 0
        }
      },
      "table_headers" : [ ],
      "column_headers" : [ {
        "cell_id" : "colHeader-1050-1082",
        "location" : {
          "begin" : 1050,
          "end" : 1083
        },
        "text" : "Three months ended September 30,",
        "text_normalized" : "Three months ended September 30,",
        "row_index_begin" : 0,
        "row_index_end" : 0,
        "column_index_begin" : 1,
        "column_index_end" : 2
      }, {
        "cell_id" : "colHeader-1270-1301",
        "location" : {
          "begin" : 1270,
          "end" : 1302
        },
        "text" : "Nine months ended September 30,",
        "text_normalized" : "Nine months ended September 30,",
        "row_index_begin" : 0,
        "row_index_end" : 0,
        "column_index_begin" : 3,
        "column_index_end" : 4
      }, {
        "cell_id" : "colHeader-1544-1548",
        "location" : {
          "begin" : 1544,
          "end" : 1549
        },
        "text" : "2005",
        "text_normalized" : "Year 1",
        "row_index_begin" : 1,
        "row_index_end" : 1,
        "column_index_begin" : 1,
        "column_index_end" : 1
      }, {
        "cell_id" : "colHeader-1712-1716",
        "location" : {
          "begin" : 1712,
          "end" : 1717
        },
        "text" : "2004",
        "text_normalized" : "Year 2",
        "row_index_begin" : 1,
        "row_index_end" : 1,
        "column_index_begin" : 2,
        "column_index_end" : 2
      }, {
        "cell_id" : "colHeader-1889-1893",
        "location" : {
          "begin" : 1889,
          "end" : 1894
        },
        "text" : "2005",
        "text_normalized" : "Year 1",
        "row_index_begin" : 1,
        "row_index_end" : 1,
        "column_index_begin" : 3,
        "column_index_end" : 3
      }, {
        "cell_id" : "colHeader-2057-2061",
        "location" : {
          "begin" : 2057,
          "end" : 2062
        },
        "text" : "2004",
```

```
        "text_normalized" : "Year 2",
        "row_index_begin" : 1,
        "row_index_end" : 1,
        "column_index_begin" : 4,
        "column_index_end" : 4
      } ],
      "row_headers" : [ {
        "cell_id" : "rowHeader-2244-2262",
        "location" : {
          "begin" : 2244,
          "end" : 2263
        },
        "text" : "Statutory tax rate",
        "text_normalized" : "Statutory tax rate",
        "row_index_begin" : 2,
        "row_index_end" : 2,
        "column_index_begin" : 0,
        "column_index_end" : 0
      }, {
        "cell_id" : "rowHeader-3197-3217",
        "location" : {
          "begin" : 3197,
          "end" : 3218
        },
        "text" : "IRS audit settlement",
        "text_normalized" : "IRS audit settlement",
        "row_index_begin" : 3,
        "row_index_end" : 3,
        "column_index_begin" : 0,
        "column_index_end" : 0
      }, {
        "cell_id" : "rowHeader-4148-4176",
        "location" : {
          "begin" : 4148,
          "end" : 4177
        },
        "text" : "Dividends received deduction",
        "text_normalized" : "Dividends received deduction",
        "row_index_begin" : 4,
        "row_index_end" : 4,
        "column_index_begin" : 0,
        "column_index_end" : 0
      }, {
        "cell_id" : "rowHeader-5106-5130",
        "location" : {
          "begin" : 5106,
          "end" : 5131
        },
        "text" : "Total effective tax rate",
        "text_normalized" : "Total effective tax rate",
        "row_index_begin" : 5,
        "row_index_end" : 5,
        "column_index_begin" : 0,
        "column_index_end" : 0
      } ],
      "key_value_pairs" : [ ],
      "body_cells" : [ {
        "cell_id" : "bodyCell-2450-2455",
        "location" : {
          "begin" : 2450,
          "end" : 2456
        },
        "text" : "35.0%",
        "row_index_begin" : 2,
        "row_index_end" : 2,
        "column_index_begin" : 1,
        "column_index_end" : 1,
        "row_header_ids" : [ "rowHeader-2244-2262" ],
        "row_header_texts" : [ "Statutory tax rate" ],
        "row_header_texts_normalized" : [ "Statutory tax rate" ],
        "column_header_ids" : [ "colHeader-1050-1082", "colHeader-1544-1548" ],
        "column_header_texts" : [ "Three months ended September 30,", "2005" ],
        "column_header_texts_normalized" : [ "Three months ended September 30,", "Year 1" ],
        "attributes": [ ]
      }, {
        "cell_id" : "bodyCell-2633-2638",
        "location" : {
          "begin" : 2633,
```

```
          "end" : 2639
        },
        "text" : "35.0%",
        "row_index_begin" : 2,
        "row_index_end" : 2,
        "column_index_begin" : 2,
        "column_index_end" : 2,
        "row_header_ids" : [ "rowHeader-2244-2262" ],
        "row_header_texts" : [ "Statutory tax rate" ],
        "row_header_texts_normalized" : [ "Statutory tax rate" ],
        "column_header_ids" : [ "colHeader-1050-1082", "colHeader-1712-1716" ],
        "column_header_texts" : [ "Three months ended September 30,", "2004" ],
        "column_header_texts_normalized" : [ "Three months ended September 30,", "Year 2" ],
        "attributes": [ ]
      }, {
        "cell_id" : "bodyCell-2825-2830",
        "location" : {
          "begin" : 2825,
          "end" : 2831
        },
        "text" : "35.0%",
        "row_index_begin" : 2,
        "row_index_end" : 2,
        "column_index_begin" : 3,
        "column_index_end" : 3,
        "row_header_ids" : [ "rowHeader-2244-2262" ],
        "row_header_texts" : [ "Statutory tax rate" ],
        "row_header_texts_normalized" : [ "Statutory tax rate" ],
        "column_header_ids" : [ "colHeader-1270-1301", "colHeader-1889-1893" ],
        "column_header_texts" : [ "Nine months ended September 30,", "2005" ],
        "column_header_texts_normalized" : [ "Nine months ended September 30,", "Year 1" ],
        "attributes": [ ]
      }, {
        "cell_id" : "bodyCell-3008-3013",
        "location" : {
          "begin" : 3008,
          "end" : 3014
        },
        "text" : "35.0%",
        "row_index_begin" : 2,
        "row_index_end" : 2,
        "column_index_begin" : 4,
        "column_index_end" : 4,
        "row_header_ids" : [ "rowHeader-2244-2262" ],
        "row_header_texts" : [ "Statutory tax rate" ],
        "row_header_texts_normalized" : [ "Statutory tax rate" ],
        "column_header_ids" : [ "colHeader-1270-1301", "colHeader-2057-2061" ],
        "column_header_texts" : [ "Nine months ended September 30,", "2004" ],
        "column_header_texts_normalized" : [ "Nine months ended September 30,", "Year 2" ],
        "attributes": [ ]
      },
      ...
    ],
    "contexts": [ ]
}
```


Understanding key-and-value pairs

Tables sometimes contain key-and-value pairs that span multiple table cells. **Table Understanding** can detect the following types of tabular pairs.

- Simple key-and-value pairs in adjacent cells, as in the following example table:

Key	Value
Item number	`123456789`
Date	`1/1/2019`
Amount	`$1,000`

Basic table

- Key-and-value pairs in the same cell, as in the following example table:

Key-value pairs	Key-value pairs
Item number: `123456789`	Amount: `$1000`
Date: `1/1/2019`	Address: `123 Anywhere Dr`

Complex table

Use SDU to analyze document structure

Analyzing documents based on their structure

Create a model that understands the content of a document based on the document's format and structure.

First, decide whether you want to use a pretrained model or define your own.

Pretrained model

Applies a noncustomizable model that extracts text and identifies tables, lists, and sections.

Instead of training the model yourself, you can apply an existing model that is trained to identify tables, lists, and sections in various types of documents.

If capturing information from tables is critical to your use case, consider using a pretrained model.

For more information, see Apply a pretrained SDU model.

User-trained model

Opens the Smart Document Understanding tool that you can use to pick certain types of text to store in fields other than the `text` field.

When you label a section of a document as a custom field, later you can apply enrichments to the field or split your documents on each occurrence of the field. You can search or filter by the field, or omit the field from the index.

For more information, see Define a user-trained SDU model.

Text extraction only

Indexes any text that is recognized in the source documents in the `text` field. This option is used by default.

Define a user-trained SDU model

Create a Smart Document Understanding (SDU) model that learns about the content of a document based on the document's structure.

Use the Smart Document Understanding tool to add custom fields to a collection so you can do the following things:

- Target prebuilt or custom enrichments at specific sections of a document.
- Break large documents into smaller documents.

For help with deciding whether SDU can help your use case, read When to use Smart Document Understanding.

If capturing information from tables is critical to your use case, consider using a pretrained model. For more information about creating a pretrained SDU model, see Apply a pretrained SDU model.

When to use Smart Document Understanding

The Smart Document Understanding (SDU) tool works better with some project types.

- The tool is most beneficial when used with *Document Retrieval* projects. Use the tool to break your documents into smaller, more consumable chunks of information. When you help Discovery index the correct set of information in your documents, you improve the answers that your application can find and return.

  For example, your documents might contain tips that are shown in sections with an H4 heading. If you want to extract the information from these tips separately, you can add a field that is named `tips`, and teach the model to recognize it. After you apply the model to your collection, you can apply an enrichment to the `tips` field only. Later, you can limit the search to return content from only the `tips` field.

  Or maybe you have extra large documents that contain subsections. You can teach the SDU model to recognize these subsections, and then split the large document into multiple, smaller, and easier-to-manage documents that begin with one of these subsections.

- The best way to prepare a collection for use in *Conversational Search* projects is to identify discrete question-and-answer pairs. You can use the SDU tool to find and annotate them. If you configure the project to contain answers in an answer field, you must update the search configuration in watsonx Assistant to get the body of the response from the custom answer field.
- A pretrained SDU model is applied to *Document Retrieval for Contracts* projects automatically. The pretrained SDU model knows how to recognize terms and concepts that are significant to contracts. As a result, you cannot apply a user-trained SDU model to this project type, but you also don't need to.
- The SDU tool is rarely used with *Content Mining* projects.

You can use the SDU tool to annotate the following file types only:

- Image files (PNG, TIFF, JPG)
- Microsoft PowerPoint
- Microsoft Word
- PDF

For a complete list of file types that Discovery supports, see Supported file types.

The Smart Document Understanding tool uses optical character recognition (OCR) to extract text from images in the files that it analyzes. Images must meet the minimum quality requirements that are supported by OCR. For more information, see Optical character recognition.

The tool cannot read documents with the following characteristics; remove them from your collection before you begin:

- Documents that appear to have text that overlays other text are considered *double overlaid* and cannot be annotated.
- Documents that contain multiple columns of text on a single page cannot be annotated.

Note: When you build a custom Smart Document Understanding model, the conversion time for your collection can increase due to the resources that are required to apply the AI model to your documents.

Start with representative documents

Documents come in all shapes and sizes. Your collection might have a mix of different document structures. Smart Document Understanding works best when the documents in a single collection have similar style characteristics. For example, the documents use consistent font sizes and colors for titles and headers, and tables in the document have similar layouts. To create the best model for your collection, take this prerequisite step:

1. Review your documents to look for style and layout patterns, and then separate the documents into groups based on their style.

   For example, if your data contains documents that follow four different formatting styles, break the documents into four separate collections, one for each style. Add documents with a uniform layout and style to each collection. A good target size per collection is 40 documents.

2. Use the SDU tool to annotate this representative set of documents and train Watson to recognize custom content in your data.
3. Apply the custom SDU model to the full collection. For more information, see Reusing SDU models.

Creating the model

To apply a user-trained Smart Document Understanding model to your collection, complete the following steps:

1. Open the **Manage collections** page from the navigation panel.
2. If your project has more than one collection, select the collection with documents that you want to annotate.
3. Open the *Identify fields* page.
4. Choose **User-trained models**.

   The **Text extraction only** option is used by default. With this model, any text that is recognized in the source documents is indexed in the `text` field.

5. Click **Submit**, and then click **Apply changes and reprocess**.

A subset of documents is available for you to annotate. A set of 20 - 50 documents is displayed in a list. The number of documents that are available differs based on several factors, including the overall number of documents in your collection and how many of them are supported file types.

Labeling video

The following video shows you how to select a label, and then apply it to a representation of the text in your document.

In the video, the user clicks the `title` field label, and then clicks the text block that represents the *Table of Contents* page title to label the text as a title. Next, the user clicks the `table_of_contents` field label and selects the table of contents text block to label it. Then, the user clicks the `footer` field label and clicks the text block that represents the page footer. After the text is labeled, the user clicks the *Submit page* button.

Your browser does not support the video tag.

Labeling the documents

Before you begin, get a feel for the structure of the document you plan to annotate. Are there subtitled sections that you want Discovery to return per answer? If so, identify all subtitles. Later you can split the document into discrete subdocuments, each starting with a subtitle. For more information, see When to use Smart Document Understanding.

To label documents, complete the following steps:

1. Review the document preview.

   A view of the original document is displayed along with a representation of the document, where the text is replaced by blocks.

   The blocks are all the color of the `text` field label because all of the current text is considered to be standard text and will be indexed in the `text` field.

   Label blocks that represent specific types of information, such as titles or page footers, with other field labels. For example, when you apply the title field label to a document title that would otherwise be indexed as text, you are defining a more precise representation of the document content.

   The process of using labels to identify different parts of the document's structure is called *annotating* the document.

2. Review the field labels that you can use to annotate the document. They are displayed in the *Field labels* panel.

   See the Default field labels table for a list of the fields and their descriptions.

3. To create a custom field label, click **Create new**.
   - Specify a field label with no spaces. For example, `complex_task` is a valid field label.

     Note: Avoid using a field label name or including characters, such as a number sign (#) or period (.), in the name that have special meaning for Discovery. For more information, see How fields are handled.

   - If you want to change the color that is used to represent the field, repeatedly click the color block until it is displayed in the color that you want to use.

     Important: You cannot change the field label color later.

   - Click **Create**.
4. First, click a field label to activate it.
5. Next, click the block that represents the content that you want to label as the field type.

   The block changes to the color of the field label. You successfully labeled the field!
6. Repeat this process to annotate more fields in the document.

   Don't worry. You don't need to label every page. As you apply labels and submit pages, Watson learns from what you annotate and starts to predict annotations.

   Follow these guidelines:
   - If there's nothing special about a section, leave it labeled as `text`, which is applied by default.
   - A label cannot span multiple pages.
   - Do not treat **bold**, *italic*, or underlined text differently. Label based on the context, not the style.
   - Use consistent labeling on all documents.
   - Work from the first page of a multipage document to the last.
   - To remove a single annotation, choose another label (such as `text`) and apply it to the item to overwrite the previous annotation.
   - To remove annotations that you added to an entire page, click the **Clear changes** icon in the toolbar.
   - To annotate a table, click the text at the start of the table and then drag to select the text in the entire table.
   - When you label one or more tables, the *Table Understanding* enrichment is enabled for the entire collection automatically. For more information, see Understanding tables.
   - Images from the source documents are not rendered in the preview. If Optical Character Recognition (OCR) is enabled, any text from the image or diagram is extracted and rendered in the preview.
   - Do not label white space.
7. When everything that you want to label is labeled, submit the page. Click **Submit page**.

   Note: Continue annotating documents until Watson can correctly and consistently map different types of content to the appropriate fields for you.

8. After you teach Watson to identify fields, click **Apply changes and reprocess**.

Custom fields that you define by using the SDU tool are indexed as root-level fields.

What to do next

When you build a user-trained model, you change where information is stored in your documents. Next, change how the search results are configured. By default, search results are retrieved from passages or the text field. You might have a better field to use as the source of the result body. For more information, see Changing the result content.

If your project is being used by a virtual assistant, update the search skill configuration to pull the answer body from a different field. For more information, see Configure the search.

You can apply enrichments, either custom or prebuilt enrichments, to the new root fields that are generated by the SDU model.

If you want to return shorter text snippet with a search result, you can split your documents based on one of the new fields that you defined, such as chapter or section.

Available fields

The following fields are available for you to apply to documents by using the Smart Document Understanding tool.

The fields are arbitrary. You can apply the `image` field to every title in the document if you want. Although, it might be difficult to know which field to search later for information that you need if the field names don't match the content. The default set are representative field types that are meant to help you get started. Only the `text` and `table` fields have special significance. Do not use them to identify anything other than text and tables.

Field	Definition
`answer`	In a question-and-answer pair (often in an FAQ), the answer to the question.
`author`	Name of author or authors.
`footer`	Use this tag to denote meta-information about the document (such as the page number or references), that appear at the end of the page.
`header`	Use this tag to denote meta-information about the document that appears at the start of the page.
`question`	In a question-and-answer pair (often in an FAQ), the question.
`subtitle`	The secondary title of the document.
`table_of_contents`	Use this tag on lists in the document table of contents.
`text`	By default, every block of text in the document is labeled as text. Apply different labels only to blocks of text with special meaning.
`title`	The main title of the document.
`table`	Use this tag to annotate tables in your document.
`image`	Images are not shown in the document preview. If you enable OCR, text from an image or diagram is displayed in the preview instead. If you want to prevent text from some images from being included in search results, tag the image text as an image. You can exclude the image field from the index later.

Default field labels

Reusing SDU models

After you define a model with the SDU tool, you can save it and reuse it in other collections by exporting it from one collection and importing it to another.

Note: Importing a new model overwrites the existing model in a collection. If the existing model is already trained such as through custom field labels and annotations, then importing a new model affects the collection and can result in data loss.

To reuse a model, complete the following steps:

1. Export the model that you want to reuse. From the SDU toolbar menu, select **Export model**.

Figure 1. Import and export menu

2. Create the collection where you want to reuse the model. Add only one document to the collection at first.
3. Import the model from the SDU toolbar. The exported model has a file extension of `.sdumodel`.
4. Add the rest of the documents to the collection. Open the **Activity** tab of the *Manage collections* page, and then click **Upload data** to add more files to the collection.

Use the imported model as-is. Do not make any more annotations. If you make annotations after you import the `.sdumodel` file, the imported model will be overwritten.

Smart Document Understanding limits

The number of custom fields that you can create per Smart Document Understanding model depends on your Discovery plan type.

Plan	Custom fields per SDU model
Cloud Pak for Data	Unlimited
Premium	100
Enterprise	100
Plus (includes Trial)	40

Custom field limits

The maximum number of documents that you can annotate to train an SDU model per collection depends on your Discovery plan type.

Plan	Documents per collection
Cloud Pak for Data	40
Premium	40
Enterprise	40
Plus (includes Trial)	40

Training set limits

Managing fields

The **Manage fields** tab contains several options:

Identify fields to index

For more information, see Excluding content from query results.

Improve query results by splitting your documents

For more information, see Split documents to make query results more succinct.

Date format settings

For more information, see Date format settings.

To access the **Manage fields** page, click the **Manage collections** icon on the navigation panel and open a collection. Click the **Manage fields** tab. For more information about collections, see Creating collections.

Apply a pretrained SDU model

Apply a prebuilt Smart Document Understanding (SDU) model that can extract text and is trained to identify tables, lists, and sections in documents.

Use the pretrained model if your documents contain tables with valuable information that you want to capture. The model is also able to preserve the meaning inherent in the nesting structure of tables, lists, and sections. Using the pretrained model speeds up the process of capturing information from the structure of a document.

If you want to customize how the document structure is used to infer meaning from a document or you want to split documents with a field that is generated by an SDU model, create a user-trained model instead. For more information, see Define a user-trained SDU model.

A pretrained model is applied to *Document Retrieval for Contracts* projects automatically. Instead of you annotating contract-related content in your documents, the project applies a model that already knows how to recognize terms and concepts that are significant to contracts.

Preparing documents

You can apply a pretrained SDU model to the following file types only:

- Image files (PNG, TIFF, JPG)
- Microsoft PowerPoint
- Microsoft Word
- PDF

For a complete list of file types that Discovery supports, see Supported file types.

The Smart Document Understanding tool uses optical character recognition (OCR) to extract text from images in the files that it analyzes. Images must meet the minimum quality requirements that are supported by OCR. For more information, see Optical character recognition.

The tool cannot read documents with the following characteristics; remove them from your collection before you begin:

- Documents that appear to have text that overlays other text are considered *double overlaid* and cannot be annotated.
- Documents that contain multiple columns of text on a single page cannot be annotated.

Note: When you apply a Smart Document Understanding model, the conversion time for your collection can increase due to the resources that are required to apply the AI model to your documents.

Apply a pretrained model

To apply a pretrained Smart Document Understanding model to your collection, complete the following steps:

1. Open the **Manage collections** page from the navigation panel.
2. Select the collection to which you want to apply the model.
3. Open the *Identify fields* page.
4. Choose **Pre-trained models**

   The **Text extraction only** option is used by default. With this model, any text that is recognized in the source documents is indexed in the `text` field.
5. Click **Submit**, and then click **Apply changes and reprocess**.

Understanding the output

If the SDU model finds and processes a structure, such as a table, in the document, it stores a representation of the structure in a field named `enriched_{field}`, where `{field}` is the field where the structure was stored.

The following excerpt shows the JSON representation of a table from the `enriched_html` field of a document that was processed by the pretrained SDU model.

Figure 1. JSON table representation

If you want to extract text from the processed structure, you can use the `location` field to find the index values that identify where the text string starts and ends.

For more information about the structure of indexed tables, see Understanding tables.

Troubleshooting issues

Follow these workarounds if you experience problems when working with the Smart Document Understanding tool.

Insufficient resources to process document

Error

When you apply a pretrained model to your collection, document processing does not complete successfully and an `Insufficient resources to process document` message is displayed.

Cause

The error is displayed because out of memory errors occur during the parse, structure identification, or assembly phases of the process that builds the machine learning model. Resources are insufficient when one or more of the documents in your collection are too large or have too many complex tables for the tool to handle.

Solution

Review your collection for large documents or documents with many tables and break them up into more smaller documents before you apply the pretrained model to the collection. Exact limits differ based on the complexity of your documents. Generally, split documents that are over 400 pages long and avoid including more than 20 complex tables in a single document.

Manage enrichments

Apply enrichments to fields in your documents to make meaningful information easier to find and return in searches.

You typically apply enrichments to fields at the time that you create the enrichment. However, you can apply enrichments to fields later. You might want to apply an existing enrichment to a new custom field that you create by using the SDU tool, for example. You can remove enrichments that you applied to fields also.

For more information about available enrichments, see the following topics:

- Adding domain-specific resources
- Applying prebuilt enrichments

To manage enrichments, complete the following steps:

1. From the navigation pane, open the **Manage collections** page, and then click a collection to open it.
2. Click the **Enrichments** tab.

   A list of available enrichments is displayed.

   You can identify built-in enrichments because they are categorized as type `System`. The list also includes any custom enrichments that were created in any of the projects in your service instance.
3. Find the enrichment that you want to apply or remove.

4. Click the twistie in the associated field to expand a list of fields.
5. Do one of the following things:
    - To apply an enrichment to documents, select the field or fields that you want to enrich. You can apply enrichments to the `text` and `html` fields, and to custom fields that were added from uploaded JSON or CSV files or from the Smart Document Understanding (SDU) tool.

      Note: If the field that you choose comes from a JSON file, after you apply the enrichment, the field data type is converted to an array. The field is converted to an array even if it contains a single value. For example, `"field1": "Discovery"` becomes `"field1": ["Discovery"]`. Only the first 50,000 characters of a custom field from a JSON file are enriched.

    - To remove an enrichment, clear the checkboxes for fields that you want to remove the enrichment from.
6. Click **Apply changes and reprocess** to apply your changes to the collection.

Deleting an enrichment

You can delete a custom enrichment that you built to teach Discovery about your service. Custom enrichments include dictionaries, regular expressions, patterns, and so on. For more information, see Adding domain-specific resources.

You cannot delete a prebuilt enrichment. Prebuilt enrichments include the Natural Language Understanding enrichments, such as the Entities enrichment, that are built into the product. To determine which enrichments are built in, check the *Type* column of the *Enrichments* page for a collection. Prebuilt enrichments have the `System` type.

To delete a custom enrichment, complete the following steps:

1. Open a project that uses the custom enrichment.
2. Click *Manage collections*, and then open a collection where the enrichment is in use.
3. From the *Enrichments* page of a collection, remove the enrichment from any fields where it is applied.
4. Click **Apply changes and reprocess**, and then wait for the system to process the documents in your collection without the enrichment.
5. Repeat the previous step for every collection in every project where the enrichment is used.

   Remember, a custom enrichment can be used by any collection in any project within a single Discovery service instance.
6. From any of the collections that used the enrichment, open the *Enrichments* page, and then click the delete icon to delete the enrichment.

The custom enrichment is removed from the *Enrichments* list everywhere in this service instance.

Using the API to manage enrichments

To apply an enrichment to data by using the API, complete the following steps:

1. First, you must know the unique ID of the enrichment that you want to apply. For more information, see Enrichment IDs.
2. Use the *Create collection* or *Update collection* methods to apply an enrichment to the documents in a collection. For more information, see Apply enrichment by using the API

Enrichment IDs

If you want to apply a custom enrichment that you created for one collection to another collection, you must know the unique ID that was generated for the enrichment when it was created. Use the API to list enrichments from the project where the custom enrichment is in use. The list that is returned includes enrichment ID information.

For prebuilt enrichments, the unique IDs do not change. The following table lists the IDs that are associated with each prebuilt enrichment type and identifies the collection languages for which the enrichment is supported. An enrichment cannot be applied to a collection unless the collection language is supported by the enrichment.

Name	Enrichment ID	Supported languages
Contracts	701db916-fc83-57ab-0000-000000000014	en
Entities	701db916-fc83-57ab-0000-00000000001e	ar, de, en, es, fr, it, ja, ko, nl, pt, zh-CN
Keywords	701db916-fc83-57ab-0000-000000000018	ar, de, en, es, fr, it, ja, ko, nl, pt, zh-CN
Part of Speech	701db916-fc83-57ab-0000-000000000002	All supported languages

Sentiment of Document	701db916-fc83-57ab-0000-000000000016	ar, de, en, es, fr, it, ja, ko, nl, pt, zh-CN
Table Understanding	701db916-fc83-57ab-0000-000000000012	All supported languages

Prebuilt enrichment IDs

For more information about all of the supported languages, see Language support.

Apply enrichments by using the API

To apply an enrichment by using the API, complete the following steps:

1. Determine the base URL for the endpoint and the token or API key for your deployment.

   For more information, see Building custom applications with the API.

2. Get the project ID for your project.

   From the product user interface, go to the **Integrate and Deploy** > **View API information** page, and then copy the project ID.

3. If you don't know the ID of the collection that you want to apply the enrichment to, get a list of your collections to find it.

   For example:

   ```
   GET $authentication $url/v2/projects/$project_id/collections?version=2019-11-22
   ```

   The `collection_id` is returned.

4. Send a GET request to return the configuration of the collection that lists the applied enrichments.

   For example:

   ```
   GET $authentication $url/v2/projects/$project_id/collections/$collection_id?version=2019-11-22
   ```

   For the enrichment IDs for the prebuilt enrichments, see Enrichment IDs.

5. Add the enrichment that you want to apply.

   For example, to add the *Keywords* enrichment, you can include the enrichment in the enrichments list. First, get its ID from the table.

   The Keywords enrichment ID is `701db916-fc83-57ab-0000-000000000018`. To indicate that you want to apply the Keywords enrichment to the content in the `text` field of the documents in the collection, you can represent it in JSON format as follows:

   ```
   {
         "enrichment_id" : "701db916-fc83-57ab-0000-000000000018",
         "fields" : [ "text" ]
   }
   ```

   Note: Any enrichments that you specify replace the default enrichments. Therefore, if you want to retain a default enrichment, don't forget to include it in the list of enrichments that you apply to the collection. For a list of default enrichments per project type, see Default enrichments per project type.

   For example, to retain the *Entities* enrichment and add the *Keywords* enrichment, you might specify the following in the request body.

   ```
   "enrichments": [
     {
       "enrichment_id": "701db916-fc83-57ab-0000-00000000001e",
       "fields": [
         "text"
       ]
     },
     {
       "enrichment_id": "701db916-fc83-57ab-0000-000000000018",
       "fields": [
         "text"
       ]
     }
   ]
   ```

6. Submit the updated JSON request body with the update collection method to apply the enrichment to your collection.

For example:

```
POST $authentication -d '$requestBody' $url/v2/projects/$project_id/collections/$collection_id?version=2019-11-22
```

Default enrichments per project type

Some prebuilt enrichments are applied automatically to collections in a project based on the project type. The following table shows the default enrichments that are applied to each project type.

Enrichment	Document Retrieval	Document Retrieval for Contracts	Conversational Search	Content Mining
Contracts		✓		
Entities	✓	✓		
Keywords				
Part of Speech				✓
Sentiment of Document				
Table Understanding		✓		

Default enrichments per project type

External enrichment API

IBM Cloud

The external enrichment is beta functionality that is only available from the API.

The external enrichment feature allows you to annotate documents with a model of your choice. Through a webhook interface, you can use custom models or advanced foundation models, and other third-party models for enriching your documents in a collection. The documents are enriched by your external application and then merged to a collection in a Discovery project.

Note: The external enrichment feature is not supported in the Analyze API.

For using the external enrichment feature, do the following things:

1. Set up the external application that can receive webhook notifications from Discovery and annotate documents.

   To do so, you must register your external app as a webhook endpoint on a project by using the `create enrichment` method. For more information, see Create enrichment in the API reference.

   After setting up the external enrichment for a project, it becomes available to all collections in the project. The external application also receives a webhook `ping` event, which notifies that an external enrichment is created.

2. Specify the collection in which you want to apply the external enrichment. You can use the API to apply the external enrichment to a collection. For more information, see Using the API to manage enrichments.

   Alternatively, on the user interface, you can browse to the *Manage collections* page, and choose the collection where you want to apply the external enrichment. Then, open the **Enrichments** tab, and apply your external enrichment to a field in the collection.

   When documents are processed or uploaded to this collection, Discovery creates a batch of documents with a unique `batch_id`. The external application also receives a webhook `enrichment.batch.created` event, which notifies that batches are ready to be pulled. Your external application can then pull batches from Discovery for external enrichment.

   If the external application shuts down or restarts in between, you can get the following by using the List batches method:

   - Notified batches that are not yet pulled by the external enrichment application.
   - Batches that are pulled, but not yet pushed to Discovery by the external enrichment application.

   For more information, see List batches in the API reference.

3. Specify the `batch_id` provided by Discovery in the `pull batches` method to pull the documents from Discovery for enrichment by your external application. For more information, see Pull batches in the API reference.

   The `pull batches` method returns a binary file attachment from Discovery. For more information about the binary attachment, see Binary

attachment from the pull batches method.

4. Specify the same `batch_id` in the `push batches` method after your external enrichment annotates the documents in the batch. For more information, see Push batches in the API reference.

   The documents are pushed to Discovery as a binary attachment. For more information, see Binary attachment in the push batches method .

5. Verify that the documents are merged and indexed in the collection. The documents must contain the annotations that are applied by your external application.

External enrichment limits

Plan	Maximum amount of webhook enrichment per collection	Maximum amount of webhook enrichment per tenant
Enterprise	1	100
Plus	1	10
Premium	1	100

External enrichment limits

Binary attachment from the pull batches method

The `pull batches` method returns a binary attachment file from Discovery.

The returned file is a compressed newline-delimited JSON (NDJSON) file. This file contains structured data that represents the document properties. For example, the following is a JSON value included in the NDJSON file:


```
{
      "document_id":  "3bafc09abfaacd90d66f57181b50d041",
      "location_encoding":  "utf-16",
      "language":  "en",
      "artifact":  "{\"text_positions\":
[0,21],\"space_above\":93.07864284515381,\"space_below\":32.53530788421631,\"is_start_of_block\":true,\"image_id\":-1}{\"text_positions\":
[22,63],\"space_above\":32.53530788421631,\"space_below\":13.935576438903809,\"is_start_of_block\":true,\"image_id\":-1}
{\"parent_document_id\":\"3bafc09abfaacd90d66f57181b50d041\",\"source\":{\"ListId\":\"f0ac1d32-b9e5-41af-b9da-
e1e37e965d99\",\"UniqueId\":\"357d7a48-4460-442c-be56-
d8bdd40a8c36\",\"ServerRelativeUrl\":\"/Lists/list1/Attachments/1/addattachments.csv\",\"FileNameAsPath\":
{\"DecodedUrl\":\"addattachments.csv\"},\"ListItemId\":\"284dcb51-8021-56d0-9213-
7f4eb134e083\",\"FileName\":\"addattachments.csv\",\"ServerRelativePath\":
{\"DecodedUrl\":\"/Lists/list1/Attachments/1/addattachments.csv\"},\"WebId\":\"ad5bf592-3b4e-4dd1-bd3e-
abc0ef179b03\"},\"ingest_datetime\":\"2023-06-
26T09:24:02.573Z\",\"application_id\":\"sharepoint\",\"application_sub_type\":\"ListItemAttachmentCollection\"}0.51vanilla ice
creamcontamination_tamperingotherchange_of_propertiesI love the ads for the new milk chocolate. Could you tell me the name of the actor in the
commercial?{\"metadata\":{\"numPages\":\"54\",\"title\":\"\",\"publicationdate\":\"2010-06-03\"},\"info\":{\"histogram\":{\"mean-char-height\":
{},\"mean-char-width\":{},\"number-of-chars\":{}},\"styles\":[]}}1451692800000",
      "features":  [
            {
                  "type":  "field",
                  "location":  {
                        "begin":  0,
                        "end":  128
                  },
                  "properties":  {
                        "field_name":  "multi_nested",
                        "field_index":  0,
                        "field_type":  "json"
                  }
            },
            {
                  "type":  "field",
                  "location":  {
                        "begin":  128,
                        "end":  258
                  },
                  "properties":  {
                        "field_name":  "multi_nested",
                        "field_index":  1,
                        "field_type":  "json"
                  }
            },
            {
                  "type":  "field",
```

```
            "location": {
                "begin": 258,
                "end": 889
            },
            "properties": {
                "field_name": "metadata",
                "field_index": 0,
                "field_type": "json"
            }
        },
        {
            "type": "field",
            "location": {
                "begin": 889,
                "end": 892
            },
            "properties": {
                "field_name": "claim_score",
                "field_index": 0,
                "field_type": "double"
            }
        },
        {
            "type": "field",
            "location": {
                "begin": 892,
                "end": 893
            },
            "properties": {
                "field_name": "claim_id",
                "field_index": 0,
                "field_type": "long"
            }
        },
        {
            "type": "field",
            "location": {
                "begin": 893,
                "end": 910
            },
            "properties": {
                "field_name": "claim_product",
                "field_index": 0,
                "field_type": "string"
            }
        },
        {
            "type": "field",
            "location": {
                "begin": 910,
                "end": 933
            },
            "properties": {
                "field_name": "label",
                "field_index": 0,
                "field_type": "string"
            }
        },
        {
            "type": "field",
            "location": {
                "begin": 933,
                "end": 938
            },
            "properties": {
                "field_name": "label",
                "field_index": 1,
                "field_type": "string"
            }
        },
        {
            "type": "field",
            "location": {
                "begin": 938,
                "end": 958
            },
            "properties": {
                "field_name": "label",
```

```
                "field_index":  2,
                "field_type":  "string"
            }
        },
        {
            "type":  "field",
            "location":  {
                "begin":  958,
                "end":  1059
            },
            "properties":  {
                "field_name":  "body",
                "field_index":  0,
                "field_type":  "string"
            }
        },
        {
            "type":  "field",
            "location":  {
                "begin":  1059,
                "end":  1230
            },
            "properties":  {
                "field_name":  "nested",
                "field_index":  0,
                "field_type":  "json"
            }
        },
        {
            "type":  "field",
            "location":  {
                "begin":  1230,
                "end":  1243
            },
            "properties":  {
                "field_name":  "claim_date",
                "field_index":  0,
                "field_type":  "date"
            }
        }
    ]
}
```


Following are the binary file properties:

Property	Type	Description
`document_id`	`string`	The identifier of the document.
`location_encoding`	`string`	The encoding type used to calculate the location of each feature. The supported types are: `utf-8`, `utf-16`, and `utf-32`. The external enrichment application must calculate the location of each feature based on the `location_encoding` of the corresponding document from Discovery. The location of features in a string representation of data varies depending on the encoding type of the programming language that is used for implementing the external enrichment. For example, C++ and Go use UTF-8, Java and JavaScript use UTF-16, and Python uses UTF-32.
`language`	`string`	The content language of the document.
`artifact`	`string`	The package of all the text values.
`features`	`array`	The list of features in a document. For more information, see Feature types.

Pull method binary file properties

Binary attachment in the push batches method

After external enrichment, the documents can be pushed to Discovery as a binary attachment in the `push batches` method.

The file must be a compressed NDJSON file with structured data that represents the document properties. For example, the following is an NDJSON file:


```
{
   "document_id":  "3bafc09abfaacd90d66f57181b50d041",
```

```
"features": [
  {
    "type": "annotation",
    "location": {
      "begin": 958,
      "end": 1000
    },
    "properties": {
      "type": "element_classes",
      "class_name": "expression",
      "confidence": 0.7905777096748352
    }
  },
  {
    "type": "annotation",
    "location": {
      "begin": 1001,
      "end": 1059
    },
    "properties": {
      "type": "element_classes",
      "class_name": "question",
      "confidence": 0.9507029056549072
    }
  },
  {
    "type": "annotation",
    "location": {
      "begin": 1035,
      "end": 1040
    },
    "properties": {
      "type": "entities",
      "entity_type": "JobTitle",
      "entity_text": "actor",
      "confidence": 0.70953685
    }
  },
  {
    "type": "annotation",
    "properties": {
      "type": "document_classes",
      "class_name": "amount.shortage",
      "confidence": 0.43297016620635986
    }
  },
  {
    "type": "notice",
    "properties": {
      "description": "something wrong happened",
    }
  },
  {
    "type": "notice",
    "properties": {
      "description": "something wrong happened again",
      "created": 1689076276402,
    }
  }
 ]
}
```


Following are the binary file properties:

Property	Type	Description
`document_id`	`string`	The identifier of the document.
`features`	`array`	The list of features in a document. For more information, see Feature types.

Push method binary file properties

Feature types

A feature `type` can be one of the following in a binary file:

Feature	Type	Description
`field`	`string`	Represents a specific field value of the document.
`annotation`	`string`	Represents a specific annotation that can enrich the document.
`notice`	`string`	Represents any error that might occur in the external application during document enrichment. The information in `notice` is used to generate a message on the Discovery UI.

Feature types

The following are the other properties in the binary file:

Feature	Type	Description
`location`	`object`	Location information to get the text value from the `artifact` by using the `begin` and `end` values. The `begin` value is a string value that represents the begin location in the artifact. The `end` value is a string value that represents an exclusive end location in the artifact. This property is null when a feature represents a document level information. For example, when `type=annotation` and `properties.type=document_classes`.
`properties`	`object`	The properties of a feature in the document. Supported properties vary depending on the `type` of feature. For more information, see Field type properties, Annotation type properties, and Notice type properties.

Other properties in the binary file

Field type properties

For `field` type, the following properties represent a certain field of the document that was converted by Discovery from an original file:

Property	Type	Description
`field_name`	`string`	The name of the field.
`field_index`	`int`	The index of a field value. This value is `0` for a single-valued field, but can be `> 0` when a field is multi-valued, such as, for an array of values.
`field_type`	`string` (enum: `long`, `double`, `date`, `json`)	The data type of the feature. This value determines how to parse the text representation of the feature in a programming language.

Field type properties

Annotation type properties

For `annotation` type, the following properties represent an annotation that can enrich a document:

Property	Type	Description
`type`	`string` (enum: `entities`, `element_classes`, `document_classes`)	The type of enriched annotation that a feature represents. The `entities` are merged to entities of enriched fields. The `element_classes` are merged to element classes of enriched fields. The `document_classes` are merged to classes of document level enrichment field.
`confidence`	`double`	The optional confidence score by the external model. It is between `0` to `1`, and is `0` by default.
`entity_type`	`string`	The type of entity that an external model assigns to a thing. Required for the `entities` type.
`entity_text`	`string`	The representative text of an entity that the external application extracts. Required for the `entities` type.
`class_name`	`string`	The name of a class that the external application assigns to a thing. Required for the `element_classes` and `document_classes` type.

Annotation type properties

Notice type properties

For `notice` type, the following properties represent errors and exceptions that occurred in the external application while enriching a document:

Property	Type	Description
`description`	`string`	The message that describes an error that occurred during external enrichment.
`created`	`long`	Unix epoch time in milliseconds when an error occurred during external enrichment.

Notice type properties

Testing and sharing your project

As you improve your project, periodically test how enrichments and search setting changes impact the query results.

For all project types except Conversational Search, you can see the fields that are associated with an indexed document by looking at the JSON view of a document that is returned by a query. Checking the JSON structure of a document can be useful if you want to check whether certain types of information are being captured.

After you enrich your collection, you can use the JSON view of a query result to check whether your enrichments are being applied and retrieved properly. For example, you can check the JSON to confirm that a synonym that you defined in a dictionary is being tagged as an occurrence of the corresponding dictionary term.

To test your project, complete the following steps:

1. From the navigation panel, open the **Improve and customize** page.
2. Retrieve query results by doing one of the following things:
   - *Content Mining* project: Choose or add a facet to apply to the documents, and then click **View filtered documents**. To analyze your data in more depth, open the Content Mining application in a new window or tab by clicking **Launch application**.
   - Other project types: Enter a test query to submit or leave the field empty and press Enter to submit an empty query.
3. From the query result list, click the link to view the document.
   A representation of the original document is displayed.
4. IBM Cloud Click **Open advanced view** to see useful summary information, such as the number of occurrences of any enrichments that are detected in the document.

   Optional: Select an enrichment to highlight every occurrence of the element within the document text.

   For a *Document Retrieval for Contracts* project, the *Contract Data* page is displayed. For more information about Contract filter options, see Understanding contracts.
5. You can learn more about information that is identified by the enrichments that are applied to your documents by reviewing the JSON representation of a document that is returned in a search result.

   IBM Cloud

   1. Click the *Display options* menu from the advanced view header, and then select **JSON**.

   IBM Cloud Pak for Data

   1. Click **JSON**.

      For a *Document Retrieval for Contracts* project, click the **Contract Data** tab. For more information about Contract filter options, see Understanding contracts.

Sharing your project

Try out your project and share it with others on your team for testing purposes. A test implementation of your project is created and hosted by IBM. Use this application preview to test your search results.

To preview and share your project, complete the following steps:

1. From the **Integrate and Deploy** > **Preview Link** page, follow the instructions to give your team members access to your project. (In Content Mining projects, the page is named **Share Link**.)

   IBM Cloud For more information about access in IBM Cloud, see Managing access to resources.
2. Click the copy icon for the **Copy Link** field to copy the URL of the preview application.
3. Paste the URL into a web browser to test it yourself or send the URL to team members.

   Don't forget to send any login credentials that are needed to access the project when you send the link to your colleagues.

Querying your data programmatically

Building custom applications with the API

Use the Discovery API to build a custom application or component that searches your data.

Service API Versioning

API requests require a version parameter that takes a date in the format `version=YYYY-MM-DD`. Whenever a backwards-incompatible API change occurs, a new minor version of the API is released.

Send the version parameter with every API request. The service uses the API version for the date you specify, or the most recent version before that date. Don't default to the current date. Instead, specify a date that matches a version that is compatible with your app, and don't change it until your app is ready for a later version.

The current version is `2023-03-31`.

Getting your project ID IBM Cloud

⚠ **Important:** This information applies to IBM Cloud only.

To use the API, you must construct the URL to use in your requests. Many of the API methods require the project ID.

1. From the IBM Cloud Resource list, expand *AI/Machine Learning*, and then find the service page for your Discovery service instance.
2. From the *Credentials* section, copy the URL. You specify this value as the `{url}` in your API requests.
3. While you're on this page, copy the API key. You specify this value as the `{apikey}`.
4. Open your project in Discovery, and then go to the **Integrate and deploy** > **API Information** page.
5. Copy the project ID. You specify this value as the `{project_id}`.

   If you're using a Content Mining project, stay on the *Share Link* page. From the web browser's *location* field, copy the URL starting with `/projects`. For example, `projects/a8ce5fed-7f33-4405-aa4b-88ffba322712/deploy/beta`. The ID that is specified after the `/projects/` segment of the URL is your project ID.
6. Construct a request URL by using the IDs you copied.

   For example, the following request lists the collections in the project:

   ```
   curl -X {request_method} -u "apikey:{apikey}" \
   "{url}/v2/projects/{project_id}/collections?version=2019-11-29 -k"
   ```

To get the `{collection_id}`, you can use the List collections API method. Alternatively, open the collection in the product user interface, and then copy the collection ID, which is displayed after the `/collections/` segment of the page URL, from the web browser location field.

Using the API from Cloud Pak for Data IBM Cloud Pak for Data

⚠ **Important:** This information applies to Discovery for Cloud Pak for Data only.

To use the API, you must construct the URL to use in your requests.

1. From the IBM Cloud Pak for Data web client main menu, expand **Services**, and then click **Instances**.
2. Find your instance, and then click it to open its summary page.
3. Scroll to the *Access information* section of the page, and then copy the URL. You will specify this value as the `{url}`.
4. Copy the bearer token also. You will need to pass the token when you make an API call.
5. From the launched application instance, go to the **Integrate and Deploy** > **API Information** page.
6. Copy the project ID. You will specify this value as the `{project_id}`.

   If you're using a Content Mining project, stay on the *Share Link* page. From the web browser's *location* field, copy the URL starting with `/projects`. For example, `projects/a8ce5fed-7f33-4405-aa4b-88ffba322712/deploy/beta`. The ID that is specified after the `/projects/` segment of the URL is your project ID.
7. Construct a request URL by using the IDs you copied.

For example, the following request lists the collections in the project:

```
curl -H "Authorization: Bearer {token}" \
"{url}/v2/projects/{project_id}/collections?version=2019-11-29 -k"
```

Important: The bearer token that is generated for an administrator can access any instance regardless of the access settings that are configured for the instance.

The bearer token expires after 12 hours. For more information about customizing the length of a session, see Setting the idle session timeout.

Next steps

A developer can make the following enhancements:

- Use the API to define more complex queries with the Discovery Query Language.
- Specify the exact document to return in response to a specific query with curations.
- Process documents without storing them in a collection by using the Analyze API.

Query API

Query overview

IBM Watson® Discovery offers powerful content search capabilities through search queries.

To retrieve data from Discovery after it is ingested, indexed, and enriched, submit a query.

As data is added to Discovery, a representation of each file is stored in the index as a JSON-formatted document. Enrichments that are applied to your collections identify meaningful information in the data and store it in new fields in these documents. To search your data, submit a query to return the most relevant documents and extract the information you're looking for.

Query types

Discovery accepts one of the following supported query types:

Query

Finds documents with values of interest in specific fields in your documents. Queries of this type use Discovery Query Language syntax to define the search criteria.

Parameter name: `query`

Natural Language Query (NLQ)

Finds answers to queries that are written in natural language. NLQ requests accept a text string value.

Parameter name: `natural_language_query`

Along with the query that you specify by using one of the supported query types, you can include one or both of the following parameters. The values for these parameters are also specified by using the Discovery Query Language (DQL) syntax:

- `filter`
- `aggregation`

For more information about the Discovery Query Language, see DQL overview.

Queries that are submitted from the product user interface are natural language queries. A few other supported parameters are specified and given default values based on the project type in use. For more information, see Default query settings.

Note: Discovery does not log query request data. You cannot opt in to request logging.

Choosing the right query type

The following table summarizes the capabilities that are supported for each query type. Use it to help you determine which type of query to submit.

Goal	Natural Language Query (NLQ)	Discovery Query Language (DQL)

Return passages from documents	✓	✓
Highlight terms in responses (unless passages per document is enabled)	✓	✓
Define custom stop words or query expansions	✓	✓
Search specific document fields or enrichments		✓
Use operators, such as boolean clauses in the search		✓
Enable spelling correction	✓	
Add curations to return hardcoded answers to certain questions	✓	
Use relevancy training	✓	
Enable answer finding to return a succinct answer from a passage	✓	
Use table retrieval	✓	

Query types comparison

Query analysis

When you submit a query, the query text string is analyzed. During query analysis, the root (or lemma) of each key term in the query is identified. Any stop words that occur in the original query string are removed and synonym expansions that are defined for any terms that occur in the original query string are added. This enhanced version of the query is what gets submitted to Discovery.

The same analysis is performed on all queries, whether they are submitted as natural language queries or by using Discovery Query Language syntax.

Query flow

The following diagram shows a conceptual illustration of how a search request is handled by Discovery.

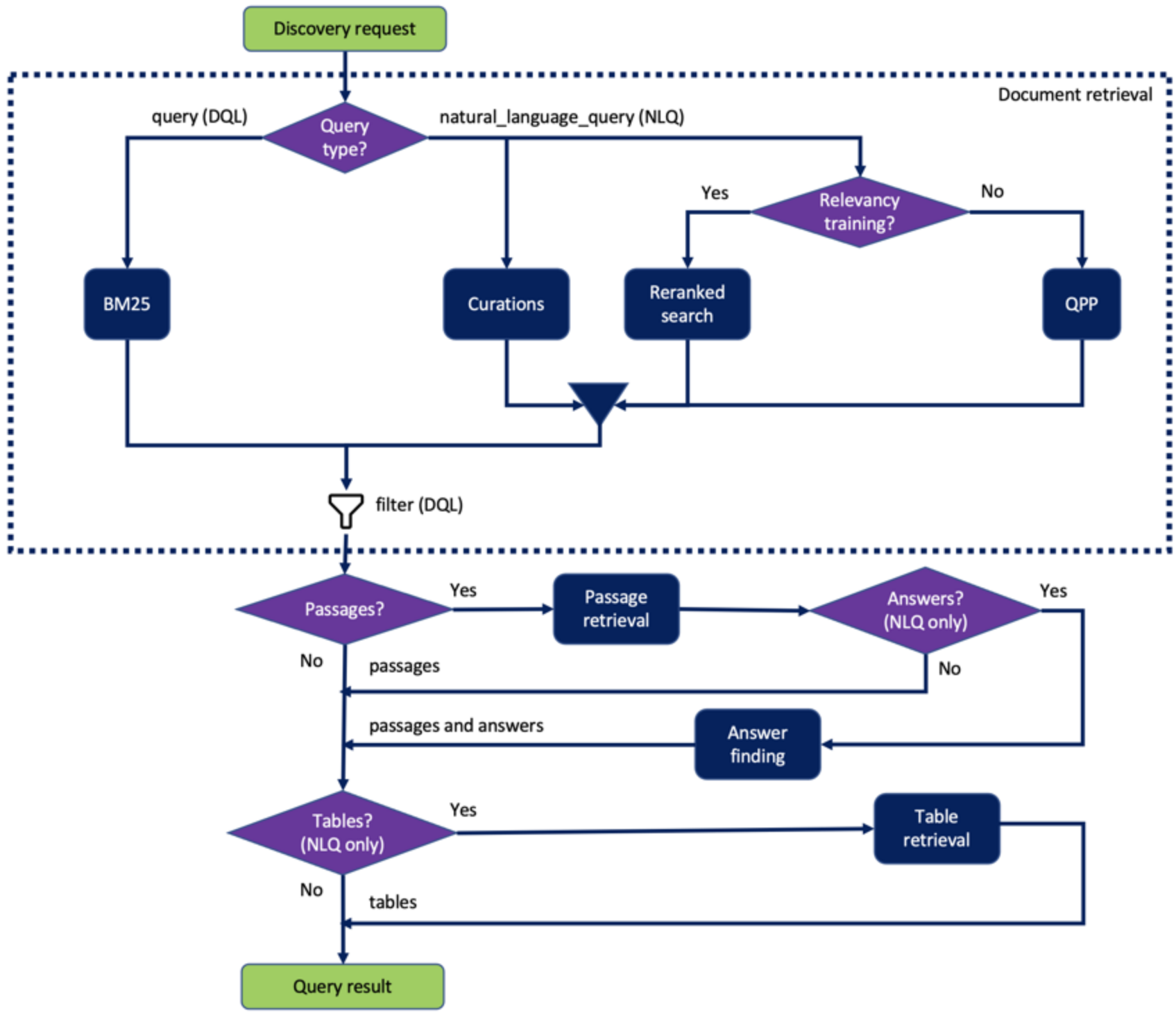


Figure 1. Flow chart that shows the processes that are used for Natural Language Queries versus Discovery Query Language queries

The following processes are shown in the flow diagram:

BM25

Uses Best Match 25 (a probabilistic information retrieval algorithm) to compute a relevance score for each document returned by search. The diagram shows that BM25 is applied to document results from the query requests, but it is not limited to query requests. It also is used along with other techniques as part of the relevancy training ranker process that is applied to natural language query results.

Curations

If the natural language query matches a predefined curation query, then certain documents and possibly a hardcoded snippet are returned. There is no query parameter to enable a curation. For curations to be used, you must define them programmatically (Create curation method). The output of any curations is merged with the output of the Relevancy training ranker or QPP results.

Relevancy training

A model that you can optionally define and apply to a project to score documents for relevance. There is no query parameter to enable relevancy training. For relevancy training to be used, you must successfully train the project either programmatically (Create training query method) or by using the product user interface.

QPP

A Query Performance Prediction algorithm that, given a query and a list of top results, produces a score that determines how relevant a document is. Used only if no Relevancy training ranker is available.

filter

The `filter` parameter can be passed along with `query` and `natural_language_query` requests to remove documents that don't meet certain criteria from the result set. The filter is shown as the last step within the document retrieval phase. However, it is used at different times in the flow. Its placement in the diagram is chosen to emphasize the fact that any documents that don't match the filter definition are excluded from the result set. The exclusion applies even to documents that might be specified in a curation.

Passage retrieval

Returns passages from documents when the `passages.enabled=true` parameter is included with a natural language query request.

Answer finding

When the `passages.find_answers=true` parameter is included with a natural language query request, returns succinct answers from passages along with the passages that are extracted from documents. If answer finding is enabled, then the final confidence score for each search result is a combination of the confidence scores from answer finding, passage retrieval, and QPP or Reranked search, whichever method is used.

Table retrieval

Returns information from tables in documents when the `table_results.enabled=true` parameter is included with a natural language query request.

Query limits

A query is any operation that submits a `POST` request to the `/query` endpoint of the API. Such operations include queries that are submitted by using the API. It does not include queries that are submitted from the search bar on the *Improve and customize* page of the product user interface.

A query is counted only if the request is successful, meaning it returns a response (with message code 200).

The number of search queries that you can submit per month per service instance depends on your Discovery plan type.

Plan	Queries per month per service instance
Cloud Pak for Data	Unlimited
Premium	Unlimited
Enterprise	Unlimited
Plus (includes Trial)	500,000

Number of queries per month

Note: For Enterprise plans only, your bill labels requests that are generated from both query searches and analyze API calls as "Queries". For more information about Analyze API calls, see Analyze API limits.

The number of queries that can be processed per second per service instance depends on your Discovery plan type.

Plan	Concurrent queries per service instance
Cloud Pak for Data	Unlimited
Premium	50
Enterprise	5
Plus (includes Trial)	5

Number of concurrent queries

For information about pricing, see Discovery pricing plans.

Estimating query usage

How to estimate the number of queries your application will use per month depends on your use case.

- For use cases that focus more on data enrichment and analysis or where the output from the document processing is not heavily searched, you can estimate query numbers based on the total number of documents.
- For use cases where many users interact with the application that uses Discovery, you can estimate by calculating the number of searches per user times the number of expected users. For example, 50% of the questions that are submitted by users to a virtual assistant are likely to be answered by Discovery. With 100,000 users per month and an average of 3 questions per user, you can expect 15,000 queries per month. (10,000 users/mo * 3 queries/user * 50% to Discovery = 15,000)

Querying with document-level security enabled

IBM Cloud Pak for Data **IBM Cloud Pak for Data only**

Note: This information applies only to installed deployments.

If you enable document-level security for a collection, only documents that the current user has permission to access are returned in search results. For more information, see Configuring document-level security.

To return search results that adhere to the security restrictions, the current user must meet these requirements:

- Have access to your Discovery instance.
- Have access to the data source.

If the current user does not meet these requirements, no search results are returned.

The username that is associated with your Discovery instance is used to generate an authorization token. The token is used to authenticate Discovery queries.

To generate each access token, run the following command:

```
$ curl -u "{username}:{password}" \
"https://{hostname}:{port}/v1/preauth/validateAuth"
```

Replace `{username}` and `{password}` with the user's Discovery credentials.

Use the bearer token that is associated with the user when you run the query.

```
$ curl -H "Authorization: Bearer {token}" \
'https://{hostname}/{instance_name}/v2/projects/{project_id}/collections/{Collection_ID}/query\?version\=2019-11-29'
```

DQL overview

The Discovery Query Language defines syntax you can use to filter, search, and analyze your data.

How to write a Discovery Query Language query

The Discovery Query Language leverages the structure of indexed documents. The following JSON snippet shows an indexed document from a collection where the *Entities* enrichment is applied. As a result of the enrichment, the JSON structure captures any mentions of known entities, such as city names, companies, or famous people.

In this example, the recognized entity is the company name `IBM`.


```
{
   "document":  {
      "document_id":  "f7f27ea30eb3e4c0ce21830618d9ee99",
      "enriched_text":  [
         {
            "entities":  [
               {
                  "model_name":  "natural_language_understanding",
                  "mentions":  [],
                  "text":"IBM",
                  "type":"Organization"
               }
            ]
         }
      ]
   }
}
```


To create a query that returns all of the documents in which the entity `IBM` is mentioned, use the following syntax:

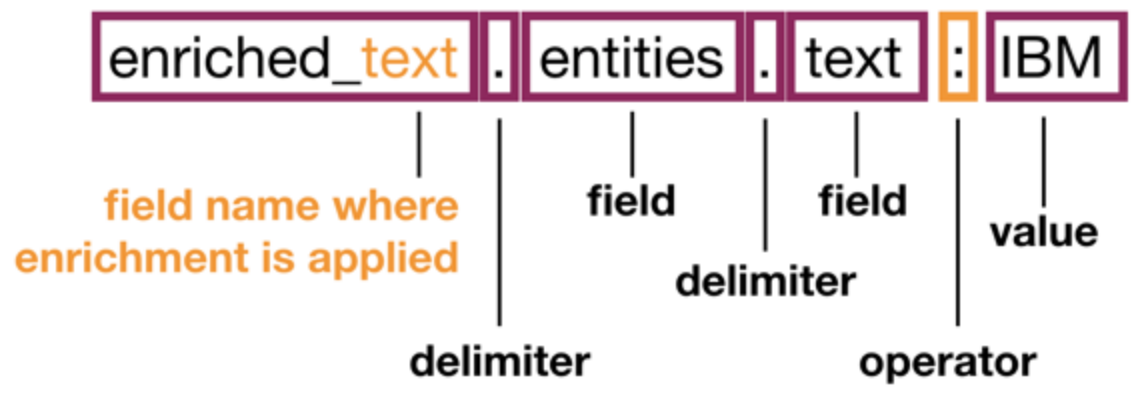


Figure 1. Example query structure

This basic query contains a nested path expression before the `:` operator. Each path element is the name of a field in the document separated by a period (`.`). The `:` operator indicates that the text that follows the operator must be included in the result.

The `::` operator indicates that the text must be matched exactly in the result. For more information, see Query operators. You can see how the two operators are used in the following examples.

- To return matching documents in order of relevance, pass the following data object in the `POST` request:

```
{
   "query":"enriched_text.entities.text:IBM"
}
```

- To return matching documents in any order, pass the following data object in the `POST` request as the query body:

```
{
   "filter":"enriched_text.entities.text::IBM"
}
```

Using the filter and query parameters together

The `filter` parameter returns faster than the `query` parameter and its results are cached. If you submit queries that use the `filter` and `query` parameters separately on a small data set, each request returns similar (if not identical) results.

In large data sets, if you need results to be returned in order of relevance, combine the `filter` and `query` parameters. Using the parameters together improves performance because the `filter` parameter is applied first. It filters the documents and caches the results. The `query` parameter then ranks the cached results.

Filter example: Get a document by its ID

Query body:

```
{
   "filter":  "document_id::b6d8c6e3-1097-421b-9e39-75717d2554aa"
}
```

If the document exists, the query returns 1 matching result. If it doesn't, the query returns no matching results.

Filter example: Find a document ID by its file name

If you don't know the `document_id` of a document, but you know the original `filename` of the document, you can use the `filter` and `return` parameters together to discover the `document_id`.

Query body:

```
{
   "filter":  "extracted_metadata.filename::100674.txt",
   "return":  [  "document_id",  "extracted_metadata"  ]
}
```

Response:

```
{
   "matching_results":  1,
   "results":  [
      {
         "document_id":  "b6d8c6e3-1097-421b-9e39-75717d2554aa",
         "extracted_metadata":  {
            "sha1":  "AD447F7592A17CDCBF0A589C4E6EC2087AF7H35F",
            "filename":  "100674.txt",
            "file_type":  "text"
         }
      }
   ]
}
```

Filter example: Find documents that mention an entity value

The query looks for documents that mention the entity `Gilroy` and finds 4 matching documents.

Query body:

```
{
   "filter":  "enriched_text.entities.text::Gilroy"
}
```

Response:

```
{
   "matching_results":  4
}
```

Filtering nested values

You can nest one filter inside another to ensure that the documents that are returned match more than one condition.

In the documents used for these examples, the entity `"Gilroy"` appears as both a `"Location"` (a town in California) and as a `"Person"` (a surname) entity type. To find documents where `"Gilroy"` appears as a location, write a query that filters on two nested fields at the same time: the entity text must be `"Gilroy"` and the entity type must be `"Location"`.

One way to write the query is as follows:

```
{
   "filter":  "enriched_text.entities.text::Gilroy,enriched_text.entities.type::Location"
}
```

This query matches documents where some path `enriched_text.entities.text` is `Gilroy` and some path `enriched_text.entities.type::Location` is `Location`. However, there is no guarantee that those two paths will be under the same `entities` object. For example, the query matches documents that have `Gilroy` as a `Person` entity type and, at the same time, have some other `Location` entity type object.

To accurately capture the nested semantics of this query, nest the filter values by using the following syntax:

Query body:

```
{
   "filter":  "enriched_text.entities:(text::Gilroy,type::Location)"
}
```

This stricter query matches only those documents in which there is an `entities` object with `text` equal to `Gilroy` and `type` equal to `Location`.

As another example, if you want to match documents that contain an `entities` object with `text` equal to `Gilroy` but `type` **not** equal to `Location`, you can use the *not equal* operator in the query, for example:

```
{
   "filter":  "enriched_text.entities:(text::Gilroy,type::!Location)"
}
```

You can also use aggregations to do more sophistocated filtering of the results. For more information about the available aggregation types, see Query aggregations.

For more information about the Discovery Query Language, see the following topics:

- Query parameters
- Query operators

Query parameters

You can use these parameters when you write queries with the Discovery Query Language. For more information, see the Discovery API reference. For an overview of query concepts, see the Query overview.

Queries that are written in the Discovery Query Language can include both search and structure parameters.

The default values for query parameters can differ by project type. For more information about default values, see Default query settings.

Search parameters

Use search parameters to search your collection, identify a result set, and analyze the result set.

The **results set** is the group of documents that are identified by the combined searches of the search parameters. The results set might be significantly larger than the returned results. If an empty query is submitted, the results set is equal to all the documents in the collection.

⚠ **Important:** Documents that you do not have permissions to access are not returned in query results.

Answer finding

IBM Cloud The `find_answers` parameter is supported in managed deployments only.

By default, Discovery provides answers by returning the entire passage that contains the answer to a natural language query. When the answer-finding feature is enabled, Discovery also provides a "short answer" within the passage, and a confidence score to show whether the "short answer" answers the question that is explicit or implicit in the user query. Applications that use the answer-finding feature can display the short answer alone or can display the short answer emphasized in the context of the full passage. For most applications, displaying the short answer emphasized within the full passage is preferable, because answers generally make more sense in context.

The answer finding feature behaves in the following ways:

Note: In the passage examples that follow, the short answers are shown in bold font.

- Finds answers. It doesn’t create answers. The answer must be part of the text; it can't be inferred.

  “What was IBM’s revenue in 2022?” can get a correct answer if you have a document that states what IBM’s revenue was in 2022. However, if you have a document that lists what IBM’s revenue was in each quarter of 2022, it doesn't add them up and give you a total.

- Handles synonyms and lexical variations if the answer is available.
  - Example question: “When did IBM purchase Red Hat?”
  - Passage: “IBM closed its $34 billion acquisition of Red Hat in **July of 2019**."
- Combines information across multiple sentences if they are close together (within approximately 2,000 characters).
  - Example question: “When did IBM purchase Red Hat?”
  - Passage: “IBM acquired Red Hat for $34 billion. The deal closed in **July of 2019**."
- Handles implicit questions similar to the way it would handle the equivalent explicit question.

  Example questions:

  - `company that developed the AS/400`
  - `What company developed the AS/400?`
- Works well with questions with longer phrase or clause answers.
  - Example question: How do I flip a pancake?
  - Passage: The key to getting a world-class pancake is flipping it properly. The best way to flip a pancake is to **stick a spatula under it, lift it at least 4 inches in the air, and quickly rotate the spatula 180 degrees.**
- Many how or why questions are only fully answered by much longer spans of text. The answer-finding feature does not return a whole document as the answer (and it doesn't summarize a document length answer).
- Handles yes or no questions that are factual and have a concise answer in the text
  - Example question: Is there a library in Timbuktu
  - Passage: Timbuktu's **main library, officially called the Ahmed Baba Institute of Higher Islamic Studies and Research** , is a treasure house that contains more than 20,000 manuscripts that cover centuries of Mali's history.
- Handles questions with very short answers, such as names and dates, especially when the type of answer that is required is explicit in the text.
- Handles opinion questions, but only by finding a statement of that opinion; it does not assess the validity of the opinion.
  - Example question: Should I try blue eyeshadow?
  - Passage: We think **blue eye shadow is trending** this year.

How the answer-finding feature works

After a user submits a query, the query is analyzed by the Discovery service. Query analysis transforms the user's original query in ways that improve the chances of finding the best search results. For example, it lemmatizing words, removes stop words, and adds query expansions. The search is performed and the resulting documents and passages are returned.

Answer finding is applied to the returned passages. Up to 60 passages are sent to the answer-finding service. How these 60 passages are chosen differs based on the `passages.per_document` parameter value.

- If `passages.per_document` is `false`, the top 60 passages from all of the documents that are returned by search are chosen based on their passage scores only.
- If `passages.per_document` is `true`, the returned documents are ranked first, and then the top 60 passages from these top documents are chosen.

For example, if you set the query to return 100 documents (count=100) and ask for 2 passages from each document (passages.max_per_document=2), then 2 passages are chosen from each of the 30 top-ranked documents (2 x 30 = 60 passages) only. No passages are chosen from the remaining 70 documents.

Tip: If your goal is to get the best 10 short answers, a good approach is to give the answer-finding feature various passages from more documents than just the top 10. To do so, set `passages.per_document` to `true`, and then request 20 documents and up to 3 passages from each document with the answer-finding feature enabled. The answer-finding feature searches for answers in up to 20*3 = 60 passages.

Answer finding does not use the transformed query string that is generated by query analysis. Instead, it uses a copy of the user's original input that is stored at query time to find the best short answer. If the answer-finding module is confident that it found an answer in one of the passages, the answer confidence score is combined with the document and passage scores to produce a final ranking, which can promote a document or passage that might otherwise be missed.

Answer-finding API details

The answer-finding API adds the following parameters to the `passage` section of the query API:

- `find_answers` is optional and defaults to `false`. If it is set to `true` (and the `natural_language_query` parameter is set to a query string), the answer-finding feature is enabled.
- `max_answers_per_passage` is optional and defaults to `1`. In this case, the answer-finding feature finds the number of answers that are specified at most from any one passage.

A section is also added to the return value within each `passage` object. That section is called `answers`, and it is a list of answer objects. The list can be up to `max_answers_per_passage` in length. Each answer object contains the following fields:

- `answer_text` is the text of the concise answer to the query.
- `confidence` is a number between `0` and `1` that is an estimate of the probability that the answer is correct. Some answers have low confidence and are unlikely to be correct. Be selective about what you do with answers based on this value. The confidence and order of documents in the search results are adjusted based on this combination if the `per_document` parameter of passage retrieval is set to `true` (which is the default).
- `start_offset` is the start character offset (the index of the first character) of the answer within the field that the passage came from. It is greater than or equal to the start offset of the passage (since the answer must be within the passage).
- `end_offset` is the end character offset (the index of the last character, plus one) of the answer within the field that the passage came from. It is less than or equal to the end offset of the passage.

To find answers across the entire project:

- Set `passages.enabled` to `true`
- Set `passages.find_answers` to `true`

To find answers within a single known document (for example, a document review application with long, complex documents):

- Set `passages.enabled` to `true`
- Set `passages.find_answers` to `true`
- Set `filter` to select the `document_id` for the document

The following example shows a query that uses this API:

```
POST /v2/projects/{project_id}/query{
  "natural_language_query": "Why did Nixon resign?",
  "passages": {
    "enabled": true, "find_answers":true
  }
}
```

Example response:

```
{
  "matching_results": 74, "retrieval_details": { "document_retrieval_strategy": "untrained"},
  "results": [
    {
      "document_id": "63919442-7d5b-4cae-ab7e-56f58b1390fe",
      "result_metadata":{"collection_id": "collection_id1234","document_retrieval_source":"search","confidence": 0.78214},
      "metadata": {"parent_document_id": "63919442-7d5b-4cae-ab7e-56f58b1390fg"},
      "title": "Watergate scandal",
      "document_passages": [
        {
          "passage_text": "With his complicity in the cover-up made public and his political support completely eroded, Nixon resigned from office on August 9, 1974. It is believed that, had he not done so, he would have been impeached by the House and removed from office by a trial in the Senate.",
          "field": "text",
          "start_offset": 281,
          "end_offset": 553,
          "answers": [
```

```
                {
                   "answer_text":  "his complicity in the cover-up made public and his political support completely eroded",
                   "start_offset":  286,  "end_offset":  373,  "confidence":  0.78214
                }
             ]
          }
       ]
}
```


`natural_language_query`

Use a natural language query to enter queries that are expressed in natural language, as might be received from a user in a conversational or free-text interface, such as IBM Watson Assistant. The parameter uses the entire input as the query text. It does not recognize operators.

The maximum query string length for a natural language query is `2048`.

Result confidence scores

When the query type is a natural language query, each result has a confidence score. The confidence score is a measure of the relevancy of the result. Each query result is evaluated and scored independently.

A variety of techniques are used to evaluate confidence. One important factor is the frequency of word matches between the query and the document.

Because a variety of techniques are used in different contexts to evaluate the result, the number range of result scores can vary widely from query to query. This variability means that comparing the confidence score to a static threshold value is an unsuitable method by which to delimit the results that are returned by your application. Results are ordered from highest to lowest confidence. You can find the best candidate answers by taking the top results, regardless of their confidence score values.

The `natural_language_query` parameter enables capabilities such as relevancy training. For more information, see Improving result relevance with training.

`query`

A query search returns all documents in your data set with full enrichments and full text in order of relevance. A query also excludes any documents that don't mention the query content.

`aggregation`

Aggregation queries return a count of documents that match a set of data values. For the full list of aggregation options, see Query aggregations.

`filter`

A cacheable query that excludes any documents that don't mention the query content. Filter search results are **not** returned in order of relevance.

When you write a query that includes both a `filter`, and an `aggregation`, `query`, or `natural_language_query` parameter, the `filter` parameter runs first, and then any `aggregation`, `query`, or `natural_language_query` parameters run in parallel.

With a simple query, especially on a small data set, the `filter` and `query` parameters often return the exact same (or similar) results. If the `filter` and `query` calls return similar results, and you don't need the responses to be returned in order of relevance, use the `filter` parameter. Filter calls are faster and are cached. Caching means that the next time you make the same call, you get a much quicker response, particularly in a big data set.

Structure parameters

Structure parameters define the content and organization of the documents in the returned JSON. Structure parameters don't affect which documents are part of the entire results set.

`return`

A comma-separated list of the portion of the document hierarchy to return. Any of the document hierarchies are valid values. If this parameter is an empty list, then all fields are returned.

`count`

The number of documents that you want to return in the response. The default is `10`. The maximum for the `count` and `offset` values together in any one query is `10000`.

`offset`

Index value of the position of the search result where the set of results to return begins. For example, if the total number of results that are returned is 10, and the offset is 8, it returns the last two results. The default is `0`. The maximum allowed value for the `count` and `offset` together in any one query is `10000`.

`spell correction`

In natural language queries, checks the query that is submitted for misspelled terms. The query is processed as-is. However, likely corrections to the

original query, if any exist, are returned in the `suggested_query` field of the response. The suggestions are not used automatically, but your application can make use of them.

`sort`

A comma-separated list of fields in the document to sort by. You can optionally specify a sort direction by prefixing the field with `-` for descending order or `+` for ascending order. Ascending order is the default sort direction.

`highlight`

A Boolean value that specifies whether to include a `highlight` object in the returned output. When included, the highlight returns keys that are field names and values that are arrays. The arrays contain segments of query-matching text that is highlighted by using the HTML emphasis (`<em>`) tag.

This parameter is ignored if `passages.enabled` and `passages.per_document` are `true`, in which case passages are returned for each document instead of highlights.

> **Note:** Currently, if the query searches for an `exact match` of an enrichment mention, only lowercase matches are highlighted. When the `includes` operator is used, upper- and lowercase matches are highlighted.

The output lists the `highlight` object after the `enriched_text` object, as shown in the following example.

```
$ curl -H "Authorization: Bearer {token}" \
'https://{hostname}/{instance_name}/v2/projects/{project_id}/collections/{collection_id}/query?version=2019-11-
29&natural_language_query=Hybrid%20cloud%20companies&highlight=true'
```

The JSON that is returned has the following format:

```
{
   "highlight":  {
      "extracted_metadata.title":  [
         "IBM to Acquire Sanovi Technologies to Expand Disaster Recovery Services for <em>Hybrid</em> <em>Cloud</em>"
      ],
      "enriched_text.concepts.text":  [
         "Privately held <em>company</em>",
         "<em>Cloud</em> computing"
      ],
      "text":  [
         " Sanovi Technologies, a privately held <em>company</em> that provides <em>hybrid</em> <em>cloud</em> recovery, <em>cloud</em>
migration",
         "IBM to Acquire Sanovi Technologies to Expand Disaster Recovery Services for <em>Hybrid</em> <em>Cloud</em>\n\nPublished",
         " undergoing digital and <em>hybrid</em> <em>cloud</em> transformation.\n\nURL:
http://www.ibm.com/press/us/en/pressrelease/50837.wss",
         " and business continuity software for enterprise data centers and <em>cloud</em> infrastructure. Adding"
      ],
      "enriched_text.categories.label":  [
         "/business and industrial/<em>company</em>/bankruptcy"
      ],
      "enriched_text.entities.type":  [
         "<em>Company</em>"
      ],
      "html":  [
         " Technologies, a privately held <em>company</em> that provides <em>hybrid</em> <em>cloud</em>\n recovery, <em>cloud</em>
migration and business",
         " Disaster Recovery Services for <em>Hybrid</em> <em>Cloud</em></title></head>\n<body>\n\n\n<p>Published: Thu, 27 Oct 2016 07:01",
         " digital and <em>hybrid</em> <em>cloud</em> transformation.</p>\n<p>URL:
http://www.ibm.com/press/us/en/pressrelease/50837.wss</p>\n\n\n\n</body></html>",
         " continuity software for \nenterprise data centers and <em>cloud</em> infrastructure. Adding these \ncapabilities"
      ]
   }
}
```

`passages`

A Boolean that specifies whether the service returns a set of the most relevant passages from the documents that are returned by a query that uses the `natural_language_query` parameter. The passages are generated by sophisticated Watson algorithms that determine the best passages of text from all of the documents returned by the query. The default value for the parameter differs based on your project type. For more information about default values, see Default query settings.

Discovery attempts to return passages that start at the beginning of a sentence and stop at the end by using sentence boundary detection. To do so, it first searches for passages approximately the length specified in the `passages.characters` parameter (for most project types, the default is `200`). It then expands each passage to the limit of twice the specified length so as to return full sentences. If your `passages.characters` parameter is short or the sentences in your documents are long there might be no sentence boundaries close enough to return the full sentence without going over twice the requested length. In that case, Discovery stays within the limit of twice the `passages.characters` parameter, so the passages that are returned might

not include the entire sentence and can omit the beginning, end, or both.

Since sentence boundary adjustments expand passage size, the average passage length can increase. If your application has limited screen space, you might want to set a smaller value for `passages.characters` or truncate the passages that are returned by Discovery. Sentence boundary detection works for all supported languages and uses language-specific logic.

Passages are grouped with each document result and are ordered by passage relevance. Including passage retrieval in queries increases the response time because it takes more time to score the passages.

You can adjust the fields in the documents for passage retrieval to search with the `passages.fields` parameter.

The `passages` parameter returns matching passages (`passage_text`), and the `score` , `document_id` , the name of the field that the passage was extracted from (`field`), and the starting and ending characters of the passage text within the field (`start_offset` and `end_offset`), as shown in the following example.

```
$  curl -H "Authorization: Bearer {token}"  'https://{hostname}/{instance_name}/v2/projects/{project_id}/collections/{collection_id}/query?version=2019-11-29&natural_language_query=Hybrid%20cloud%20companies&passages=true&passages.per_document=false'
```

The JSON that is returned from the query has the following format:

```
{
   "matching_results":2,
   "passages":[
      {
         "document_id":"ab7be56bcc9476493516b511169739f0",
         "passage_score":15.230205287402338,
         "passage_text":"a privately held company that provides hybrid cloud recovery, cloud migration and business continuity software for enterprise data centers and cloud infrastructure.",
         "start_offset":120,
         "end_offset":300,
         "field":"text"
      },
      {
         "passage_text":"Disaster Recovery Services for Hybrid Cloud</title></head>\n<body>\n\n\n<p>Published: Thu, 27 Oct 2016 07:01:21 GMT</p>\n",
         "passage_score":10.153470191601558,
         "document_id":"fbb5dcb4d8a6a29f572ebdeb6fbed20e",
         "start_offset":70,
         "end_offset":120,
         "field":"html"
      }
   ]
}
```

`passages.fields`

A comma-separated list of fields in the index that passages are drawn from. If this parameter is not specified, then passages from all root-level fields are included.

You can specify fields in both the `return` and `passages.fields` parameters. When you specify both parameters, each with different values, they are treated separately.

For example, the request might include the parameters `"return": ["docno"]` and `"passages":{"fields": ["body"]`. The `body` field is specified in `passages.fields` , but not in `return` . In the result, passages from the document body are returned, but the contents of the body field itself is not returned.

`passages.count`

The maximum number of passages to return. The search returns fewer passages if the specified count is the total number found. The default value is `10` . The maximum value is `100` .

`passages.characters`

The approximate number of characters that any one passage can have. The default value is `200` . The minimum is `50` . The maximum is `2,000` . Passages that are returned can contain up to twice the requested length (if necessary) to get them to begin and end at sentence boundaries.

`passages.max_per_document`

One passage is returned per document by default. You can increase the maximum number of passages to return per document by specifying a higher number in the `passages.max_per_document` parameter.

`similar`

Finds documents that are similar to documents that you identify as being of interest to you. To find similar documents, Discovery identifies the 25 most relevant terms from the original document and then searches for documents with similar relevant terms.

If `similar.enabled` is `true`, you must specify the `similar.document_ids` field to include a comma-separated list of the documents of interest.

> **Note:** In installed deployments, support for this parameter was added with the 4.6.0 release.

`table retrieval`

If Table understanding is enabled in your collection, a `natural_language_query` finds tables with content or context that match a search query.

Example query:

```
$  curl -H "Authorization: Bearer {token}" \
 'https://{hostname}/{instance_name}/v2/projects/{project_id}/collections/{collection_id}/query?version=2019-11-
29&natural_language_query=interest%20appraised&table_results=true'
```

The JSON that is returned from the query has the following format:

```
{
   "matching_results":  1,
   "session_token":  "1_FDjAVkn9SW6oH9y5_9Ek3KsNFG",
   "results":  [
      {}
   ]
   {
      "table_results":  [
         {
            "table_id":  "e883d3df1d45251121cd3d5aef86e4edc9658b21",
            "source_document_id":  "c774c3df0c90255191cc0d4bb8b5e8edc6638d96",
            "collection_id":  "collection_id",
            "table_html":  "html snippet of the table info",
            "table_html_offset":  42500,
            "table":  [
               {
                  "location":  {
                     "begin":  42878,
                     "end":  44757
                  },
                  "text":  "Appraisal Premise Interest Appraised Date of Value Value Conclusion\nMarket Value \"As Is\" Fee Simple Estate January 12, 2016
$1,100,000\n",
                  "section_title":  {
                     "location":  {
                        "begin":  42300,
                        "end":  42323
                     },
                     "text":  "MARKET VALUE CONCLUSION"
                  },
                  "title":  {},
                  "table_headers":  [],
                  "row_headers":  [
                     {
                        "cell_id":  "rowHeader-42878-42896",
                        "location":  {
                           "begin":  42878,
                           "end":  42896
                        },
                        "text":  "Appraisal Premise",
                        "text_normalized":  "Appraisal Premise",
                        "row_index_begin":  0,
                        "row_index_end":  0,
                        "column_index_begin":  0,
                        "column_index_end":  0
                     }
                  ],
                  "column_headers":  [],
                  "body_cells":  [
                     {
                        "cell_id":  "bodyCell-43410-43424",
                        "location":  {
                           "begin":  43410,
                           "end":  43424
                        },
                        "text":  "Date of Value",
                        "row_index_begin":  0,
```

```
            "row_index_end":  0,
            "column_index_begin":  2,
            "column_index_end":  2,
            "row_header_ids":  [
               "rowHeader-42878-42896",
               "rowHeader-43145-43164"
            ],
            "row_header_texts":  [
               "Appraisal Premise",
               "Interest Appraised"
            ],
            "row_header_texts_normalized":  [
               "Appraisal Premise",
               "Interest Appraised"
            ],
            "column_header_ids":  [],
            "column_header_texts":  [],
            "column_header_texts_normalized":  [],
            "attributes":  []
         }
      ],
      "contexts":  [
         {
            "location":  {
               "begin":  44980,
               "end":  44996
            },
            "text":  "Compiled by CBRE"
         }
      ],
      "key_value_pairs":  []
   }
  ]
 }
```


`table_results.enabled`

When `true`, a `table_results` array is included in the response with a list of table objects that match the `natural_language_query` value in order of scored relevance. For all project types, except *Document Retrieval for Contracts*, the default value is `false`.

`table_results.count`

This parameter specifies the maximum number of tables that can be included in the `table_results` array. Only returned if `table_results.enabled` = `true`. The default value is `10`.

Query operators

You can use operators when you write queries to submit to Discovery by using the Query API.

The types of operators that are supported differ by query type:

- Natural language queries
- Discovery Query Language (DQL) queries

Natural Language Query (NLQ) operator

The `natural_language_query` parameter accepts a string value.

`""` (Phrase query)

Use quotation marks to emphasize a single word or phrase in the query that is most important to match. For example, the following request boosts documents that contain the term “nomination” in them.

```
{
   "natural_language_query":"What is the process for \"nomination\" of bonds?"
}
```

Specifying a quoted phrase does not prevent documents without the phrase from being returned. It merely gives more weight to documents with the phrase than those without it. For example, the query results might also contain documents that mention “bonds” or “process” and do not contain the word “nomination”.

The following request boosts the phrase “change in monetary policy” and also matches “change” or “monetary” or “policy”.

```
{
   "natural_language_query":"\"change in monetary policy\""
}
```

Single quotation marks (') are not supported. You cannot use wildcards (*) in phrase queries.

Discovery Query Language (DQL) operators

Operators are the separators between different parts of a query.

. (JSON delimiter)

This delimiter separates the levels of hierarchy in the JSON schema

For example, the following query argument identifies the section of the enriched_text object that contains entities and the text recognized as an entity.

```
enriched_text.entities.text
```

The JSON representation of this section looks as follows:

Figure 1. JSON representation of the enriched_text.entities.text field

: (Includes)

This operator specifies inclusion of the full query term.

For example, the following query searches for documents that contain the term `cloud computing` in the `text` field:

```
{
   "query":"enriched_text.entities.text:\"cloud computing\""
}
```

The **includes** operator does not return a partial match for the query term. If you want to find a partial match for a term, use a **wildcard** operator with the **includes** operator. For example, if you want to find any occurrences of `TP53` or `p53` in the `test_results` field, the following query will *not* find occurrences of both terms:

```
{
   "query":"test_results:P53"
}
```

Instead, include a wildcard in the request. For example, use the following query request. Because we are using the wildcard operator, we also changed the term to lowercase.

```
{
   "query":"test_results:*p53"
}
```

With this syntax, occurrences of `p53`, `tp53`, `P53`, or `TP53` are all returned.

"" (Phrase query)

Phrase queries only match occurrences of the whole phrase. The order of the words in the phrase must match.

For example, the following query returns only documents that contain a field named `quotation` with the text, `There's no crying in baseball`.

```
{
   "query":"quotation:\"There's no crying in baseball\""
}
```

A document with a `quotation` field that says `Jimmy Dugan said there's no crying in baseball` is also returned. However, documents that only mention `baseball` or `crying` without the entire phrase are not matched. Neither is a document with `In baseball, there's no`

`crying`. Documents that contain the right text in the wrong field also are not matched. For example, a document with the text `There's no crying in baseball` in the `text` field is not returned.

Single quotation marks (') are not supported. You cannot use wildcards (*) in phrase queries.

`::` (Exact match)

This operator specifies an exact match for the query term. Exact matches are case-sensitive.

For example, the following query searches for documents that contain entities of type `Organization`:

```
{
   "query":"enriched_text.entities.type::Organization"
}
```

The entire content of the field that you specify must match the phrase you specify. For example, the following query finds documents in which only entity mentions of `IBM Cloud` are detected, not `IBM Cloud Pak for Data` or `IBM cloud` or `Cloud`.

```
{
   "query":"enriched_text.entities.text::\"IBM Cloud\""
}
```

Note: Cannot match document fields that are greater than 256 characters in length.

`:!` (Does not include)

This operator specifies that the results do not contain a match for the query term.

For example:

```
{
   "query":"enriched_text.entities.text:!\"cloud computing\""
}
```

`::!` (Not an exact match)

This operator specifies that the results do not exactly match the query term.

For example:

```
{
   "query":"enriched_text.entities.text::!\"Cloud computing\""
}
```

Exact matches are case-sensitive.

Note: Will retrieve document fields matching the query term if the field is over 256 characters in length.

`\` (Escape character)

Escape character that preserves the literal value of the operator that follows it.

The complete list of valid escape sequences within text queries (except phrase queries):

`\"`, `\\`, `\(`, `\)`, `\[`, `\]`, `\,`, `\|`, `\^`, `\~`, `\:`, `\<=`, `\>=`, `\<`, `\>`, `\:!`, `\::`, `\::!`, `\*`, `\!`

For example, `message:\>=D`, `method::foo\(String\)`.

Within a phrase query, the only valid escape sequence is `\"`.

For example, `name:"Shane \"Rapha\" Hendrixson"`, `method::"foo(String)"`.

DQL is submitted to the Query API as JSON string fields, which require their own additional layer of escaping, for example:

```
{
   "query":"name:\"Shane \\\"Rapha\\\" Hendrixson\""
}
```

`()`,`[]` (Nested grouping)

Logical groupings can be formed to specify more specific information.

For example:

```
{
   "query":"enriched_text.entities:(text:IBM,type:Company)"
}
```

`|` (or)

Boolean operator for "or".

In the following example, documents in which `Google` or `IBM` are identified as entities are returned:

```
{
   "query":"enriched_text.entities.text:Google|enriched_text.entities.text:IBM"
}
```

Note: The includes (`:` , `:!`) and match (`::` , `::!`) operators have precedence over the `OR` operator.

For example, the following syntax searches for documents in which `Google` is identified as an entity or the string `IBM` is present:

```
{
   "query":"enriched_text.entities.text:Google|IBM"
}
```

It is treated as follows:

```
(enriched_text.entities.text:Google) OR IBM
```

`,` (and)

Boolean operator for "and".

In the following example, documents in which `Google` and `IBM` both are identified as entities are returned:

```
{
   "query":"enriched_text.entities.text:Google,enriched_text.entities.text:IBM"
}
```

Note: The includes (`:` , `:!`) and match (`::` , `::!`) operators have precedence over the `AND` operator.

For example, the following syntax searches for documents in which `Google` is identified as an entity and the string `IBM` is present:

```
{
   "query":"enriched_text.entities.text:Google,IBM"
}
```

It is treated as follows:

```
(enriched_text.entities.text:Google) AND IBM
```

`<=, >=, >, <` (Numerical comparisons)

Creates numerical comparisons of `less than` or `equal to` , `greater than` or `equal to` , `greater than` , and `less than` .

Only use numerical comparison operators when the value is a `number` or `date` .

Tip: Any value that is surrounded by quotations is a String. Therefore, `score>=0.5` is a valid query and `score>="0.5"` is not.

For example:

```
{
   "query":"invoice.total>100.50"
```

```
}
```

^x (Score multiplier)

Increases the score value of a search term.

For example:

```
{
   "query":"enriched_text.entities.text:IBM^3"
}
```

* (Wildcard)

Matches unknown characters in a search expression. Do not use capital letters with wildcards.

For example:

```
{
   "query":"enriched_text.entities.text:ib*"
}
```

~n (String variation)

The number of character differences that are allowed when matching a string. The maximum variation number that can be used is 2.

For example, the following query returns documents that contain `car` in the title field, as well as `cap`, `cat`, `can`, `sat`, and so on:

```
{
   "query":"title:cat~1"
}
```

The normalized version of the word is used for matching. Therefore, if the input contains "cats", the search looks for "cat", which is the normalized form of the plural cats.

When a phrase is submitted, each term in the phrase is allowed the specified number of variations. For example, the following input matches `cat dog` and `far log` in addition to `car hog`.

For example:

```
{
   "query":"title:\"car hog\"~1"
}
```

:* (Exists)

Used to return all results where the specified field exists.

For example:

```
{
   "query":"title:*"
}
```

:!* (Does not exist)

Used to return all results that do not include the specified field.

For example:

```
{
   "query":"title:!*"
}
```

For more information, see the Discovery API reference.

For an overview of query concepts, see the Query overview.

Query aggregations

Use aggregations to group, analyze, or compare results that are returned by a query request.

An aggregation is defined by an `aggregation` parameter that you can specify in the Query API. The input to the aggregation parameter is the document set that is returned from the `query`, `filter`, or `natural_language_query` parameter that is specified as a separate parameter in the same query request. Otherwise, the aggregation is applied to all of the documents in the project.

You can use an aggregation to do calculatations from values in the result document set. For example, to get information about the highest dollar amount in the `order.total` field of the documents that are returned as query results, use `max(order.total)` as the value of the `aggregation` parameter.

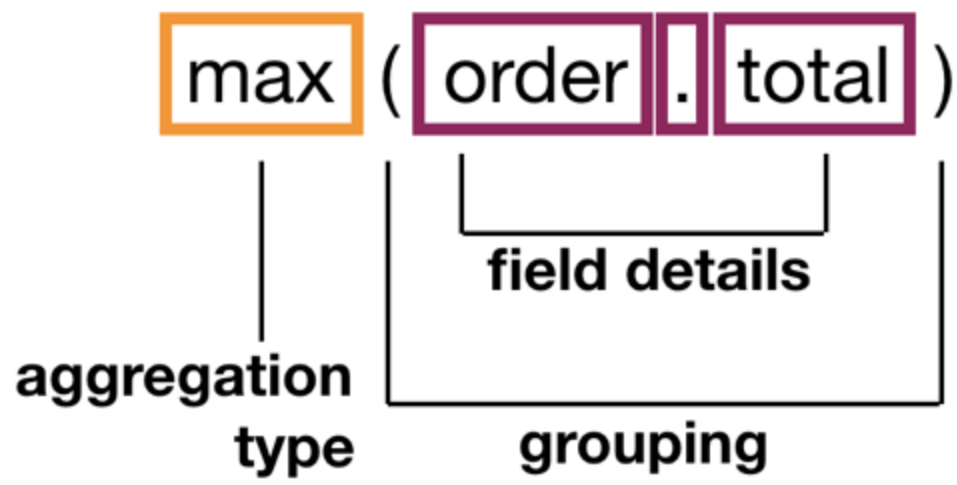


Figure 1. Aggregation query structure example

The aggregation parameter returns data about the field with the highest value.

```
"aggregations":  [
   {
      "type":  "max",
      "field":  "order.total",
      "value":  100668.00
   }
]
```

Grouping documents

In addition to doing calculations, you can use an aggregation to group documents in the result set that match certain values, so you can count them or analyze them further. For example, you can use an aggregation to search a set of traffic incident reports for documents that mention the term `brake`. And from the returned documents, find reports from the US states with the most relevant mentions of the term.

In the following example request, the count parameter that returns only 3 aggregation results is included to make the example easier to follow.

```
{
      "query":"brake",
      "aggregation":  "term(field:STATE,count:3,relevancy:true)"
}
```

The output of the aggregation parameter is returned in an `aggregations` object that is displayed before the `results` object, which contains the query results. A maximum of 50,000 values can be returned in the `aggregations` object for a single query.

The resulting `aggregations` object contains summary information about the query results. In this example, for instance, it shows that traffic incident reports from New York, California, and Florida have the most relevant mentions of the term `brake`.

```
{
   "matching_results":  9064,
   "retrieval_details":  {
      "document_retrieval_strategy":  "untrained"
   },
   "aggregations":  [
      {
         "type":  "term",
         "field":  "STATE",
         "results":  [
            {
               "key":  "NY",
               "matching_results":  693,
               "relevancy":  1.1649531567631084,
               "total_matching_documents":  2156,
               "estimated_matching_results":  542
            },
            {
               "key":  "CA",
               "matching_results":  1210,
```

```
            "relevancy": 1.1170819184294765,
            "total_matching_documents": 4017,
            "estimated_matching_results": 1011
         },
         {
            "key": "FL",
            "matching_results": 511,
            "relevancy": 0.828014956418841,
            "total_matching_documents": 2199,
            "estimated_matching_results": 553
         }
      ]
   }
],
"results": []
```


Combining aggregation types

There are different types of aggregations that you can use to analyze or group the query results. And you can combine more than one aggregation in a request to do more targeted analysis.

The following example shows a request that is composed of two term operators. The first term aggregation groups the input documents by US STATE values and selects 3 groups. The second term aggregation applies to each of those 3 groups and groups them further by the value of CITY. Only 2 of those CITY subgroups are returned per STATE group.

The relevancy parameter is being excluded to make the results easier to read.

```
{
      "query":"brake",
      "aggregation": "term(field:STATE,count:3).term(field:CITY,count:2)"
}
```

The response contains city information from each state.


```
{
   "matching_results": 9064,
   "retrieval_details": {
      "document_retrieval_strategy": "untrained"
   },
   "aggregations": [
      {
         "type": "term",
         "field": "STATE",
         "count": 3,
         "results": [
            {
               "key": "CA",
               "matching_results": 1210,
               "aggregations": [
                  {
                     "type": "term",
                     "field": "CITY",
                     "count": 2,
                     "results": [
                        {
                           "key": "LOS ANGELES",
                           "matching_results": 77
                        },
                        {
                           "key": "SAN DIEGO",
                           "matching_results": 66
                        }
                     ]
                  }
               ]
            },
            {
               "key": "NY",
               "matching_results": 693,
               "aggregations": [
                  {
                     "type": "term",
                     "field": "CITY",
                     "count": 2,
                     "results": [
                        {
```

```
                  "key":  "BROOKLYN",
                  "matching_results":  35
               },
               {
                  "key":  "NEW YORK",
                  "matching_results":  21
               }
            ]
         }
      ]
   },
   {
      "key":  "FL",
      "matching_results":  511,
      "aggregations":  [
         {
            "type":  "term",
            "field":  "CITY",
            "count":  2,
            "results":  [
               {
                  "key":  "JACKSONVILLE",
                  "matching_results":  33
               },
               {
                  "key":  "TAMPA",
                  "matching_results":  29
               }
            ]
         }
      ]
   },
   "results":  []
```


The order in which you specify the aggregations matters. For example, if you reverse the order of the term aggregations from the previous example, you get different results.

```
{
      "query":"brake",
      "aggregation":  "term(field:CITY,count:3).term(field:STATE,count:1)"
}
```

The new order produces results that surface Chicago, a city that wasn't included in the previous set of results. When the request starts by grouping by state, Illinois, which has only one city with a high number of traffic incident reports, is not included in the results. New York and Florida, which both have more than one city with many incident reports, produce a higher number of statewide matches and therefore, were returned. When you group by city first, the results change.


```
{
"matching_results":  9064,
"retrieval_details":  {
   "document_retrieval_strategy":  "untrained"
},
"aggregations":  [
   {
      "type":  "term",
      "field":  "CITY",
      "count":  4,
      "results":  [
         {
            "key":  "LOS ANGELES",
            "matching_results":  77,
            "aggregations":  [
               {
                  "type":  "term",
                  "field":  "STATE",
                  "count":  1,
                  "results":  [
                     {
                        "key":  "CA",
                        "matching_results":  77
                     }
                  ]
               }
```

```
            ]
         },
         {
            "key":  "SAN DIEGO",
            "matching_results":  66,
            "aggregations":  [
               {
                  "type":  "term",
                  "field":  "STATE",
                  "count":  1,
                  "results":  [
                     {
                        "key":  "CA",
                        "matching_results":  66
                     }
                  ]
               }
            ]
         },
         {
            "key":  "CHICAGO",
            "matching_results":  59,
            "aggregations":  [
               {
                  "type":  "term",
                  "field":  "STATE",
                  "count":  1,
                  "results":  [
                     {
                        "key":  "IL",
                        "matching_results":  59
                     }
                  ]
               }
            ]
         }
      ],
      "results":  []
```

Using aggregations to explore enrichments

The `term()` aggregation is especially useful for analyzing results to find out how many enrichments are recognized in the documents. For example, to count how many times each entity type is recognized in the filtered documents, you can submit the following query parameters:

```
{
   "filter":  "enriched_text.entities:(text::Gilroy,type::Location)",
   "aggregation":  "term(enriched_text.entities.type)"
}
```

The query first selects the documents that have at least one entity of type `Location` and whose text is `Gilroy`. This action returns 3 documents. From the returned documents, the aggregation then counts the number of documents in which each entity type appears.

```
{
   "matching_results":  3,
   "retrieval_details":  {
      "document_retrieval_strategy":  "untrained"
   },
   "aggregations":  [
      {
         "type":  "term",
         "field":  "enriched_text.entities.type",
         "results":  [
            {
               "key":  "Location",
               "matching_results":  3
            },
            {
               "key":  "Person",
               "matching_results":  3
            },
            {
               "key":  "Company",
               "matching_results":  2
            },
```

```
            {
               "key":  "GeographicFeature",
               "matching_results":  2
            },
            {
               "key":  "Organization",
               "matching_results":  2
            },
            {
               "key":  "Quantity",
               "matching_results":  2
            },
            {
               "key":  "Facility",
               "matching_results":  1
            },
            {
               "key":  "PrintMedia",
               "matching_results":  1
            }
         ]
      }
   ]
}
```

The 3 matching documents all have a `Location` and a `Person` entity type (`"matching_results": 3`). However, only 2 of the matching documents have a `Company` entity type.

By default, the top 10 matches are returned, sorted by relevance. You can change the number of results by adding the `count` parameter to the aggregation.

```
{
   "filter":  "enriched_text.entities:(text::Gilroy,type::Location)",
   "aggregation":  "term(enriched_text.entities.type,count:20)"
}
```

Add a filter

Use the `filter()` in the aggregation clause to filter results. For example, you can specify the same filter that was submitted separately in the previous example directly in the `aggregation` clause.

```
{
   "aggregation":  "filter(enriched_text.entities:(text::Gilroy,type::Location)).term(enriched_text.entities.type)"
}
```

In this case, the `filter().term()` aggregation finds the same result as the earlier example with the separate `filter` and `aggregation` clauses. However, results are ranked differently when the `filter` clause is used. You can leverage this difference by using the `filter()` clause within the `aggregation` clause to filter results from a sequence of expressions, as shown in the next example.

Start with nested objects

In the previous examples, the `"matching_counts"` value represents the number of documents that match the filter and aggregation. You might want to count how many *nested* objects are present in the query response. The `nested()` aggregation allows you to change the set of documents that is used as input to other aggregation terms.

For example, in the following query the `nested()` segment selects all `enriched_text.entities` nested objects as the input used by the `filter()` and `term()` segments.

```
{
   "aggregation":  "nested(enriched_text.entities).filter(enriched_text.entities.type::Organization).term(enriched_text.entities.text,count:3)"
}
```

The query results in an `aggregations` object that looks as follows:

```
{
   "aggregations":  [
      {
         "type":  "nested",
         "path":  "enriched_text.entities",
         "matching_results":  1993,
         "aggregations":  [
            {
```

```
            "type": "filter",
            "match": "enriched_text.entities.type::Organization",
            "matching_results": 645,
            "aggregations": [
              {
                "type": "term",
                "field": "enriched_text.entities.text",
                "count": 3,
                "results": [
                  {
                    "key": "IBM",
                    "matching_results": 36
                  },
                  {
                    "key": "Docker",
                    "matching_results": 12
                  },
                  {
                    "key": "OpenShift",
                    "matching_results": 12
                  }
                ]
              }
            ]
          }
        ]
      }
    ]
}
```


The `nested()` segment of the query found 1993 `enriched_text.entities` nested objects. The filter was applied to those objects and found 645 `enriched_text.entities` of type `Organization`.

Terminal operations

For most aggregation types, when you construct a query with multiple aggregation operations, the first operation is applied to the documents. Then, the output of that operation is used as the input for the next operation. However, a subset of the aggregation types are *terminal operations*. The output of a terminal operation is not used as input for the next aggregation. Instead, the output is returned in a discrete group.

For an example of a request that combines aggregation types and includes an aggregation that performs a terminal operation, see the second example for the `average` aggregation type.

Aggregation types

The following types of aggregations are supported:

- average
- filter
- group_by
- histogram
- max
- min
- nested
- pair
- sum
- term
- timeslice
- top_hits
- trend
- topic
- unique_count

For Document Retrieval project types, when you don't include an aggregation parameter in a query request, a default aggregation request is applied. For more information, see Document Retrieval project aggregations.

For more information about how to submit a query, see the Discovery API reference.

average

Returns the mean of values of the specified field across all matching documents.

Syntax

```
average(field)
```

Example

Product	Price
I Series	200
J Series	450
X Series	325

Table 1. Sample product prices

When the `average` aggregation type is applied to a set of documents in which the `price` field contains the values that are shown in Table 1, the result is `325`.

```
average(price)=325
```

This aggregation type performs a terminal operation. When combined with other aggregations, the output is not used as input for the next aggregation. The output is returned in a discrete group.

```
{
      "query":"brake",
      "aggregation":  "term(field:STATE,count:3).average(field:VEH_SPEED).term(field:CITY,count:2)"
}
```

For each state returned by the first `term` aggregation operation, the response shows the average vehicle speed specified in the incident reports. Notice that the second `term` aggregation uses the output from the first `term` aggregation, not the `average` aggregation, as its input.

```
{
   "matching_results":  9064,
   "retrieval_details":  {
      "document_retrieval_strategy":  "untrained"
   },
   "aggregations":  [
      {
         "type":  "term",
         "field":  "STATE",
         "count":  3,
         "results":  [
            {
               "key":  "CA",
               "matching_results":  1210,
               "aggregations":  [
                  {
                     "type":  "average",
                     "field":  "VEH_SPEED",
                     "value":  26.239653512993264
                  },
                  {
                     "type":  "term",
                     "field":  "CITY",
                     "count":  2,
                     "results":  [
                        {
                           "key":  "LOS ANGELES",
                           "matching_results":  77
                        },
                        {
                           "key":  "SAN DIEGO",
                           "matching_results":  66
                        }
                     ]
                  }
               ]
            }
         ]
      }
```

filter

A modifier that narrows the document set of the aggregation query that it precedes.

Syntax

```
filter(field)
```

Example

The following example filters the matching document set to include only documents that mention `IBM`.

```
filter(enriched_text.entities.text:IBM)
```

When combined with other aggregations, filters the matching documents set to include only those documents that meet the condition you specify.

```
{
      "query":"brake",
      "aggregation":  "filter(VEH_SPEED>50).term(field:STATE,count:3).term(field:CITY,count:2)"
}
```

The query response shows cities where incidents happen that involve the brakes and the vehicle speed is over 50.


```
{
   "matching_results":  9064,
"retrieval_details":  {
   "document_retrieval_strategy":  "untrained"
},
"aggregations":  [
   {
      "type":  "filter",
      "match":  "VEH_SPEED>50",
      "matching_results":  1075,
      "aggregations":  [
         {
            "type":  "term",
            "field":  "STATE",
            "count":  3,
            "results":  [
               {
                  "key":  "CA",
                  "matching_results":  176,
                  "aggregations":  [
                     {
                        "type":  "term",
                        "field":  "CITY",
                        "count":  2,
                        "results":  [
                           {
                              "key":  "FONTANA",
                              "matching_results":  6
                           },
                           {
                              "key":  "ALTA LOMA",
                              "matching_results":  5
                           }
                        ]
                     }
                  ]
               }
            ]
         }
```


group_by

Separates results into groups that you define.

Syntax

```
group_by(condition:[(condition 1),(condition 2)...])
```

Each condition must be specified as a valid Discovery Query Language expression surrounded by parentheses. For example, `(age<20)` or `(flavor:chocolate)`. The maximum number of conditions that you can define is 50.

You can optionally include the `relevancy` parameter and set it to `true` to return the relevancy value of the set of documents that meet the specified condition. When `true`, the results are sorted by relevance. When `false`, the results are sorted by the highest number of `matching_results`.

Example

The following request looks for documents that mention the term `engine`, and groups them by car manufacturing year. The documents are sorted into 3 groups, one group of traffic incident reports involving cars that were manufactured before 2000, one group for cars manufactured in 2000, and one group for cars manufactured after 2000.

```
{
      "query":"engine",
      "aggregation":  "group_by(condition:[(YEARTXT<2000),(YEARTXT=2000),(YEARTXT>2000)],relevancy:true)"
}
```

The results might look like this:

```
{
   "type":  "group_by",
   "results":  [
      {
      "key":  "YEARTXT<2000",
         "matching_results":  2034,
         "relevancy":  1.0,
         "total_matching_documents":  2034,
         "estimated_matching_results":  2034
      },
      {
         "key":  "YEARTXT=2000",
         "matching_results":  1738,
         "relevancy":  1.0,
         "total_matching_documents":  1738,
         "estimated_matching_results":  1738
      },
      {
         "key":  "YEARTXT>2000",
         "matching_results":  32708,
         "relevancy":  1.0,
         "total_matching_documents":  32708,
         "estimated_matching_results":  32708
      }
   ]
}
```

histogram

Creates numeric interval segments to categorize documents.

Syntax

```
histogram({field},{interval})
```

Uses field values from a single numeric field to describe the category. The field that is used to create the histogram must have a number data type, such as `integer`, `float`, `double`, or `date`.

Nonnumber types such as `string` are not supported. For example, `"price": 1.30` is a number value that works, and `"price": "1.30"` is a string, so it doesn't work.

Use the `interval` argument to define the size of the sections for the results to be split into. Interval values must be whole, nonnegative numbers. Choose a value that makes sense for segmenting the typical values from the field.

Histograms can process decimal values that are specified in a field, but the interval must be a whole number.

You can optionally include a custom name by including a `name` parameter.

Example

For example, if your data set includes the price of several items, like: `“price”: 1.30`, `“price”: 1.99`, and `“price”: 2.99`, you might use intervals of `1`, so that you see everything that is grouped in the range `1 - 2`, and `2` and `3`. You do not want to use an interval of `100` because then all of the data ends up in the same segment.

```
histogram(product_price,interval:1)
```

max

Returns the highest value in the specified field across all matching documents.

Syntax

```
max(field)
```

Example

Product	Price
I Series	200
J Series	450
X Series	325

Table 2. Sample product prices

When the `max` aggregation type is applied to a set of documents in which the `price` field contains the values that are shown in Table 2, the result is `450`.

```
max(price)=450
```

This aggregation type performs a terminal operation. When combined with other aggregations, the output is not used as input for the next aggregation. The output is returned in a discrete group.

min

Returns the lowest value in the specified field across all matching documents.

Syntax

```
min(field)
```

Example

Product	Price
I Series	200
J Series	450
X Series	325

Table 3. Sample product prices

When the `min` aggregation type is applied to a set of documents in which the `price` field contains the values that are shown in Table 3, the result is `200`.

```
min(price)=200
```

This aggregation type performs a terminal operation. When combined with other aggregations, the output is not used as input for the next aggregation. The output is returned in a discrete group.

nested

Applying `nested` before an aggregation query restricts the aggregation to the area of the results that are specified.

For example, `nested(enriched_text.entities)` means that only the `enriched_text.entities` components of any result are used to aggregate against.

The following example checks how many mentions are returned per model type.

```
nested(enriched_text.entities).term(enriched_text.entities.model_name)
```

The result shows that there are a total of 50 recognized entities and all of them are of type NLU.

```
"aggregations": [
  {
    "type": "nested",
    "path": "enriched_text.entities",
    "matching_results": 50,
    "aggregations": [
      {
        "type": "term",
        "field": "enriched_text.entities.model_name",
        "results": [
          {
            "key": "natural_language_understanding",
            "matching_results": 50
          }
        ]
      }
    ]
  }
]
```

For another example, see Starting with nested objects.

pair

Analyzes relationships between two fields.

Syntax

```
pair(first:{aggregation},second:{aggregation})
```

The first and second `{aggregation}` values must be one of the following aggregation types:

- `term`
- `group_by`
- `histogram`
- `timeslice`

The `relevancy` parameter from the `term` or `group_by` aggregation is ignored. The `pair` aggregation type calculates relevancy values by using combinations of document sets from the results of the two aggregations.

Only one pair aggregation can be used per query request, and it cannot be combined with any other aggregations.

Example

For example, you might specify `term(model_name)` as the first aggregation and `term(component_name)` as the second. Each of the aggregations returns the following values as keys of aggregated document sets:

- term(model_name): Accord, CR-V
- term(component_name): engine, brake, radiator

The calculated relevancy values of combinations of each of the document sets might look like this:

- Accord x engine
- Accord x brake
- Accord x radiator
- CR-V x engine
- CR-V x brake
- CR-V x radiator

The response defines a two-dimensional array of aggregation results, which can be represented in a table.

Car model	Component: engine	Component: brake	Component: radiator
Accord	Accord x engine	Accord x brake	Accord x radiator
CR-V	CR-V x engine	CR-V x brake	CR-V x radiator

Table 4. Pair aggregation example

Each array of columns and rows of the table is sorted in the same order of the results of the first and second aggregations. For example, if you specify the `term` aggregation as the first argument, the resulting column arrays are sorted by frequency of terms. If you use the `timeslice` aggregation as the

second argument, the row arrays are sorted by date or time.

sum

Adds the values of the specified field across all matching documents.

Syntax

```
sum(field)
```

Example

Product	Price
I Series	200
J Series	450
X Series	325

Table 6. Sample product prices

When the `sum` aggregation type is applied to a set of documents in which the `price` field contains the values that are shown in Table 6, the result is `975`.

```
sum(price)=975
```

This aggregation type performs a terminal operation. When combined with other aggregations, the output is not used as input for the next aggregation. The output is returned in a discrete group.

term

Indicates the frequency of a term or set of terms in a set of queried documents.

Syntax

```
term(field:{field_name})
```

You can optionally specify the following parameters:

- `count`: Specifies the maximum number of terms to return.
- `name`: You can optionally include a custom name. Not returned if relevancy information is included in the request.
- `relevancy`: Boolean value that indicates whether to include relevancy information in the result. You can use relevancy to get a score that indicates the level of relevancy between the term and keywords in the query. This parameter is `false` by default. If set to true, the following fields are returned also:
  - `total_matching_documents`: Number of documents in the collection where the term is mentioned in the specified field.
  - `estimated_matching_results`: Number of documents that are estimated to have the term in the specified field in the set of documents that are returned by the query.

Example

The following example returns the text from the recognized entities in the document, and specifies to return a maximum of 10 terms.

For example:

```
term(enriched_text.entities.text,count:10)
```

When `relevancy` is set to `true`, a relevancy score is shown in the results. Relevancy measures the level of uniqueness of the frequency count compared to other documents that match your query. If the relevancy shows 2.0, it means that the number of times that the two data points intersect is 2 times larger than expected.

For more examples, see Grouping documents and Combining aggregation types.

timeslice

A specialized histogram that uses dates to create interval segments.

Syntax

The syntax is `timeslice({field},{interval},{time_zone})`.

- The field that you specify must have a `date` data type. For more information about date field, see How dates are handled.
- Valid interval values are `1second` or `{n}seconds`, `1minute` or `{n}minutes`, `1hour` or `{n}hours`, `1day` or `{n}days`, `1week` or `{n}weeks`, `1month` or `{n}months`, and `1year` or `{n}years` where {n} is a number.
- You can optionally include a custom name by including a `name` parameter.

Example

The following example shows the number of matches for each day value.

```
timeslice(field:DATEA,interval:1day)
```

The results look as follows.

```
"aggregations":  [
   {
      "type":  "timeslice",
"field":  "DATEA",
"interval":  "1d",
"results":  [
      {
"key":  1262304000000,
"key_as_string":  "2010-01-01T00:00:00.000Z",
"matching_results":  5
      },
      {
"key":  1262390400000,
"key_as_string":  "2010-01-02T00:00:00.000Z",
"matching_results":  18
      },
      {
"key":  1262476800000,
"key_as_string":  "2010-01-03T00:00:00.000Z",
"matching_results":  38
      },
      {
"key":  1262563200000,
"key_as_string":  "2010-01-04T00:00:00.000Z",
"matching_results":  66
      }
```

top_hits

Returns the documents ranked by the score of the query or enrichment. Can be used with any query parameter or aggregation.

Syntax

```
{aggregation}.top_hits({n})
```

Example

The following example returns the top hit for the term `halt` per city.

```
{
   "query":"halt",
   "aggregation":  "term(CITY).top_hits(1)"
}
```

The response contains the top query results for the term `halt` grouped by cities mentioned in documents where the term is most mentioned. Ten results are returned by default. For each of the 10 cities, the document with the top score is returned as the `hit` object. The content for each `hit` in the `hits` array matches the content in each `result` in the `results` array. Only the order of the results is different.

```
"aggregations":  [
   {
      "type":  "term",
      "field":  "CITY",
```

```
      "results": [
        {
          "key": "LOS ALTOS",
          "matching_results": 3,
          "aggregations": [
            {
              "type": "top_hits",
              "size": 1,
              "hits": {
                "matching_results": 3,
                "hits": [
                  {
                    "document_id": "2bed19a9069442fd82542827ebe260d5_7015",
                    ...
                  }
                ]
              }
            }
          ]
        },
        {
          "key": "ANDOVER",
          "matching_results": 2,
          "aggregations": [
            {
              "type": "top_hits",
              "size": 1,
              "hits": {
                "matching_results": 2,
                "hits": [
                  {
                    "document_id": "2bed19a9069442fd82542827ebe260d5_18329",
                    ...
                  }
                ]
              }
            }
          ]
        },
        ...
        {
          "key":"ACTON",
          "maatching_results": 1,
          "aggregations": []
        }
        ...
```


This aggregation type performs a terminal operation. When combined with other aggregations, the output is not used as input for the next aggregation. The output is returned in a discrete group.

trend

Detects sharp and unexpected changes in the frequency of a keyword value in a specified time period based on the past frequency changes of the keyword value.

Sytnax

```
trend(facet:{aggregation},time_segments:{aggregation})
```

The first (`facet`) aggregation must be one of the following types of aggregations:

- `term`
- `group_by`

The `relevancy` parameter from the `term` or `group_by` aggregation is ignored.

The second (`time_segments`) aggregation must be an aggregation of type `timeslice`.

You can alternatively include the following parameters:

- `show_estimated_matching_results:true`: Indicates whether to include the `estimated_matching_results` information in the result. This field contains the number of documents that are estimated to have the term in the specified field or meet the conditions in the specified aggregation for the specified time interval in the set of documents that are returned by the query.
- `show_total_matching_documents:true`: Indicates whether to include the `total_matching_documents` information in the result. This field contains the number of documents in the collection where the term is mentioned in the specified field or the condition is met.

Only one trend aggregation can be used per query request, and it cannot be combined with any other aggregations.

Example

The following example calculates the *trend indicator* or *trend index* by using combinations of results from the following aggregations:

- term(flavor): vanilla, chocolate, mint
- timeslice(date, 1month): Jan 2020, Feb 2020, Mar 2020, Apr 2020, May 2020, Jun 2020

```
trend( facet: aggregation(<parameter>...), time_segments: timeslice(<parameter>...)),
show_estimated_matching_results: <true_or_false>, show_total_matching_documents: <true_or_false> )
```

The resulting matrix can be represented in a table.

Month in 2020	Flavor: vanilla	Flavor: chocolate	Flavor: mint
Jan	vanilla x Jan	chocolate x Jan	mint x Jan
Feb	vanilla x Feb	chocolate x Feb	mint x Feb
Mar	vanilla x Mar	chocolate x Mar	mint x Mar
Apr	vanilla x Apr	chocolate x Apr	mint x Apr
May	vanilla x May	chocolate x May	mint x May
Jun	vanilla x Jun	chocolate x Jun	mint x Jun

Table 5. Trend aggregation example

In the following sample response, the key information is the `trend_indicator` value. The trend indicator measures the increase ratio of the frequency of a given facet value for a given time interval compared to the expected average frequency. The excepted average frequency is calculated based on the changes in the past time interval frequencies of the given facet value, using a weighted arithmetic mean.

If the standarized residual value is less than -2, the observed frequency is less than the expected frequency. If it is greater than 2, the observed frequency is greater than the expected frequency. If the standardized residual is greater or less than the expected frequency by 3 or more, then something unusual is happening and suggests that there might be an anomaly that is worth investigating.

For example, the expected number of feedback submissions for the `vanilla` flavor in May is calculated from the number of feedback submissions that were received previously (from Jan to Apr). The result is `5.341`. The actual number of feedback submissions in May is `10`. The results indicate that the vanilla flavor got about twice the number of feedback submissions as expected. The standardized residual value is `2.016`, which is greater than expected, but not unusually so.


```
{
"aggregations": [
  {
"type": "trend",
"facet": "term(flavor),",
"time_segments": "timeslice(date, 1month)",
"show_estimated_matching_results": true,
"show_total_matching_documents": true,
"results": [
  {
    "aggregations": [
      {
        "type": "term",
        "field": "flavor",
        "results": [
          {
            "key": "vanilla",
            "matching_results": 36,
            "aggregations": [
              {
                "type": "timeslice",
                "field": "date",
                "results": [
                  {
                    "key": 1577836800000,
                    "key_as_string": "2020-01-01T00:00:00.000Z",
                    "matching_results": 4,
```

```
                        "trend_indicator": 0.0,
                        "total_matching_documents": 7,
                        "estimated_matching_results": 0.0
                     },
                     {
                        "key": 1588291200000,
                        "key_as_string": "2020-05-01T00:00:00.000Z",
                        "matching_results": 10,
                        "trend_indicator": 2.016106745,
                        "total_matching_documents": 12,
                        "estimated_matching_results": 5.340760209
                     },
                     {
                        "key": 1590969600000,
                        "key_as_string": "2020-06-01T00:00:00.000Z",
                        "matching_results": 5,
                        "trend_indicator": -0.763212711,
                        "total_matching_documents": 11,
                        "estimated_matching_results": 7.022515985
                     }
                  ]
               }
            ]
         },
         {
            "key": "chocolate",
            "matching_results": 10,
            "aggregations": [...]
         },
         {
            "key": "mint",
            "matching_results": 25,
            "aggregations": [...]
...
}
```


topic

Detects how much the frequency of a keyword value deviates from the expected average for the specified time period. This aggregation type does not use data from previous time periods. It calculates an index by using the averages of frequency counts of other keyword values for the specified time period.

Syntax

```
topic(facet:{aggregation},time_segments:{aggregation})
```

The first (`facet`) aggregation must be one of the following types of aggregations:

- `term`
- `group_by`

The `relevancy` parameter from the `term` or `group_by` aggregation is ignored.

The second (`time_segments`) aggregation must be an aggregation of type `timeslice` .

You can alternatively include the following parameters:

- `show_estimated_matching_results:true`: Indicates whether to include the `estimated_matching_results` information in the result. This field contains the number of documents that are estimated to have the term in the specified field or meet the conditions in the specified aggregation for the specified time interval in the set of documents that are returned by the query.
- `show_total_matching_documents:true`: Indicates whether to include the `total_matching_documents` information in the result. This field contains the number of documents in the collection where the term is mentioned in the specified field or the condition is met.

Only one topic aggregation can be used per query request, and it cannot be combined with any other aggregations.

Example

```
{
       "query: like",
       "aggregation":  "topic( facet: term(flavor), time_segments: timeslice(date, 1month), show_estimated_matching_results: true,
show_total_matching_documents: true )"
}
```

With the same data set and aggregation as is used in the term aggregation example, the results might look as follows.

Notice that the `topic_indicator` values are different from the `trend_indicator` values that are returned by the `trend` aggregation. While both

are calculated from the actual and expected frequencies, they differ because their expected frequencies are computed differently. In the `trend` aggregation, the expected frequency of the feedback submissions for vanilla-flavored ice cream in May is computed from the number of feedback submissions that were received for vanilla previously (from Jan to Apr) and the total number of feedback submissions received for all of the flavors in May. However, in the `topic` aggregation, the expected frequency of feedback submissions for vanilla-flavored ice cream in May is calculated from the number of feedback submissions that were received for vanilla and the total number of feedback submissions received for all of the flavors in May. In this example, the expected frequency result is `12.169`, the actual frequency is `10`, and the `topic_indicator` is `-0.621777032`.


```
{
"aggregations": [
  {
    "type": "topic",
    "facet": "term(flavor)",
    "time_segments": "timeslice(date, 1month)",
    "show_estimated_matching_results": true,
    "show_total_matching_documents": true,
    "results": [
      {
        "aggregations": [
          {
            "type": "term",
            "field": "flavor",
            "results": [
              {
"key": "vanilla",
"matching_results": 36,
"aggregations": [
  {
    "type": "timeslice",
    "field": "date",
    "results": [
      {
        "key": 1577836800000,
        "key_as_string": "2020-01-01T00:00:00.000Z",
        "matching_results": 4,
        "topic_indicator": -0.027972712,
        "total_matching_documents": 7,
        "estimated_matching_results": 4.056338028
      },
      {
        "key": 1588291200000,
        "key_as_string": "2020-05-01T00:00:00.000Z",
        "matching_results": 10,
        "topic_indicator": -0.621777032,
        "total_matching_documents": 12,
        "estimated_matching_results": 12.16901408
      },
      {
        "key": 1590969600000,
        "key_as_string": "2020-06-01T00:00:00.000Z",
        "matching_results": 5,
        "topic_indicator": -0.787665504,
        "total_matching_documents": 11,
        "estimated_matching_results": 7.098591549
      }
    ]
  }
]
},
{
  "key": "chocolate",
  ...
},
{
  "key": "mint",
  ...
}
}
```


unique_count

Returns a count of the unique instances of the specified field in the collection.

Syntax

```
unique_count(field)
```

Example

The following aggregation requests the number of unique enrichment types that are recognized in the query.

```
unique_count(enriched_text.keyword.type)
```

The result indicates that there are 17 matching results. In those 17 documents, 14 entity types are mentioned.

```
{
   "matching_results":  17,
   "retrieval_details":  {
      "document_retrieval_strategy":  "untrained"
   },
   "aggregations":  [
      {
         "type":  "unique_count",
         "field":  "enriched_text.entities.type",
         "value":  14.0
      }
   ],
   "results":  []
}
```

This aggregation type performs a terminal operation. When combined with other aggregations, the output is not used as input for the next aggregation. The output is returned in a discrete group.

In the following example, the aggregation parameter requests for the results to show the first 45 most-frequently mentioned entities. Per entity, it indicates how many documents mention the term and how many times in total that the term occurs.

```
term(enriched_text.entities.text,count:45).unique_count(enriched_text.entities.type)
```

The results include several aggregations such as the following group for the term `PostgreSQL`. The aggregation indicates that the term appears in 4 documents and is mentioned 12 times.

```
{
   "key":  "PostgreSQL",
   "matching_results":  4,
   "aggregations":  [
      {
         "type":  "unique_count",
         "field":  "enriched_text.entities.type",
         "value":  12.0
      }
   ]
}
```

Curations API

The Curations feature is beta functionality.

Use curations to specify the exact document to return in response to a specific natural language query. Curations can guarantee that frequent or important questions always return the most valuable document. The `confidence_score` for a curated query is always `1.00000`.

This beta feature is only available from the API and is applied only to natural language queries, not queries that are specified by using the Discovery Query Language. Beta features are not available from the SDKs.

You can define up to 1,000 curations. For more information, see Create curation in the API reference.

This example shows how a curation is added with the API. When querying with the same or similar `natural_language_query` the document with the `document_id` of `document_id1234` is returned.

```
{
   "natural_language_query":  "curations in watson discovery",
   "curated_results":  [
      {
         "document_id":  "document_id1234",
        "collection_id":  "collection_id1234"
      }
    ]
 }
```

The natural language query that is submitted by the customer must be an exact match for the query that is specified in the curation. Both queries, the one

submitted by the user at run time and the one that is submitted by the curation API and then stored in the index, undergo query analysis. The query analyzer lemmatizes text, removes stop words, and adds query expansions.

You can optionally specify a hard-coded response to the query by including a snippet. A snippet is a response that you author and that is returned when the associated document is returned for the specified natural language query.

```
{
   "curations":  [
      {
         "curation_id":  "c1175536f509405bc68a9f76235fa7bbb6f9af2f",
         "natural_language_query":  "What is a project",
         "curated_results":  [
            {
               "collection_id":  "47477591-b520-6039-0000-017ea213e837",
               "document_id":  "web_crawl_123a2a56-8c26-5acb-9544-c4702ac899a4",
               "snippet":  "A project is a convenient way to collect and manage the resources in your application. You can assign a project type and
connect your data to the project by creating a collection."
            }
         ]
      }
   ]
}
```

If **passages.per_document** is `true`, the text snippet that you specify is returned as the top passage in the `passage_text` field instead of the original passage that is chosen by search. Only one text snippet can be specified per document. If **passages.max_per_document** is greater than `1`, the snippet is returned first, followed by the passages that are chosen by search. Query filters are applied to curation results.

Analyzing data with the Content Mining application

Analyzing your data with the Content Mining application

Use the Discovery Content Mining application to analyze your data. The application shows subsets of your information in visualizations that can help you to find patterns, trends, and anomalies.

Note: Only users of installed deployments (IBM Cloud Pak for Data) or Enterprise and Premium plan managed deployments can use the Content Mining application.

Overview video

View video: IBM Watson Discovery Content Mining

Video transcript

Watson Discovery Content Mining Project presented by Stuart Strolin. (Music intro) The purpose of this video is to familiarize you with the content mining project in Watson Discovery.

Content mining is one of the primary use cases for Watson Discovery and is used for analyzing and exploring both structured and unstructured data to find insights and extract hidden meaning. It is used by both the citizen analyst and the data scientist.

The content mining project can be used for all types of analysis because the user interface is not specific to a particular industry or set of data.

In this scenario, you are an analyst for a fictitious automobile company. Operational reports have alerted the company to an unusual accident rate for one of their cars. Your job is to find out why.

Using the content mining project, you begin your analysis by looking at the unstructured data from the national motor vehicle incident reports. You are presented with an interface that allows you to select the car model and begin your analysis (on the Collections page). In this case, you are interested in the Hill Walker. You could type that information into the search section at the start of the page. But it's easier just to click on the item. You can add as many search terms and conditions as you like. But in reality, you want to let the application guide your analysis.

What you see now is the navigation view (in Guided mode). It keeps track of your analysis and provides options for next steps. It also provides a count of the number of documents that match your current state of analysis. In this small collection, the number of documents relating to the Hill Walker is only 51. In a production data set, the number would usually be much larger. Analyzing trends and anomalies is often a good way to start as it allows you to see if anything seems out of the ordinary.

Immediately, you notice that the Hill Walker has problems in December and January. You decide to investigate further by narrowing this initial exploration to just the month of December.

Notice how the navigation view at the top always keep you informed of where you are in your analysis. Next, you select *Analyze cause and characteristics* because you are interested in why things are happening.

You notice that words like 'snow' and 'brake' are highlighted together (in the Part of Speech section), so you add these to your analysis.

The Content Miner project has narrowed your investigation to a small number of complaints that can be easily read. (clicks Show Documents)

The common theme here is that there is an unexpected problem with the way the brakes are working in snowy conditions. You now have the information you need to ask the engineering department to perform a detailed inspection of the braking system and determine why it is not working as expected in snowy conditions.

In this demonstration, you saw how a citizen analyst using Watson Discovery and content mining can easily discover hidden meaning in unstructured text. (list of features, functionality, and use cases)

What will you do with Watson Discovery? (Music outro)

How it works

To analyze your data, you use *facets*. Facets give you a way to slice your data and visualize a subset of information so it is easier to comprehend.

From the data analysis page for your collection, you can choose for the data to be shown in one of the following views:

Facets

: Shows facets that are derived from from annotations that are added to your documents by enrichments that are applied to your documents. Enrichments can include built-in Natural Language Processing enrichments, such as *Part of Speech* or *Entities*. They can also include custom enrichments that you add, such as dictionaries, regular expression patterns, and machine learning models.

Metadata facets

: Shows facets that are derived from your data. When you add files to a collection, Discovery analyzes and indexes the data. Annotations are added to identify content types and are shown as metadata facets. The best metadata facets result when you ingest structured data, such as records from a CSV file.

Custom

: Shows only the facets that you choose to add to the view. You can add a mix of enrichment-derived and content-derived facets to your custom view.

When you create a *Content Mining* project type, the *Part of Speech* facet is applied to your data automatically. This facet is a great place to start because it is valid for all data, no matter the subject. The output gives you a quick look at the terminology that is most common in the data.

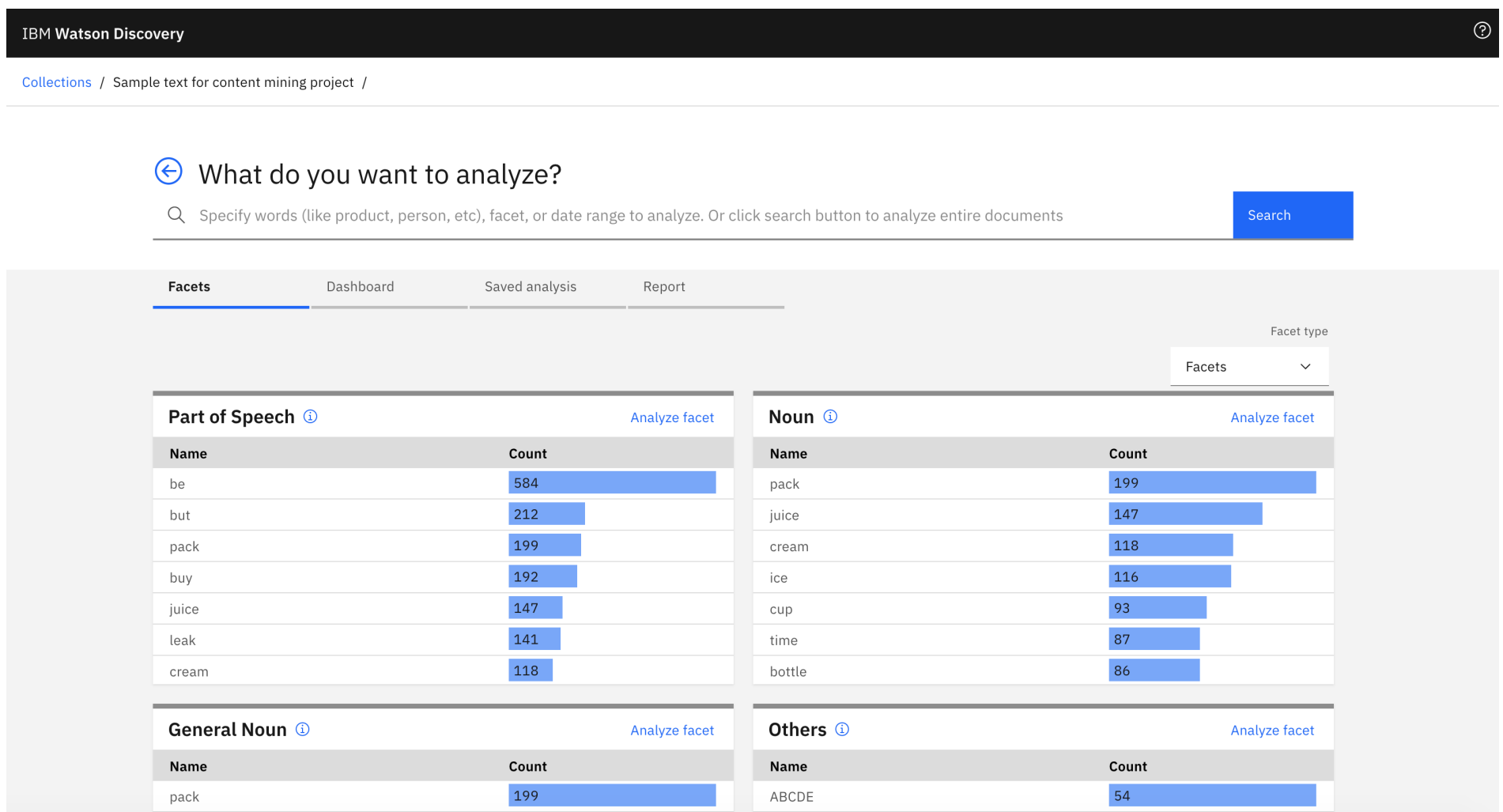


Figure 1. Watson Discovery Content Mining application home page

From this starting point, you can determine other ways to filter the data that might be useful.

If your data consists of traffic reports, for example, the *Part of Speech* facet might show that high frequency keywords include terms such as *engine*, *brake*, *fire*, *smoke*, and *spark*. Given this common terminology, you can create dictionaries to help you categorize and filter the data. The keywords from the example might lead you to create the following dictionaries:

- `component` dictionary for terms such as engine and brake
- `phenomenon` dictionary for terms such as fire, smoke, and spark

When you apply the dictionary enrichment to your data, it generates *annotations*. You can think of annotations as tags that you add to words or phrases, where the tag categorizes or identifies the meaning of the word or phrase. The resulting annotations function as new facets that you can use to filter and dissect your data further.

With your new `component` and `phenomenon` facets, for example, you can look for correlations between components and phenomena that are involved in traffic incidents.

Learn about the ways that you can analyze your data.

Digging deeper

To dig even deeper into your data, apply or create AI models that can find different types of information in your documents. You can apply built-in natural language processing models, such as the *Entities* enrichment that can recognize mentions of commonly known things, such as business or location names and other types of proper nouns. Or you can apply a custom model that recognizes terms and categories that are unique to your data.

Extend your analysis by adding your own facets.

Getting started

Before you can use the application, you must create a Discovery Content Mining project. After the project is created and data is uploaded, you can open the Content Mining application.

For more information, see Creating projects.

Of course, you can't get out useful insights if you don't put the right type of information in. Be sure to include consistent data. If you want to find trends over time, your data must include data points that specify a date.

Data that is submitted in CSV file format is optimal. For a sample of a CSV file that provides interesting analysis capabilities, see Analyzing CSV files.

Data analysis methods

Use tools from the Content Mining application to analyze your data.

You can analyze your data in the following ways:

- Look for relevant keywords
- Find trends
- Identify anomalies in cyclical patterns
- Find characteristic words
- Analyze relationships
- Analyze relationships between many facets

As you review the results of your analysis, you can flag documents that you want to research further later. For more information, see Flagging documents.

When you find important insights, you can take a snapshot of the view, and then add it to a report to share with others. For more information, see Creating a report.

Start your analysis

Use the content mining application to analyze documents in your collection based on the document text and any annotations or enrichments that are stored in the documents.

To start your analysis, complete the following steps:

1. Enter a search term, click a facet with which to filter the documents, or leave the search field blank to return all of your documents.
2. Click **Search**.

The guided mode view of the results shows suggested next steps that you can take to analyze your data further. If you don't want to see suggestions, you can switch to **Expert mode**. In Expert mode, the *Documents* view that lists the search results is returned whenever you submit a search.

The tasks in this topic describe how to use the application in guided mode.

Look for relevant keywords

To analyze keyword relevance, complete the following steps:

1. From the initial search page, submit a keyword search to filter the documents.
2. From the search results page in guided mode, click **Analyze cause or characteristics**.

   After the characteristic words pane, a pane with relevancy information for each facet type is displayed.

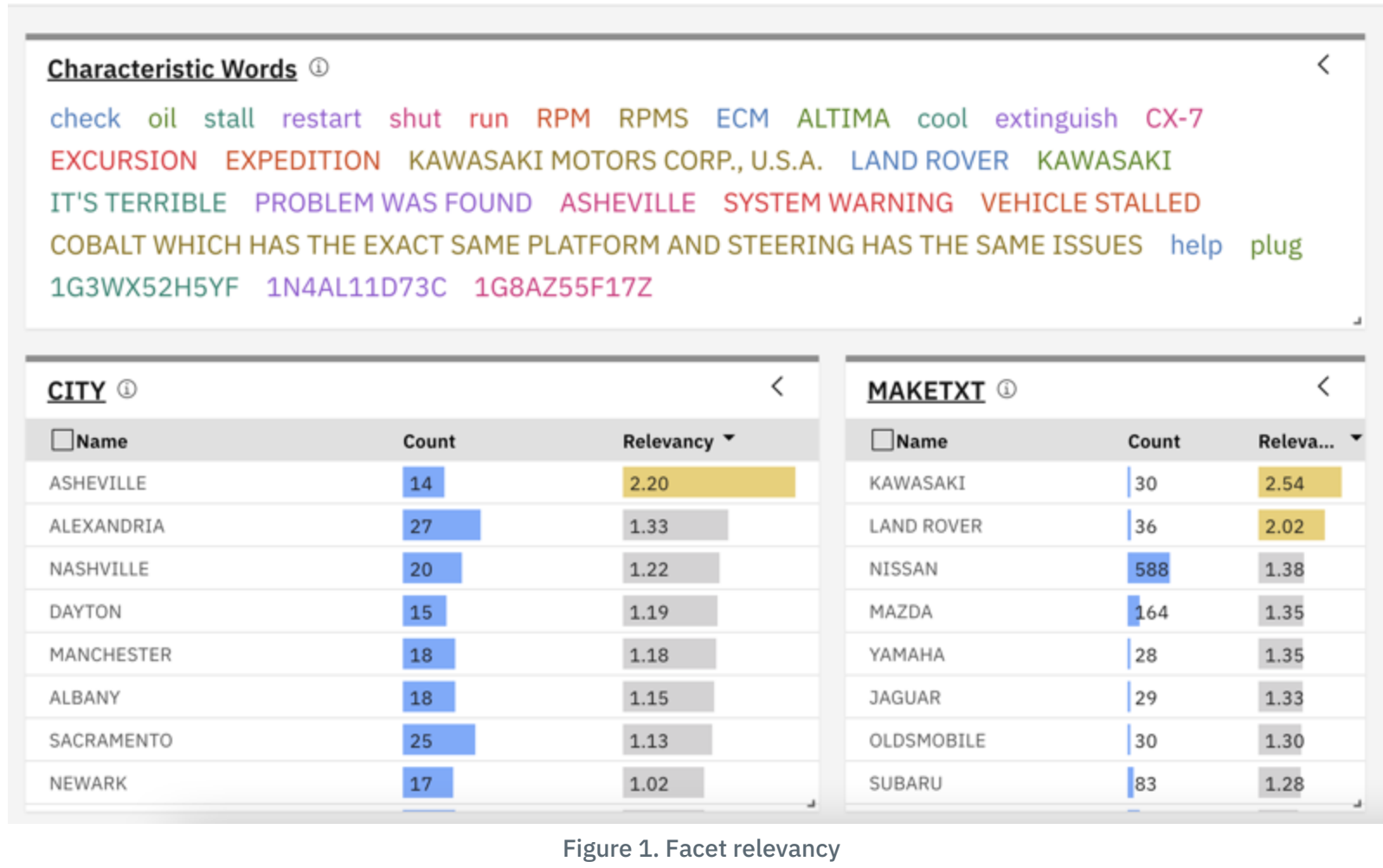


Figure 1. Facet relevancy

Each relevancy pane shows a list of the keywords that occur in the documents that match the facet type.

The *Count* column shows the number of documents in the current result set that contain the keyword. The *Relevancy* column shows the level of uniqueness of the frequency count compared to other documents that match your query. High relevancy values are shown in shades of color with increasing intensity. The color begins at yellow, then increases to orange, and then to red.

Find trends

Use *Trends* analysis to find trends in your data. For example, you might see that a new product release aligns with an uptick in customer interest. Or that a new customer care approach is followed by an increase in customer satisfaction.

> ⚠ **Important:** Your documents must contain at least one date field for trend information to be available.

To find trends, complete the following steps:

1. From the initial search page, enter a keyword or select a facet with number values to filter the documents.
2. Click **Find trends and anomaly** from the list of suggested next steps that is displayed in the guided mode view.

   The resulting bar graph shows the number of documents that mention the term or facet value that you specified in the search query over time.

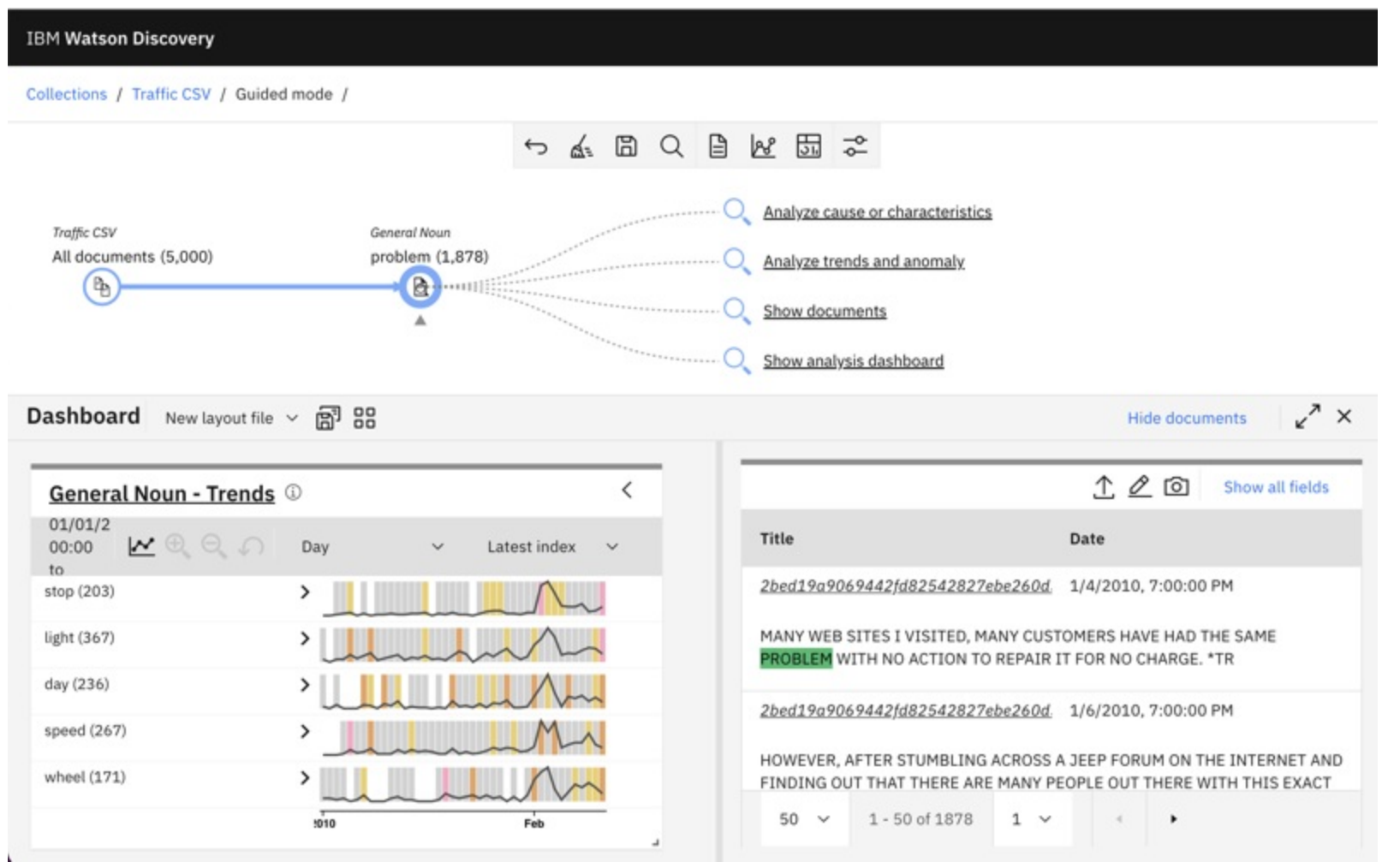


Figure 2. Facet trend graph

The time series chart is rendered as a heat map. Each cell color indicates a level of relevancy.

3. You can click a facet to investigate it more closely. The facet is shown in a bar graph.

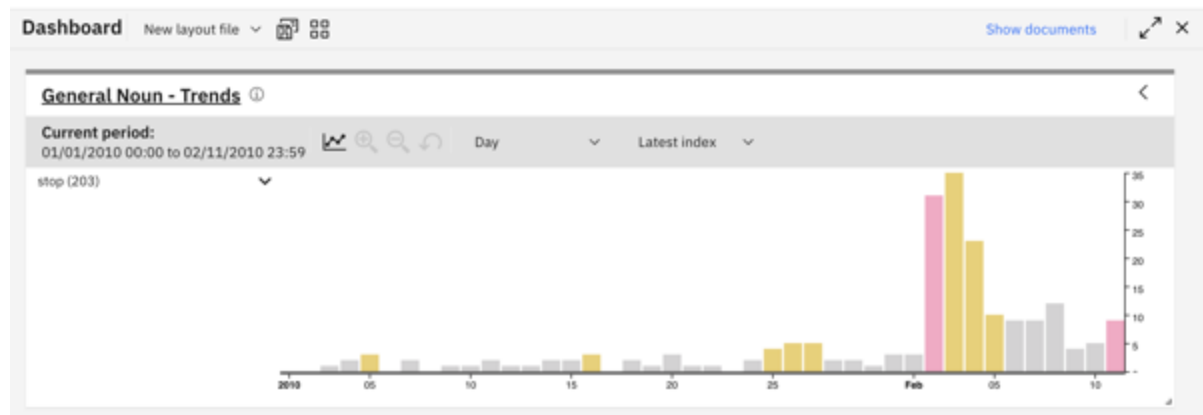


Figure 3. Facet trend detail in bar graph

Each individual bar graph highlights trends in your data that deviate from the normal distribution by displaying *increase indicators*.

Increase indicators measure how much the frequency of a facet value on a specific date or in a particular time interval deviates from the expected average frequency. The average is calculated based on the changes in the past time interval frequencies.

You can click individual items in a visualization or click and drag the cursor to select contiguous items.

The cyclical data is calculated from the current time zone setting of your collection. If you want to change the time zone that is used by the graph, see Change the time zone.

Identify anomalies in cyclical patterns

Use *Topic* analysis to find anomalies in seasonal, monthly, or even daily patterns that are present in your data.

⚠ **Important:** Your documents must contain at least one date or time field for topic information to be available.

Topic analysis focuses on how much the frequency of a keyword deviates from the expected average frequency in a specific time period. The expected average uses all of the averages of the frequency counts for other keywords in the same time period. This method of analysis is useful for identifying patterns that occur cyclically and highlights any unexpected changes that might occur in these cyclical patterns.

To find anomalies, complete the following steps:

1. From the initial search page, enter a keyword or select a facet with number values to filter the documents.
2. From the search results page in guided mode, click **Analyze cause or characteristics**.
3. From the *Facet analysis* pane, select **Topic**.
4. Adjust the following values to suit your analysis:
    - Number of results
    - Date facet
    - Time scale
    - Date range
5. Choose a target facet or subfacet, and then click **Analyze**.

   The resulting time series graph shows changes in the frequency of keyword mentions over time.

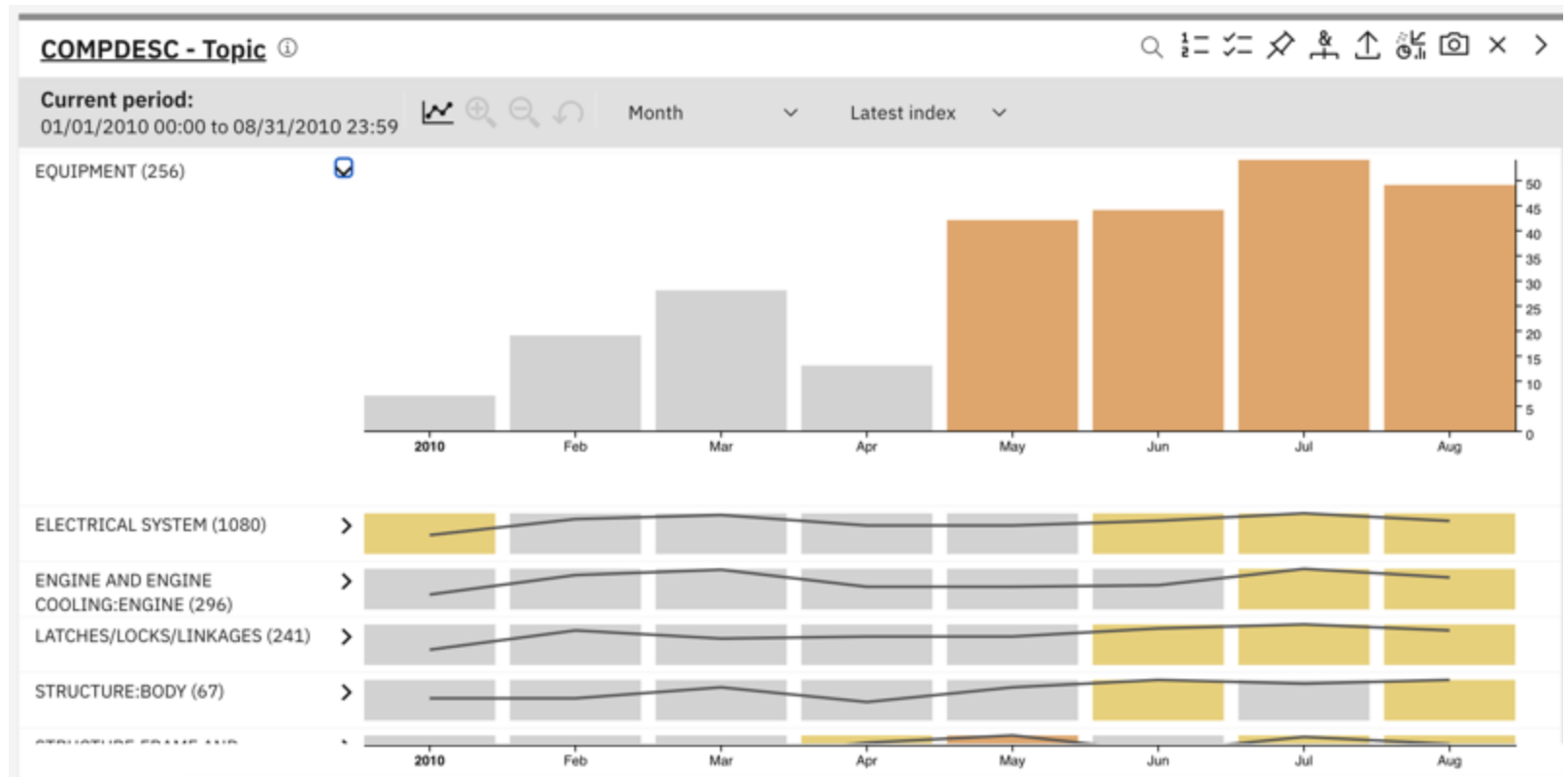


Figure 4. Topic analysis time series view

Color coding is used to highlight when the number of mentions deviates from the expected frequency. The higher the deviation, the more intense the color, from yellow to orange to red. The average is calculated based on the frequency of occurrence of other keywords in the same time period.

The cyclical data is calculated from the current time zone setting of your collection. If you want to change the time zone that is used by the graph, see Change the time zone.

Find significant terms

Find characteristic words from your data set. The characteristic words view is a word cloud that shows terms that are mentioned frequently in the documents you are analyzing.

You can click a word from the word cloud to add it to the existing query and filter the current document set to include only documents that also mention the specified word.

To find significant terms, complete the following steps:

1. From the search results page in guided mode, click **Analyze cause or characteristics**.

   The characteristic words view is displayed.

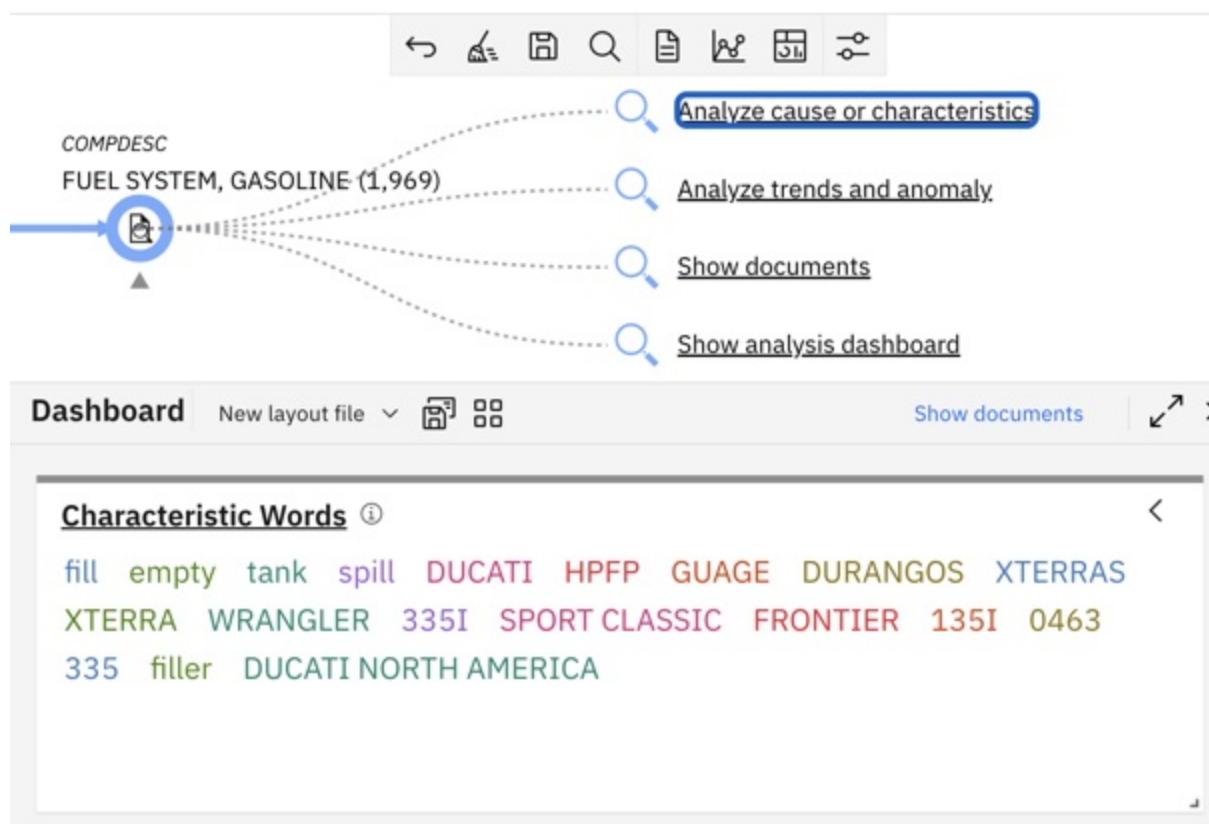


Figure 5. Characteristic word cloud

Note: The different font colors help to distinguish the words from one another; they have no statistical meaning.

2. Click a word in the cloud to limit the document set to include only documents that mention the word.

Analyze relationships between two facets

Use *Pairs* analysis to see how two facets are related to one another.

To compare two facets, complete the following steps:

1. From the *Facet analysis* pane, select **Pairs**.
2. Find the first facet that you want to compare in the list. Click either the X- or Y-axis icon that is associated with the facet to indicate where you want the facet values to be displayed in a two-dimensional graph.
3. Find the second facet, and then click the remaining axis icon. For example, if you selected the X-axis icon previously, select the Y-axis icon for the second facet.

   Data from the two facets is displayed in a graph.

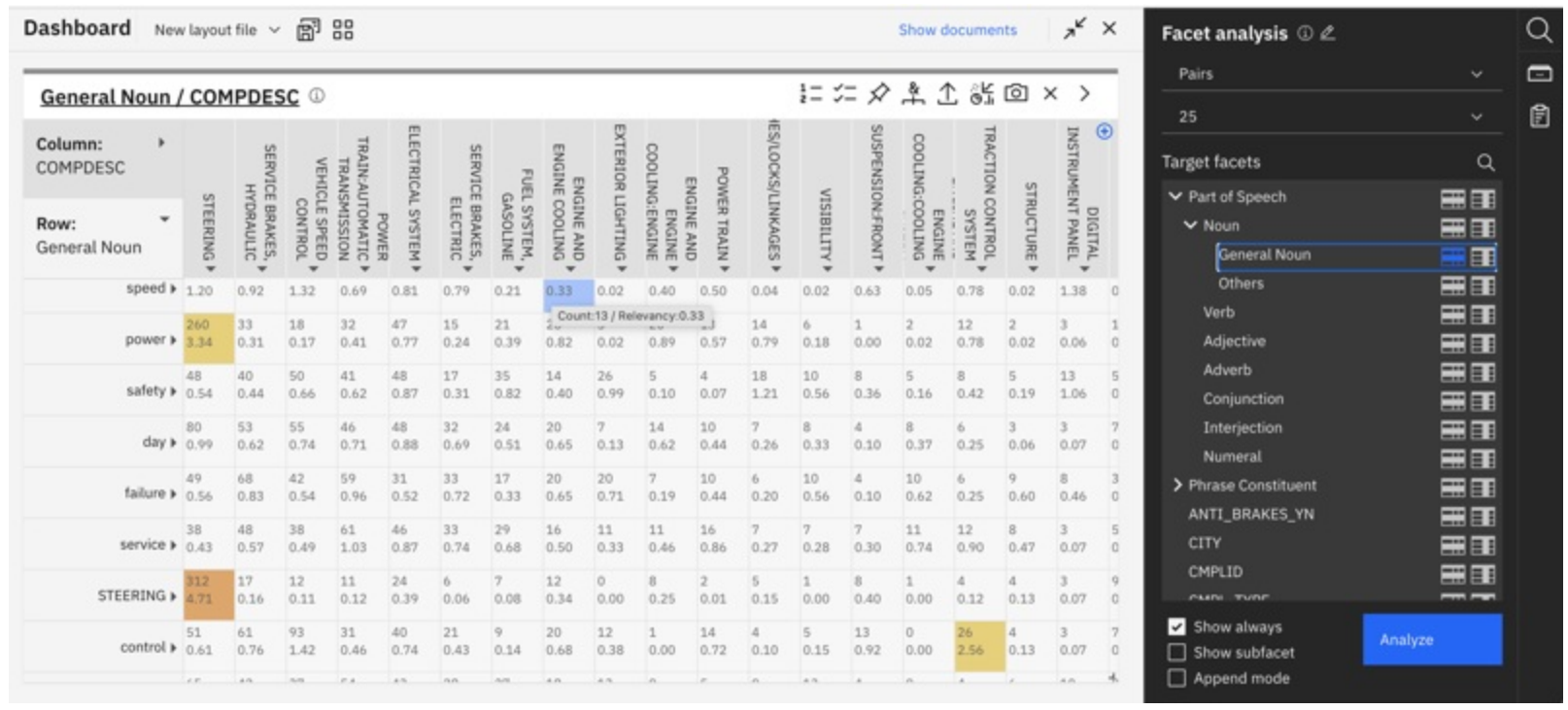


Figure 6. Facet comparison graph

The graph shows two numbers. The first number is a frequency count and the second number is a relevancy value. The frequency count measures how many times the two data points are found together in a document. Relevancy measures the level of uniqueness of the frequency count compared to other documents that match your query. If the relevancy shows 2.0, it means that the number of times that the two data points intersect is 2 times larger than expected. To help you identify anomalies that might require more in-depth analysis, high relevancy values are shown in shades of color with increasing intensity, from yellow to orange to red.

Analyze relationships between many facets

Use *Connections* analysis to see how multiple facets are related to each other.

To compare two or more facets, complete the following steps:

1. From the *Facet analysis* pane, select **Connections**.
2. Select the root facet that you want to compare to other facets first.
3. Select up to 4 more facets from the list, and then click **Analyze**.

   Pair analysis is done between the first facet and each other facet in turn.

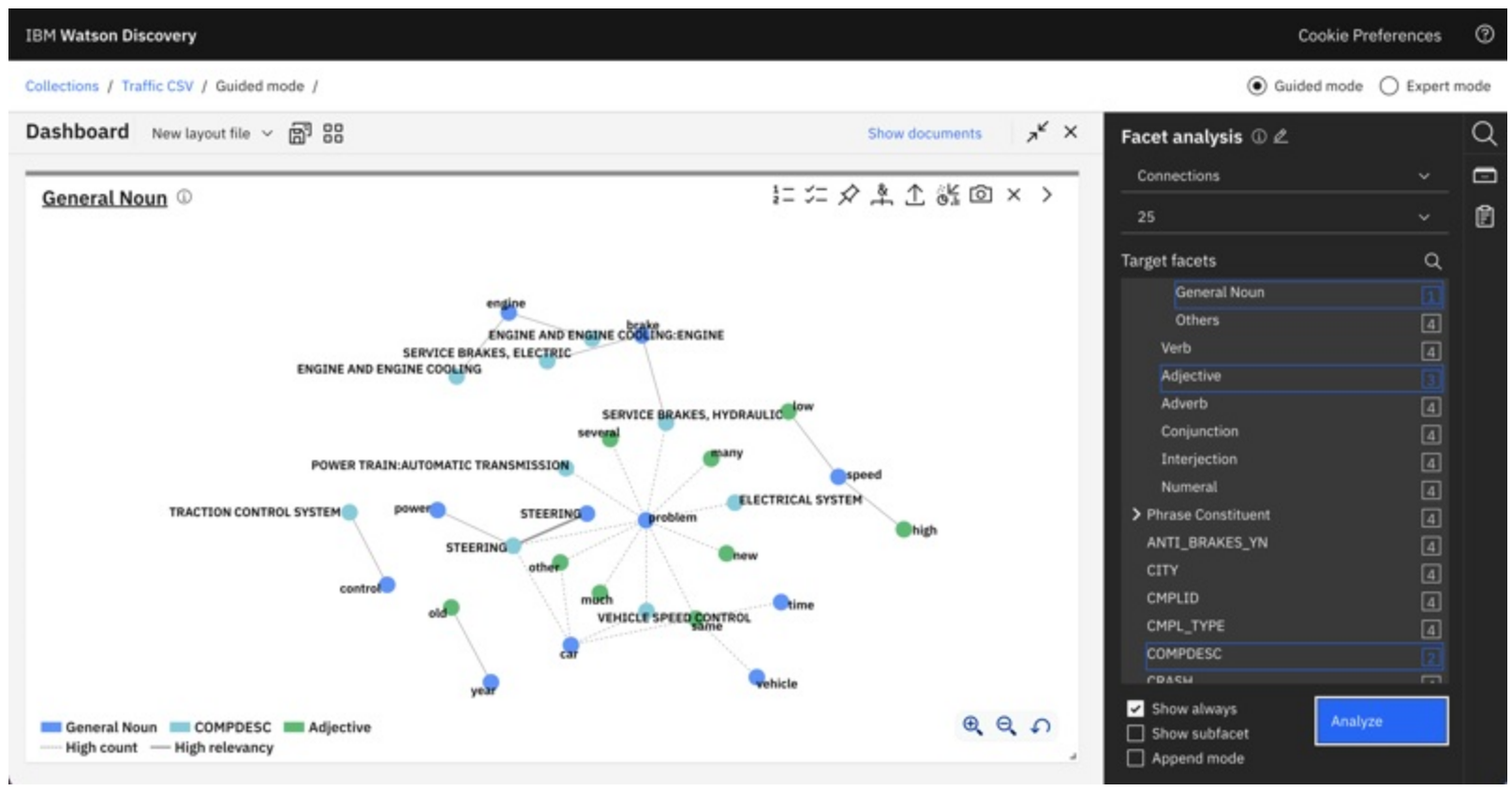


Figure 7. Facet network graph

The resulting network graph shows only highly relevant and high-frequency pairs. Each node represents a facet value. The node color reflects the facet type. A solid-line connection between nodes identifies highly relevant pairs. A dotted-line connection identifies high-frequency pairs.

Changing number ranges

If the scale of a graph is not optimized for your data, you can change it. For example, to plot vehicle speeds, you might want a range that increments by tens or twenties rather than by thousands.

To change the scale of a graph for a facet, complete the following steps:

1. Click **Collections** link in the page header.
2. In the tile for your collection, click the *Open and close list of options* icon, and then choose **Edit collection**.
3. In the *Facet* tab, find the facet for which you want to change the number range.
4. In the Range field, click **Edit**.
5. Define each range that you want to use as a JSON object. You can add or remove objects to change the number of data points in the range.

   For example, the JSON objects that identify the ranges for vehicle speeds might look as follows:

```
[
   {
      "query":  "[1, 20)",
      "label":  "1 - 19"
   },
   {
      "query":  "[20, 40)",
      "label":  "20 - 39"
   },
   {
      "query":  "[40, 60)",
      "label":  "40 - 59"
   },
   {
      "query":  "[60, 80)",
      "label":  "60 - 79"
   },
   {
      "query":  "[80, 100000)",
      "label":  "80+"
   }
]
```

6. Click **Apply**.
7. Click **Save**, and then click **Close**.
8. Click your collection tile to return to the collection and continue your analysis.

The changes to the number ranges for vehicle speeds introduce more opportunities for relationships or anomalies in the data to be highlighted.

STATE / VEH_SPEED

Column: VEH_SPEED / Row: STATE	1 - 19	20 - 39	40 - 59	60 - 79	80+
AZ	0.70	0.36	0.74	0.54	0.00
IN	25 0.41	34 0.63	34 1.00	11 0.40	0 0.00
CO	31 0.55	35 0.66	16 0.39	17 0.72	0 0.00
WI	33 0.59	38 0.73	18 0.46	16 0.67	1 0.02
CT	35 0.65	49 1.02	9 0.18	11 0.41	0 0.00
MN	55 1.15	38 0.76	14 0.34	10 0.37	0 0.00
TN	27 0.52	29 0.59	21 0.63	14 0.63	5 2.04
SC	20 0.49	19 0.47	16 0.60	10 0.54	0 0.00
AL	7 0.10	15 0.36	20 0.85	10 0.57	0 0.00

Figure 8. Results after changed number range

Showing results in a map visualization

Facets that represent geographical locations can be shown in a map visualization. For example, if you have a collection with a US states facet, you might want to display data per state from a visualization that enables users to select each state from a map.

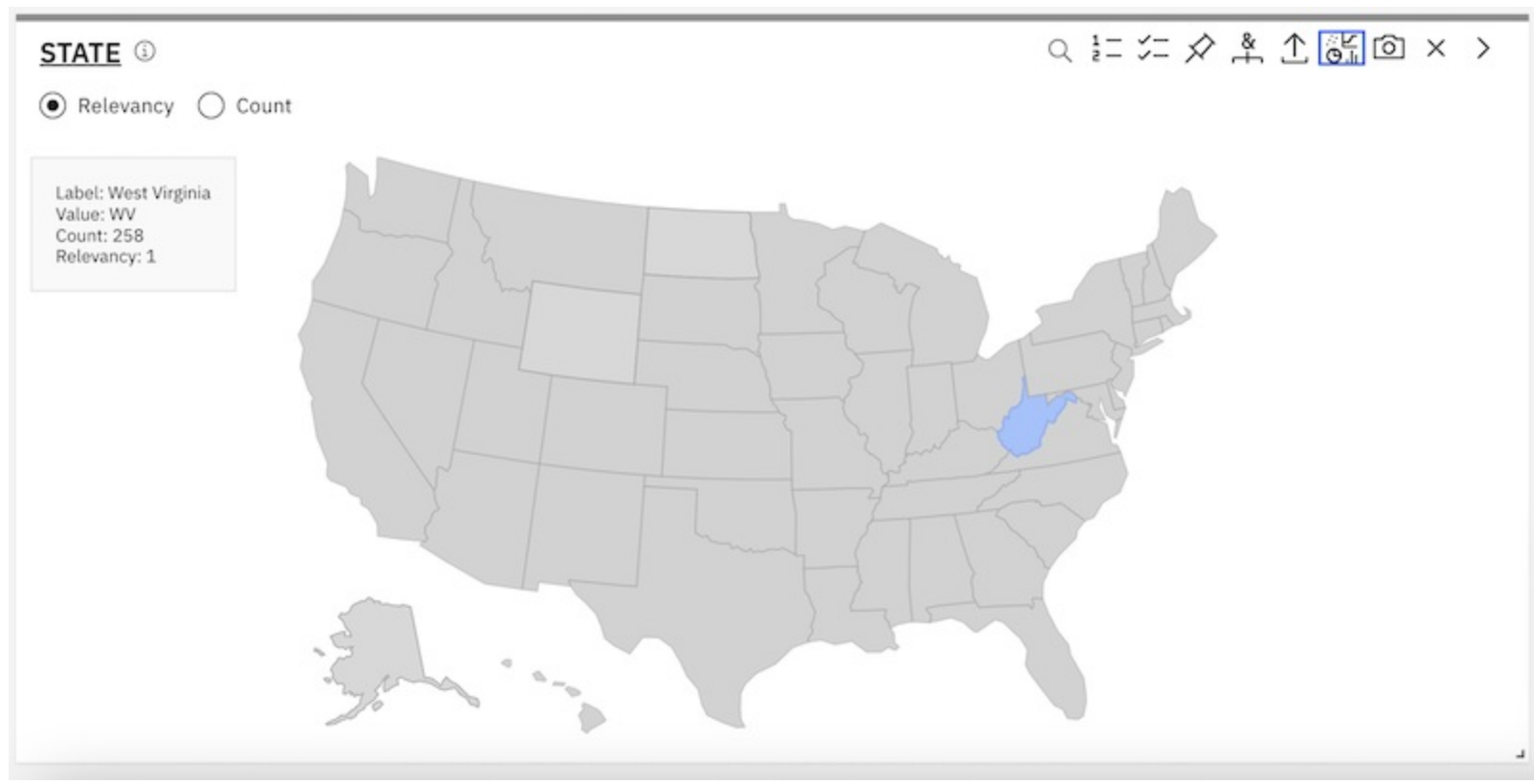


Figure 9. Results shown in a map visualization

A US Map is available by default. You can add a custom map that is built in GeoJSON format. For more information, see RFC7946.

To use a map that you define, complete the following steps to import the map definition:

1. From the Content Mining application home page, click **Collections** from the breadcrumb in the page header.
2. Click the *Settings* icon at the start of the page.
3. Click **Manage customization resources**.
4. Click **Add resource**.
5. Name the resource, and then click **Next**.
6. Add your map file, and then click **Save**.

To make the map that you added available as a visualization option for a facet, you must edit the facet.

1. Click **Home** from the breadcrumb in the page header.
2. Right-click the overflow menu for your collection, and then choose **Edit Collection**
3. Open the *Facet* tab, and then find the facet with which you want to associated the map visualization.
4. Change the *Visualization type* value to **Map**, and then pick the map that you added from the list in the **Resource** field.
5. Click **Save**, and then click **Close**.

Flag documents of interest

Use document flags to assign a custom flag to a document or a group of documents for classification, export, or further analysis.

Flagging documents is a useful way to highlight documents that you want to examine further later.

Before you can flag documents, you must create flags for your collection. For more information, see Add document flags.

To apply flags, complete the following steps:

1. From the analysis view of your collection, create a query that returns a set of documents with specific characteristics.
2. From the documents view, click the *Document flags* icon.
3. Select a flag.
4. You can choose to apply the flag to all query results or to selected documents, and then click **Apply**.

> **Note:** You can't set a document flag more than 50 times per collection. Whether you flag one document that you select individually or flag a query, which might return many documents, each action counts as setting a flag one time.

A flagged document set dynamically changes as the collection is updated. Flagged document sets are stored as queries in the index. Each flag has a query that represents the document set that it is associated with. For example, after you create the document flag and you search for the term `ice cream` and apply a red flag to all of the documents that have this word, `ice cream` is stored as the query that represents the flag. Then, if you search for the term `coffee` and apply the red flag to all of the documents that have that word, the internal flag query changes to `(ice cream) OR coffee`. Therefore, if new documents that contain the word `coffee` are ingested, the red flag is applied to those documents automatically.

Viewing flagged documents

To view the documents to which a flag is applied, complete the following steps:

1. In the *Facet analysis* panel, scroll down to the *Document flags* facet.
2. Select the facet, and then click **Analyze** to open the *Document flags* dashboard.
3. Click one of the flags, click **Analyze more**, and then click **Show documents**.

Removing document flags from a Document Flags query

To remove a document flag, complete the following steps:

1. From the *What do you want to analyze?* page, submit an empty query by clicking **Search**.

   The empty query returns all of the documents in your collection.

2. Click **Show documents**.
3. Click the **Document flags** icon on the toolbar, clear the checkbox of the document flag, and then click **Apply**.

   The document flags are removed from your documents.

Adding facets

Add more facets that you can use to filter your data.

When you apply custom enrichments to your collection, annotations are added to its documents. The annotations feed into new facets that you can use to sort your data.

The following table describes the types of facets that you can create from annotations.

Information to recognize	Annotator type
Commonly understood terms, such as organization or people names.	Built-in Natural Language Processing models
Phrases that express an opinion and evaluate whether the opinion is positive or negative.	Phrase sentiment

Alternative words that share a meaning with terms in a finite list.	Dictionary
Terms that match a syntactical pattern	Regular expression
Custom terms by the context in which they are used.	Machine learning model
Documents that fit into categories that you define.	Document classifier

Custom facet types

Grouping facets

To organize your facets, you can group them in folders.

Grouping facets does not combine the data from the facets. It merely makes the facets easier to find because they are organized in named folders.

To associate facets such that you can combine data from multiple facets, the facets must have a facet and subfacet relationship. Such hierarchical relationships must be defined at the time that the facet enrichment or annotation is created and applied to the collection.

To group facets, complete the following steps:

1. From the initial search page, submit a search.
2. From the *Facet analysis* pane, click the *Edit* icon.
3. Name the group, and then select the facets that you want to group together.

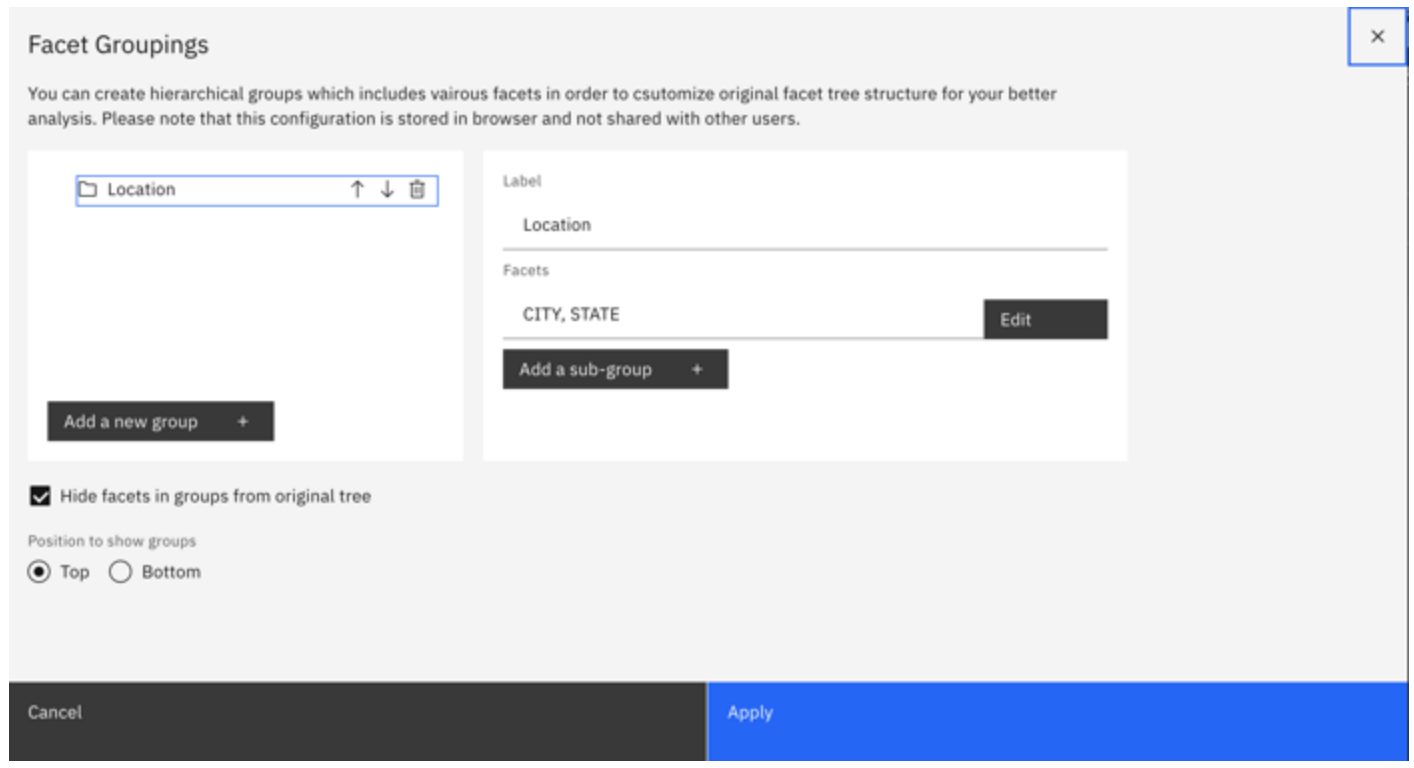


Figure 1. Facet grouping dialog

4. Click **Apply**.
5. The facets that you grouped are now available from a folder with the group name.

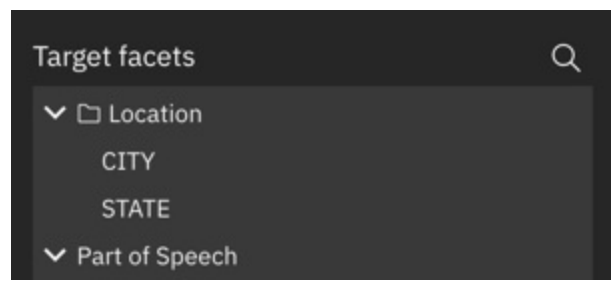


Figure 2. Facet folder from the facet list

Creating custom annotators

You can create a dictionary, regular expression, or machine learning annotator to generate new facets that can help you to analyze your data.

Before you begin, have the following data ready.

Annotator type	Description	Data

Dictionary	Assigns facets to terms that match dictionary entries that you define or upload.	You can optionally upload a file of dictionary terms.
Machine learning	Assigns facets to mentions that are recognized by a machine learning model that you upload.	A compressed file of a machine learning model is required.
Regular expression	Assigns facets to text that matches Java regular expression patterns that you define or upload.	You can optionally upload a JSON file that contains regular expression patterns.

Custom annotator prerequisite data

To create a custom annotator, complete the following steps:

1. From the analysis view of your collection, click the **Collections** link in the breadcrumb to open the *Create a collection for your analytics solutions* page of the Content Mining application.
2. To create an annotator, click **collection**, and then select **custom annotator** from the list.

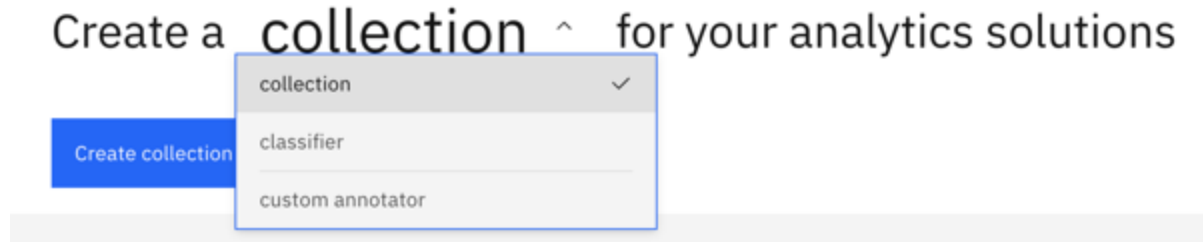


Figure 1. Collection menu

3. Click **Create custom annotator**.
4. Name your annotator, and then optionally add a description.
5. Choose the annotator type, and then click **Next**.
6. Follow the on-screen instructions.

   For more information about how to configure each annotator type, see one of the following sections:

   - Dictionary
   - Machine learning model
   - Regular expressions

Dictionary configuration

You can import an existing dictionary by uploading it or you can create a dictionary by adding terms one at a time.

If you plan to import a dictionary, the dictionary terms must be defined in a CSV file. Specify each term and its synonyms in a separate line. Use the following syntax to specify each term:

```
{term},{synonym},{synonym},...
```

To add a dictionary, complete the following steps:

1. Do one of the following things:
   - To import the dictionary terms:
     1. Click **Import**, and then browse for the file with your dictionary terms.
     2. Click **Import**.
   - To define the dictionary terms:
     1. Click **Add**.
     2. Click *Word list* to add the dictionary terms.
     3. Click **Add**, and then add the term in the **Base word** field and any synonyms that you want to define for the term in the **Other words** field. Separate multiple synonyms with commas. Click **OK**.
     4. Repeat the previous step to add more dictionary terms.
     5. After you finish adding dictionary terms, click *Basic settings*.
2. Name the dictionary.
3. If you plan to define terms with a part of speech other than a noun, specify the part of speech.
4. Decide how you want to handle case.

When case is ignored, the terms `Sat`, `SAT`, and `sat` are all labeled as occurrences of the `Sat` dictionary term.

When you deselect the **Ignore case** checkbox to create a case-sensitive dictionary, the surface form of the term with uppercase match is used. Annotations are added for the term exactly as written and for variations of the term in which the letters are uppercase.

For example, a `sat` entry in the dictionary results in annotations for `sat`, `Sat`, or `SAT` mentions when they occur in text. For a `Sat` entry in the dictionary, annotations are added for occurrences of `Sat` and `SAT`, but not for `sat`.

5. Identify the facet name to use for this dictionary.

   The facet name that you specify for the annotator is the facet name that is displayed from the collection search view.

   You can create a hierarchical facet by including a period (.) in the facet name. For example, you might create one dictionary with the facet path `Food.Vegetables` and others with the facet paths `Food.Fruits` and `Food.Proteins`. Add more facet groups with more periods. For example, you can add `Food.Proteins.Nuts` and `Food.Proteins.Meats` to categorize proteins even further.

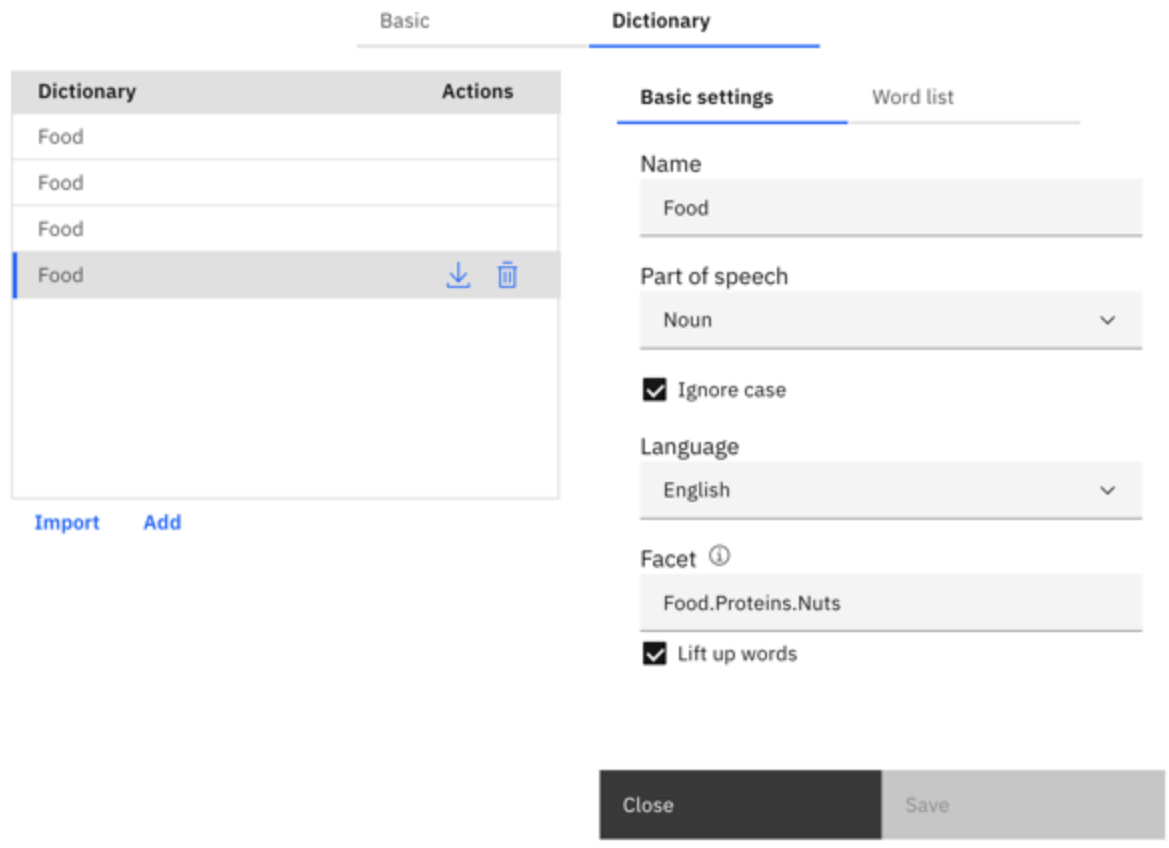


Figure 2. Adding a dictionary

6. If you want documents that are returned for a subfacet to be included when a user filters on the root facet, select **Lift up words**.

   For example, you might enable *Lift up words* for `Food.Fruits` and `Food.Proteins` but not `Food.Vegetables`. As a result, when a user clicks the Food facet, the returned documents include documents that mention terms included in the Fruits and Meats dictionaries, such as *apples* and *beef*.

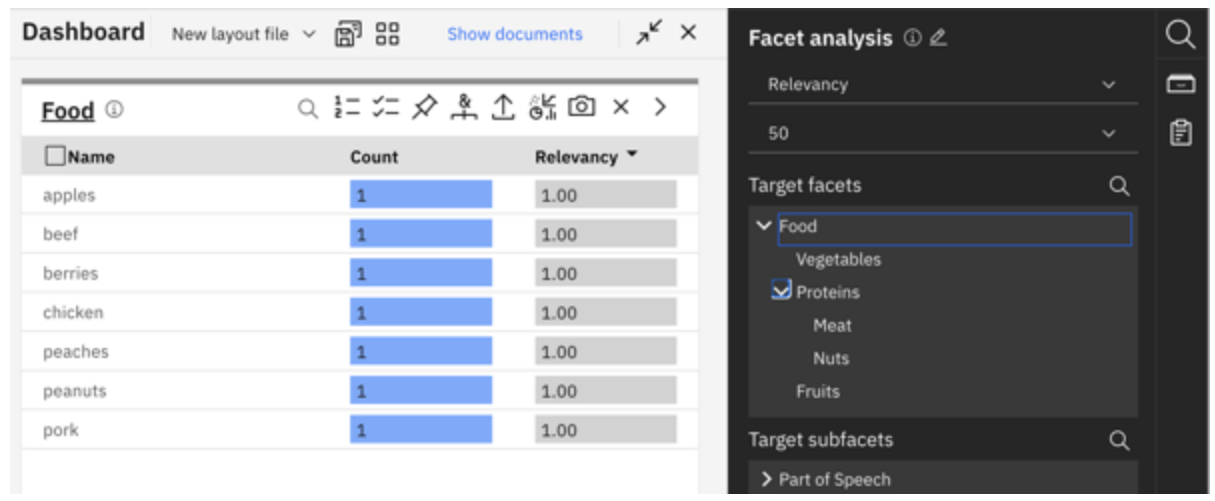


Figure 3. Dictionary enrichment application

However, a user must click the *Food>Vegetables* facet explicitly to get documents that mention terms in the Vegetables dictionary, such as *lettuce*, to be returned.

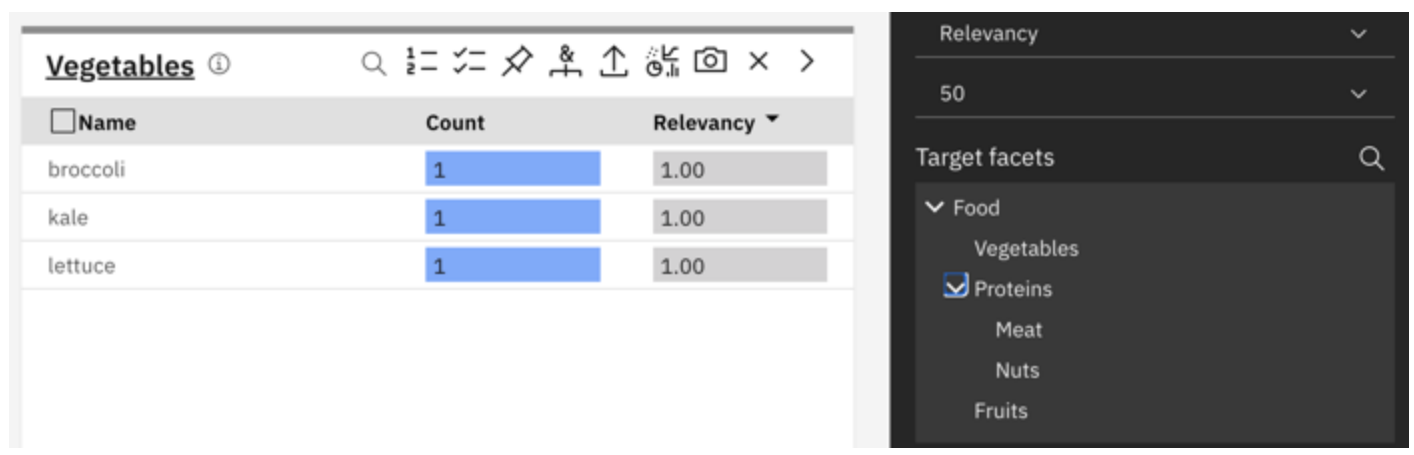


Figure 4. Subfacets

7. Repeat previous steps to add more dictionaries.
8. Click **Save**.

From the custom annotator page, you can see dictionaries that were created in other projects, including non-Content Mining projects. Dictionaries from other project types show the enrichment name as the annotator name. The *Ignore case* and *Lift up words* settings are disabled and the dictionary is named `custom dict`.

Dictionary limits

Plan	Number of dictionaries per service instance	Number of base words per dictionary	Number of terms for which suggestions can be generated
Cloud Pak for Data	Unlimited	Unlimited	1,000
Premium	100	10,000	1,000
Enterprise	100	10,000	1,000

Dictionary plan limits

Totals include enrichments that you create in this Content Mining project and in other projects in the same service instance.

Machine learning configuration

You can import an existing machine learning model.

To use Discovery to create a model, see Entity extractor.

To import a model, complete the following steps:

1. Click **Select file**, and then browse for the machine learning model file.
2. In the **Facet path** field, specify the root facet name to use for the model.

   The facet name that you specify for the annotator is the facet name that is displayed from the collection search view.
3. Click **Save**.

Machine learning model limits

Plan	ML models per service instance
Cloud Pak for Data	Unlimited
Premium	10
Enterprise	10

ML model plan limits

Totals include enrichments that you create in this Content Mining project and in other projects in the same service instance.

Regular expressions configuration

You can import existing patterns by uploading them in a JSON file or you can add patterns.

To add patterns, complete the following steps:

1. Add the regular expression pattern to the **New pattern** field, and then click **Add**.
2. Specify a name for the pattern, and then identify the facet name to use for this pattern.

   The facet name that you specify for the annotator is the facet name that is displayed from the collection search view.
3. **Optional**: Specify a facet value. You can specify a value from the options that are described in the table.

Facet value	Description
`$0`	Displays the matched text as-is.
`$n`	If your regular expression pattern contains groups, you can specify a group number to return the matched text from the pattern group only. For example, if your regular expression consists of 3 groups that define a US phone number pattern, such as `(\d{3})-(\d{3})-(\d{4})`, and you want to return only the area code portion of the phone number, you can specify `$1`. If the matched text is `212-555-1234`, then the facet value is displayed as `212`. Only specify a group as the facet value for patterns that you know will return matches.
`{prefix-text}:$0`	Adds hardcoded text in front of the facet name. You might want to use this option if you want to distinguish facets that are generated by this regular expression from facets that are similar but generated in some other way. For example, `MyRegex:$0` results in a facet named `MyRegex:212-555-1234`.

Regular expression facet value options

4. Click **Save**.

To import patterns, complete the following steps:

1. Define the patterns that you want to add in a JSON file.

   The pattern definition must use the following syntax:

   ```
   [
      {
         "name":  "US Phone number",
         "description":  "US mobile phone number",
         "pattern":  "(\\d{3})-(\\d{3})-(\\d{4})",
         "facetPath":  ".regex.usphonenumber",
         "facetValue":  "$0"
      }
   ]
   ```

   Keep the following notes in mind:

   - The patterns must be defined in an array, even if you plan to define only one pattern.
   - Escape any backslash (`\`) characters with a backslash.
   - For more information about the facet value options, see the *Regular expression facet value options* table.

2. Click **Import**, and then choose the JSON file where the patterns are defined.

3. Click **Save**.

Regular expression limits

Plan	Regex enrichments per service instance	Regex patterns per service instance
Cloud Pak for Data	Unlimited	Unlimited
Premium	100	50
Enterprise	100	50

Regular expression plan limits

Totals include enrichments that you create in this Content Mining project and in other projects in the same service instance.

Applying the annotator

After the annotator is created, you must apply it to your collection.

1. From the *Create a custom annotator for your analytics solutions* page of the Content Mining application, click **custom annotator**, and then select **collection** from the list.

2. In the tile for your collection, click the *options* icon, and choose **Edit collection**.

3. Click the **Enrichment** tab, and then select the annotator that you created.

You might need to scroll to find it.

4. Click **Save**, and then confirm the action.

Give the index time to rebuild.

Filtering documents with your facet

1. Click the collection tile to open your collection in the data analysis page.
2. Do one of the following things:
    - Your custom facets are listed in the *Facets* view. Scroll and click **Load more** repeatedly until your facets are displayed.
    - Submit an empty search to return all documents. In the *Facet analysis* pane, select the facet that you created.
    - To access your custom facets more quickly, add them to a custom view. Select **Custom** as the view, and then click **Edit**. Select one or more facets to add to the view, and then click **Save**.

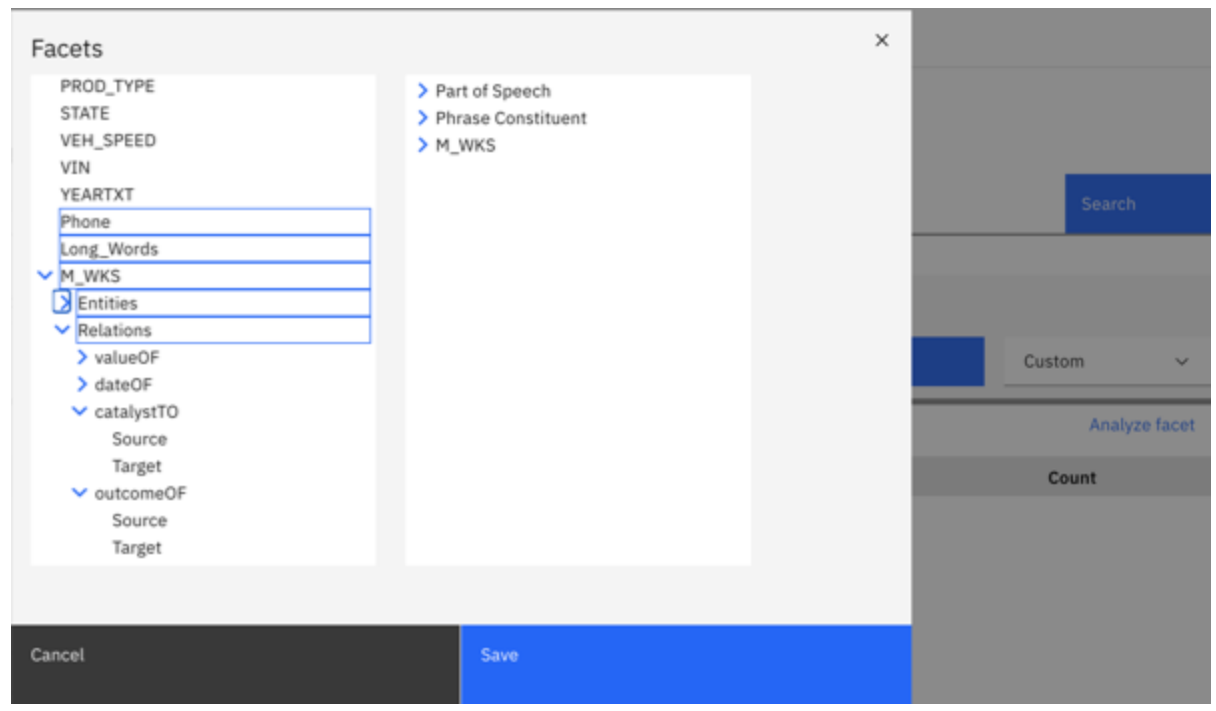


Figure 1. Collection menu

Classifying documents

A document classifier machine learning model analyzes documents and tags them with the appropriate label from a set of labels that you define.

Classifying documents is useful when you want to sort many documents into groups programmatically. For example, you might have a collection that contains customer comments about products that you sell. If you can automatically sort the feedback into classes, you can isolate urgent issues that customers mention and tackle them first. Based on previous feedback, you might define classes such as the following labels:

- Not functioning correctly
- Features not as advertised
- Difficult to use
- Missing parts
- Parts shipped don't match parts list in assembly instructions

To create a document classifier, you build a machine learning model that can recognize which class best captures the point of customer feedback that is specified in natural language. You pair them with class labels that represent real scenarios that make sense for your business.

What's the difference between a document classifier and a text classifier?

> A document classifier can classify documents based on words and phrases extracted from the body text fields with information from their part of speech and the other enrichments that are applied to the body text taken into account. The information from the other non-body fields are also used. A text classifier can classify documents based on words and phrases extracted from the body text with their part of speech information taken into account. For more information about how to create a text classifier, see Classifier.

Before you begin

To train the document classifier model, you must provide sample documents that are labeled appropriately. Prepare the following files:

Training data

Required. CSV file that is used to train the document classifier machine learning model. The file can contain key data points per column. The data points can vary, but the file must include the following columns:

- Natural language text that you want to classify or label.
- Label or class name that categorizes the idea that is expressed in the document text. You can apply more than one label to a text sample. Separate multiple label values with a semicolon.

Test data

Optional. CSV file that is used to test the document classifier machine learning model after it is trained. If you don't specify a separate file for testing, a subset of the training data content is used for testing purposes.

Target data

Required. CSV file with the data that you want to classify.

⚠ **Important:** All of the CSV files (training, test, and target) must have the same column names. The data in the columns must have the same data types, such as string, number, and so on.

You can use a CSV file that you uploaded at the time that you created the Content Mining project or you can create a new collection.

For more information, see the following topics:

- Adding collections
- Analyzing CSV files

Document classifier training data sample

The following table shows an example of the type of content that might be stored in CSV files that are used to train a document classifier.

Claim_id	Date	Product_line	Product	Client_segments	Client_location	Client age	Feedback	Label
0	2016/1/1	tea	lemon tea	Not Member	Manhattan	20	The straw was peeled off from the juice pack.	package_container
1	2016/1/2	ice cream	vanilla ice cream	Silver Member	Queens	20	I got some ice cream for my children, but there was something like a piece of thread inside the cup.	contamination_tampering

Table 1. Sample data for CSV files

Note that the two required fields are present in the sample. The required fields have the following names:

- `Feedback`: Natural language text to label.
- `Label`: Label to apply to the feedback.

Opening the Content Mining application

If you didn't do so, create the project and add a collection to it. If you already created the project and collection, you can skip this procedure and create the document classifier.

1. In Discovery, create a Content Mining project.

2. Choose to upload data to create the collection. Name your collection, and click **Next**.
3. Upload the CSV file that contains your training data.

   The training data file must contain the following information at a minimum:

   - A column that contains sample text that you want to classify. For example, the sample text might be a product review.
   - A column that contains a class or category label that is assigned to the sample text.
4. After collection processing is complete, click **Launch application** to open the Content Mining application.

   The facet details are displayed for the collection.

Creating a document classifier

To create a document classifier, complete the following steps:

1. From the Content Mining application, click the **Collections** link in the breadcrumb to open the *Create a collection* page.

   The status of index creation is displayed. Wait for the collection to be fully indexed before you continue with this procedure.
2. To create a classifier, click **collection**, and then choose **classifier** from the list.

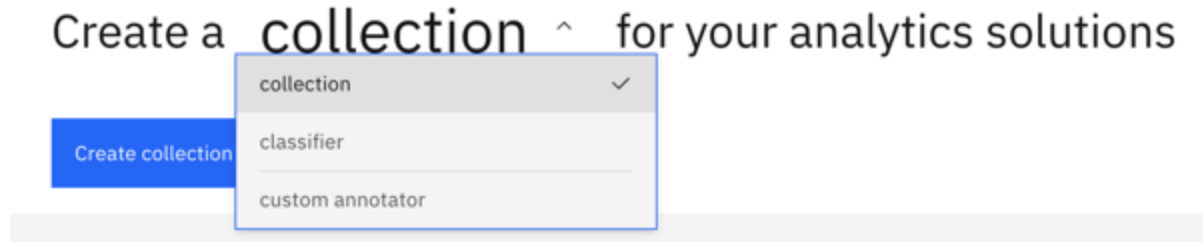


Figure 1. Collection menu

3. Click **Create classifier**.
4. Name your classifier.

   When you deploy the model as an enrichment later, the enrichment is given a name with the format `{classifier name} - {model name}`. For example, if your classifier is named `Product reviews` and the model is named `v0.1`, then the enrichment name is `Product reviews - v0.1`.

   Optionally, add a description and identify the language of your training data by selecting it from the *Language* field.
5. Click **Next**
6. On the *Training data* page, select the file that you uploaded previously from the list, and then click **Next**.

   Alternatively, you can upload a CSV file that contains your training data.

   The *Fields* page is displayed. It shows details about the fields that are generated from the file that you added. Typically, each column in a CSV file is converted into a field and is assigned a name that is copied from the column header.
7. Deselect any metadata fields that you want to exclude from the data set for your document classifier to learn from, and then click **Next**.

   Any fields that you include are used as additional features in the classification. All of the fields are selected by default. You might need to scroll horizontally to review all of the fields.
8. On the *Classifier* page, specify the fields to use for machine learning training and prediction.

   Answer field

   Select the field from your training data file with the classification label. From the earlier example, the `Label` field is the best choice.

   Predicted field

   The name of the facet that is generated for the predicted class values. By default, the facet name has the syntax `<Answer field value>_predicted`. For example, `Label_predicted`.

   Test dataset

   Specifies the data set to use to test the classifier model. By default, the training data CSV file that you uploaded and configured is split into three data sets that are used for training, validation, and test respectively. However, you can optionally specify a separate data set to use for testing the model.

   Train federated model

   Creates more than one model, based on values from a specific field in the data set. For example, if the document has a `Product` field, you can

configure the classifier to create a separate classifier model for each product name value that is specified in the field. By default, the classifier creates one machine learning classifier model.

> **Note:** You don't need to specify the field that contains the text to be classified. The system detects this field automatically. You can check which field the analyzable text is extracted from and change it or augment it by changing index type of another field. For more information, see Identifying the text field.

Click **Next**.

9. If you want to apply an enrichment to the text in your training data, select at least one field from the **Target fields** list where you want to apply enrichments.

   Typically, you want to choose the field that contains the body of text that you want to classify. From the earlier example, the `Feedback` field is the best choice.

   Next, select any annotators that you want to apply to enrich the text in the target field or fields, and then click **Next**.

   The *Part of speech* annotator is selected by default.

10. On the *Confirm* page, review your classifier configuration settings. To make changes, use the **Back** button. Otherwise, click **Save**.

    An *Overview* page is displayed.

11. Click **New model** to create and train your machine learning model.

12. You can optionally change the name of the model and add a description.

    You can change the default ratio values that are specified for the following data sets:

    - Training dataset: Updates the weights of the training model.
    - Validation set: Monitors the accuracy of the training model during training. The accuracy result is used to draw a training loss graph.
    - Test dataset: Calculates the score of the trained model.

13. Click **Create**.

    It might take several minutes for model training to complete.

Deploying the document classifier model

After the model is trained, deploy the model as an enrichment.

1. Click the overflow menu icon in the **Actions** column, and then click **Deploy model**. Specify the name and other details, and then click **Deploy**.
2. Do one of the following things:

   - To apply the document classifier to a collection in your Content Mining project, see Enriching your collection.
   - To apply the document classifier to a collection in a different project, complete the following steps:

     1. In Discovery, create or open the collection that has the documents that you want to classify.

        > **Note:** The data in the collection where you apply the enrichment must have the same fields as the collection that you used to train the model.

     2. In the **Enrichments** tab, locate your classifier in the **Name** column. From the **Fields to enrich** field, choose the same text field that was used to train the model. (This field is determined by the system and is indexed as the *Analyzable text content* field. For more information, see Identifying the text field.)

     3. Click **Apply changes and reprocess**.

Results of classification

After the enrichment is applied to a collection, a facet is generated that you can use to find the predicted classes. In this example, the predicted field is named `label_answer_predicted`.

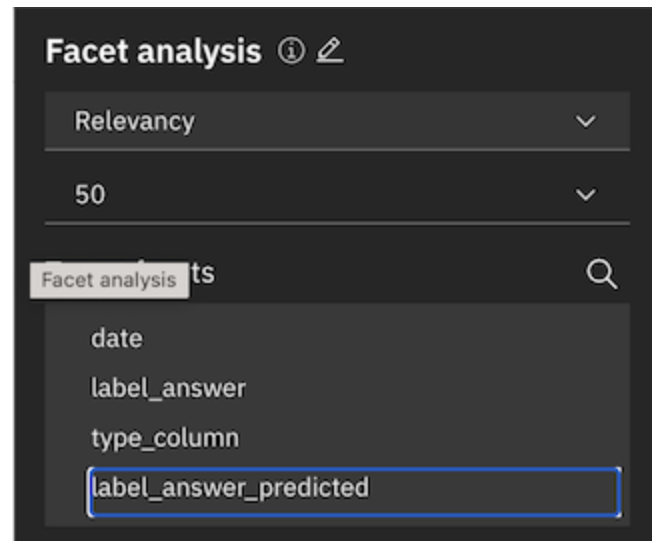


Figure 2. A Label_answer_predicted facet is generated

Use the generated facet to filter documents by classification and analyze subsets of documents. Doing so helps you to find patterns and discover other insights. You can export these target documents to share with team members or to analyze further. For more information, see Exporting data.

When the document classifier classifies a document, it stores the classification in the `document_level_enrichment.classes.class_name` field.

For example, the following JSON excerpt shows a document that was classified with the `package_container` class.

Figure 2. Document classifier enrichment syntax

Document classifier limits

The number of document classifiers and labels that you can create per service instance depends on your Discovery plan type.

Limit	Enterprise	Premium	Cloud Pak for Data
Number of document classifiers per service instance	20	20	Unlimited
Number of labeled data rows	20,000	20,000	20,000
Maximum size in MB of training data after enrichment	1,024	1,024	1,024
Number of labels	1,000	1,000	1,000
Number of target fields	50	50	50

Document classifier plan limits

Detecting phrases that express sentiment

Analyze a document to find phrases that express an opinion or reaction and assess whether the sentiment expressed is positive, neutral, or negative. For English and Japanese, you can also detect specific sentiment targets. The Content Mining application marks these extractions as annotations.

For example, if a product feedback form contains the following sentence, you want to find it and indicate that it is a `positive` statement.

```
I love my XYZ blender...
```

What's the difference between phrase and document sentiment?

> Document sentiment is a built-in Natural Language Processing enrichment that is available for all project types. Document sentiment evaluates the overall sentiment that is expressed in a document to determine whether it is positive, neutral, or negative. Phrase sentiment does the same. However, phrase sentiment can detect and assess multiple opinions in a single document and, in English and Japanese documents, can find specific phrases. For more information about the document sentiment enrichment, see Sentiment.

Complete the following steps to enable phrase sentiment analysis:

1. From the analysis view of your collection, click the **Collections** breadcrumb link in the page header.
2. In the tile for your collection, click the *open and close list of options* icon, and then choose **Edit collection**.
3. Click the **Enrichment** tab, and then select the **Sentiment of phrases** annotator.
4. Click **Save**, and then click **OK** to verify the change.

   The collection is reindexed. Wait for processing to be completed.
5. Click **Close** to return to the *Collections* page, and then click your collection tile.
6. In the *What do you want to analyze?* field, enter a term to search for in your documents or select one or more facets, and then click **Search** to filter the documents.

   The search results are displayed in the mining graph. The *Facet analysis* pane is displayed also. By default, *Relevancy* analysis is shown.
7. In the drop-down menu from the *Facet analysis* pane, select **Sentiment**.
8. In *Target facets* from the *Facet analysis* pane, expand the **Sentiment Analysis** option to see facets that are available for analysis in your documents.
9. Click a facet to explore.

   For example, if you click **Positive Expression**, you can see the following information:

   - Positive expressions that were identified in your documents
   - Sentiment percentage
   - Side-by-side comparison of positive and negative expressions
   - Number of instances of the expression
   - Expression relevancy
10. Click one or more options in the facet list, or select one or both facet lists, and then click **Analyze more**.

    View the phrase, expression, or target in the *Documents* or *Trends* views.

Note: Text from the body field of the document is analyzed. For more information about which field is used for the body text, see Identifying the text field.

Adding collections

You can add a collection directly to the Content Mining application.

You might want to add a collection from within the Content Mining application to make data available for use as training data for a document classifier, for example.

The collection can contain an uploaded CSV file only. For information about file guidelines, see Analyzing CSV files.

The collection that you create is not added to your existing Content Mining project. A new Content Mining project is created to store the collection. The project that is generated is given the name that you specify for the collection.

To add a collection, complete the following steps:

1. From the analysis view of your collection, click the **Collections** link in the breadcrumb to open the *Create a collection for your analytics solutions* page

of the Content Mining application.

2. Click **Create collection**.
3. Drag your CSV file to the *Import your files* dialog, or click Open to browse for the file. When the button is available, click **Next**.
4. You can optionally customize the columns that you want to include or exclude from the collection, and adjust the data types of the fields from the *Fields* page. Click **Next**.
5. From the *Enrichments* page, you can optionally apply or remove any enrichments from the collection, and then click **Next**.

   The *Part of Speech* enrichment is applied automatically.
6. On the *Facets* page, you can optionally customize the data that is displayed for facets. Click **Next**.
7. Click **Save** to save and index the collection.

Editing your collection

You can change the characteristics of your collection from the Content Mining application.

You can change the following characteristics:

- Change the time zone of your collection
- Add document flags that you can use to tag documents of interest in your collection
- Change or augment the field that is designated as the source for the text body of your documents
- Group text body fields
- Add, remove, or change the enrichments that are applied to the collection

Edit a collection

1. From the analysis view of your collection, click the **Collections** link in the page header.
2. In the tile for your collection, click the *Open and close list of options* icon, and then choose **Edit collection**.
3. Use the appropriate tab to change characteristics of the collection.
4. When you are done making changes, click **Save**.

   The following message is displayed:

   ```
   You need to clear index to make these changes.
   After clearing index, fully build the index to
   analyze using this collection.
   ```

   You can ignore the message. The index is rebuilt automatically when you click *OK*.
5. Click **OK** to verify the change.
6. Click **Close** to return to the *Collections* page.

   Tip: Wait for the index to be rebuilt before you continue your analysis. From the *Collections* page, you can see the progress of the index rebuild.
7. Click your collection tile to return to the data analysis page.

Change the time zone

To change the time zone that is used by the trend graph, you must edit the default time zone for the collection.

1. Complete the steps in Editing a collection to get the collection into edit mode.
2. In the *Edit* tab, change the value of the **Time zone** field, and then click **Save**.

Add document flags

To add document flags, complete the following steps:

1. Complete the steps in Editing a collection to get the collection into edit mode.
2. Click the **Document flags** tab, and then click **Add flag**.
3. In the *Document flag* dialog box, name the flag, add a description, choose a flag color, and then click **Add**.
4. Repeat the previous steps to add more flags.
5. From the *Document flags* view, select **Enabled** so that the flags appear in your documents, and then click **Save** to make them available in your

collection.

For more information about how to flag documents, see Flag documents of interest.

Identify the text field

When you analyze data with the Content Mining application, Discovery determines which field contains the *body* of the text to be analyzed. It does so by looking for the field with the highest average word count.

You can check which field is designated as the main text body field, and change it or augment it by changing the index type of another field.

1. Complete the steps in Editing a collection to get the collection into edit mode.
2. Click the **Fields** tab. Check the **Index type** column to find the field designated with the **Analyzable text content** index type.

   You can change the field or set more than one text field to be an **Analyzable text content** index type.
3. Click **Save**.

If you select multiple fields to analyze, you cannot see the facet analysis for only one field. To view the analysis for multiple fields, you must group them.

Group multiple text fields

1. Complete the steps in Editing a collection to get the collection into edit mode.
2. Click the **Contextual view** tab, and then click **Add view**.
3. Complete the following fields:
   - **Name**: The name or label of your grouped view.
   - **Id**: The alphanumeric ID that Discovery uses when you submit a text query. For example, `ans1`.
   - **Fields**: The text fields that have the **Analyzable text content** setting applied. Select one or multiple text fields that you want to group for facet analysis.
4. Click **Add**.

   Repeat this task if you want to add more text fields that you want to group for facet analysis.
5. Click **Save**.

Now you can return to the data analysis page for your collection. From the **Facet analysis** panel, you can click **Contextual view selection** to see the text fields that you grouped. You can select one of the text fields to view the facet analysis for that field.

Enrich your collection

Discovery provides built-in natural language processing models, such as the *Entities* enrichment that can recognize mentions of commonly known things, such as business or location names and other types of proper nouns. You can apply these built-in NLP enrichments to your collection.

You can also apply a document classifier enrichment that you created in the Content Mining application to your collection.

Alternatively, you can apply enrichments that were built in other projects in the same service instance to the collection in your content mining project. For example, you can apply a dictionary or text classifier that was built in another project in the same service instance to your collection.

To apply enrichments to your collection, complete the following steps:

1. Complete the steps in Editing a collection to get the collection into edit mode.
2. Click the **Enrichment** tab, and then select the enrichments that you want to apply to your collection.
3. Click **Save**.

Analyzing CSV files

You can add the data that you want to analyze as a comma-separated value (CSV) formatted file.

The content mining project works well with CSV files. When your CSV file is ingested, each row in the spreadsheet is stored as a separate document in the collection index. Each column becomes a root-level field in the document.

Follow these guidelines when you create a CSV file for use in the project:

- Add each record that you want to analyze as a row in the spreadsheet.
- Include a column for each significant data point.
- Specify column headers.

  The root-level field that is added to the document is given the column header name. If no header exists, hardcoded names, such as *column_0* and

column_1, are applied to the columns. Specify column names to ensure that the resulting document fields have meaningful names.

- If you want to find trends over time, be sure that each record has some date information that can be used to plot the information on a timeline.

  Discovery recognizes the following date formats automatically:

  ```
  yyyy-MM-dd'T'HH:mm:ssZ
  yyyy-MM-dd'T'HH:mm:ssXXX
  yyyy-MM-dd'T'HH:mm:ss.SSSZ
  yyyy-MM-dd'T'HH:mm:ss.SSSX
  yyyy-MM-dd
  M/d/yy
  yyyyMMdd
  yyyy/MM/dd
  ```

  If you store dates in other formats, you can add the format to the list of supported formats.

  From the Discovery user interface, open the *Manage collection* page. Click your collection tile. From the *Manage fields* page for the collection, add a format to the **Date formats** field. Specify a date format that is supported by the Java SimpleDateFormat class.

  For example, if your records store only year values for dates, add `yyyy` to the supported date formats list. You can then set the data type for the field that contains a year value to *Date*, and reprocess your collection. As a result, an occurrence of `2019` in the date field is stored as `2019-01-01T05:00:00Z` in the index.

Sample CSV file

The following image shows an excerpt from a CSV file with data that is well suited for analysis with the Content Mining application. The data comes from 2010 traffic records that are published by the National Highway Traffic Safety Administration (NHTSA). Each record includes car make, model, and year information, the date of the traffic incident, and text from the driver's statement, along with other useful data points.

	MAKETXT	MODELTXT	YEARTXT	CRASH	FAILDATE	FIRE	COMPDESC	CITY	STATE	DATEA	LDATE	MILES	CDESCR
2	TOYOTA	SIENNA	2010	N	20100101	N	ENGINE AND ENGINE COOLING	AVON	IN	20100101	20100101	3000	ENGINE SPEED CONTROL , IT DOESN'T GO UP SMOOTHLY FROM
3	FORD	EXPLORER	2002	N	20100101	N	LATCHES/LOCKS/LINKAGES	LOUISVILLE	KY	20100101	20100101	176000	2002 FORD EXPLORER DOOR LOCKS WILL NOT FUNCTION PROPI
4	FORD	FREESTAR	2005	N	20100101	N	POWER TRAIN:AUTOMATIC TRANSMISSION	NILES	MI	20100101	20100101	55000	WE WERE IN MY WIFE'S 2005 FORD FREESTAR DRIVING HOME FR
5	MERCEDES BENZ	E430	2000	N	20100101	N	AIR BAGS:FRONTAL	VIRGINIA BEACH	VA	20100101	20100101		ON E-CLASS MERCEDES, PASSENGER SEAT HAS FUNCTION TO C
6	CHEVROLET	IMPALA	2007	N	20100101	N	POWER TRAIN:AUTOMATIC TRANSMISSION	TEMPE	AZ	20100101	20100101	40000	TRANSMISSION "SLIPS" THEN ENGAGES HARD. HAS PROGRESS
7	JEEP	LIBERTY	2002	N	20100102	N	WHEELS	AF	UT	20100102	20100102		FRONT LUG NUTS LOOSEN ON 2002 JEEP LIBERTY. THIRD TIME (
8	JEEP	GRAND CHEROKEE	2002	N	20100101	N	ELECTRICAL SYSTEM:IGNITION	MINNEAPOLIS	MN	20100102	20100102	98400	KEY WON'T TURN IN IGNITION, STEERING WHEEL LOCKED, CAR
9	JEEP	GRAND CHEROKEE	2002	N	20100101	N	STEERING	MINNEAPOLIS	MN	20100102	20100102	98400	KEY WON'T TURN IN IGNITION, STEERING WHEEL LOCKED, CAR
10	CHEVROLET	HHR	2007	N	20100102	N	SERVICE BRAKES, HYDRAULIC	MOORESVILLE	NC	20100102	20100102	30000	WITH 61,000 MILES ON MY 2007 CHEVY HHR, I AM HAVING TO RE

Figure 1. Sample CSV file

For more information about the sample data, see https://www.nhtsa.gov/data/traffic-records.

Creating a report

If you discover insights as you analyze your data, you can save and share them with others by creating a report. A report consists of snapshots and notes about the analysis.

Take a snapshot

1. Click the camera icon from the dashboard toolbar.

   Note: You can also take a snapshot of the document preview. When you select one or more documents, only the selected documents are stored and displayed in the report. When no selection is made, all documents in the current page are stored and displayed in the report.

2. Thumbnails of the snapshot are displayed in the *Report* pane, which is a temporary store for snapshots.

   This store is cleared when the browser is refreshed or another collection is opened.

3. From the menu icon of the snapshot's thumbnail, you can enter comments, or delete the snapshot. You can also edit comments later.
4. Choose thumbnails that you want to add to a new report, and then click **Create**.

Create a report

To create a report, complete the following steps:

1. From the *Report* pane, click **Create**.
2. On the *Basic* tab, name and date the report.
3. **Optional**: On the *Comments* tab, edit the title of the analysis result, and enter a comment.
4. Review the preview on the *Preview* tab.

5. When you're done editing, click **Save**.

Your report is added to the *Report* tab on the application launch page. From the *Actions* menu, you can copy the link for your report to share it with others.

Exporting data

If you discover insights as you analyze your data, you can export the data to share with others or analyze further in another business insights tool, for example.

You can export your data as a CSV file or you can generate a separate JSON file for each record.

IBM Cloud Pak for Data For IBM Cloud Pak for Data only, you cannot export secured collections. For more information, see Supporting document-level security.

To export your data, complete the following steps:

1. Submit a search to find the documents of interest.
2. Click **Show documents** to open the *Documents* view, and then click the *Export* icon in the toolbar.
3. Complete the appropriate steps for the format in which you want to export the data.
    - If you want to export the data in JSON format, complete the following steps:
        1. Choose **Export JSON** to generate one JSON file for each record.
        2. **Optional**: You can change the following values:
            - Name. The file is named `export_document_{today's_date}` by default.
            - Encoding. `UTF-8` is used by default.
            - Choose whether to include fields and facets. They are excluded by default.
    - If you want to export the data in CSV format, complete the following steps:
        1. **Optional**: To customize the CSV output, choose **Export CSV with advanced options**.

           You can define the format of the following elements:

            - Text content field: This is the main body field (or fields, if you configured more than one field with analyzable text). You can choose to exclude it from the export. It is exported as a column for a fact table by default.
            - All other fields: You can choose to export them as columns for fact tables or export them as dimension tables. They are excluded from the export by default.
            - Facets: You can choose to export the facets as separate CSV files that can be used as dimension tables. They are excluded from the export by default.

           After customizing the CSV format, click **Save**, and then click the *Export* icon from the toolbar again.

           Note: If you use the same web browser to export data in CSV format again later, your saved settings are applied automatically.

        2. Choose **Export CSV**.
        3. **Optional**: You can change the following values:
            - Name. The file is named `export_document_{today's_date}` by default.
            - Encoding. `UTF-8` is used by default.
            - Date and time format. `Unix epoch time` is used by default.
4. Click **Export**.

Analyzing data on demand with the Analyze API

Use the Analyze API to process text documents through the enrichment pipeline of the Discovery service without storing any data from the source documents.

Note: The Analyze API is supported by Enterprise plan deployments and installed deployments only.

This approach is ideal for business automation purposes. For example, if you want to classify emails, you can use the Analyze API to synchronously call Discovery to get a classification of the email. Then, you can use the output of that classification in your business logic.

Important: The Analyze API supports JSON documents only.

When you analyze a document with the API, you indicate how you want the document to be processed by specifying the collection to associate with the analysis. The document is not stored in the collection. Instead, the configuration settings of the collection are applied to the document. For example, if you want to find entity references in a document, run the Analyze API against a collection where the *Entities* enrichment is applied. The resulting document analysis identifies any entity mentions in the document.

Submit a request for analysis against only one collection that is configured with the enrichments that you want to use to analyze your document on demand. Remember, the documents in the collection are not significant. It is the enrichments that are defined for the collection that matter. If you submit requests to several collections, then several models are initiated at the same time, which can cause request failures.

The following enrichments are supported in the Analyze API:

- Advanced rules models
- Contracts
- Custom entities
- Dictionary
- Document classifier
- Entities (NLP)
- Keywords (NLP)
- Machine learning and Watson Explorer Content Analytics Studio models
- Regular expressions
- Patterns (Enterprise plan only)
- Sentiment of documents
- Table understanding [1]
- Text classifier

For the complete list of the enrichments that are supported in each language, see Language support.

For more information, see the Discovery API reference.

Analysis example

The data that you submit for analysis must be in JSON format. The text must be specified as a string; it cannot be specified as an array. For example, the following JSON file contains a quotation in the `Quote` field that you want to analyze to find any keyword mentions in the text.

```
{
   "Author":  "Jane Austen",
   "Book":  "Pride and Prejudice",
   "Quote":  "From this day you must be a stranger to one of your parents. Your mother will never see you again if you do not marry Mr. Collins, and I will never see you again if you do.",
   "Year":  "1813/01/01",
   "Subject":"Parental love",
   "Speaker":  "Mr. Bennett",
   "url":  "https://www.gutenberg.org/files/1342/1342-h/1342-h.htm#link2HCH0020"
}
```

You know the name of a collection in your project where the *Keywords* enrichment is configured to be applied to documents in the collection. You can use the API to list your collections to find the ID associated with the collection that you look for by name.

After you get the collection ID, include it in the POST request that you submit to apply the configuration settings from the collection to your JSON file. For example, the following request submits the JSON snippet in a file that is named `favorites2.json` for keyword analysis.

```
curl --location --request POST \
'https://my-cloud-pak-for-data-cluster/discovery/zen-wd/instances/{instance-id}/api/v2/ \
projects/{project-id}/collections/{collection-id}/analyze?version=2020-08-30' \
--header 'Authorization: Bearer ...' \
--form 'file=@"/quotations/favorites2.json"'
```

The result contains a list of keywords that were recognized in the quotation.


```
{
   "result": {
      "enriched_Quote": [
         {
            "keywords": [
               {
                  "text": "day",
                  "mentions": [
                     {
                        "text": "day",
                        "location": {
                           "begin": 10,
                           "end": 13
                        }
                     }
                  ],
                  "relevance": 0.673739
               },
               {
                  "text": "stranger",
                  "mentions": [
                     {
                        "text": "stranger",
                        "location": {
                           "begin": 28,
                           "end": 36
                        }
                     }
                  ],
                  "relevance": 0.596757
               },
               {
                  "text": "parents",
                  "mentions": [
                     {
                        "text": "parents",
                        "location": {
                           "begin": 52,
                           "end": 59
                        }
                     }
                  ],
                  "relevance": 0.568336
               },
               {
                  "text": "mother",
                  "mentions": [
                     {
                        "text": "mother",
                        "location": {
                           "begin": 66,
                           "end": 72
                        }
                     }
                  ],
                  "relevance": 0.755562
               },
               {
                  "text": "Mr. Collins",
                  "mentions": [
                     {
                        "text": "Mr. Collins",
                        "location": {
                           "begin": 118,
                           "end": 129
                        }
                     }
                  ],
                  "relevance": 0.945891
               }
            ]
         }
      ],
      "url": "https://www.gutenberg.org/files/1342/1342-h/1342-h.htm#link2HCH0020",
      "Subject": "Parental love",
      "Year": "1813/01/01",
```

```
      "Book":  "Pride and Prejudice",
      "Author":  "Jane Austen",
      "Quote":  [
         "From this day you must be a stranger to one of your parents. Your mother will never see you again if you do not marry Mr. Collins, and I will never
see you again if you do."
      ],
      "metadata":  {
         "name":  "favorites2.json"
      },
      "Speaker":  "Mr. Bennett"
   },
   "notices":  []
}
```

You cannot submit an array of objects as input. For example, you might want to analyze multiple quotations, so your source might look as follows:

```
{
   "quotations":[
      {
         "Author":  "Jane Austen",
         "Book":  "Sense and Sensibility",
         "Quote":  "Is there a felicity in the world superior to this?",
         "Year":  "1811/01/01",
         "Subject":  "Nature",
         "Speaker":  "Marianne Dashwood",
         "url":  "https://www.gutenberg.org/files/1342/1342-h/1342-h.htm#link2HCH0059"
      },
      {
         "Author":  "Jane Austen",
         "Book":  "Persuasion",
         "Quote":  "A man does not recover from such a devotion of the heart to such a woman. He ought not; he does not.",
         "Subject":  "Romantic love",
         "Year":  "1818/01/01",
         "Speaker":  "Captain Wentworth",
         "url":  "https://www.gutenberg.org/files/105/105-h/105-h.htm#chap20"
      }
   ]
}
```

If so, break each object into a separate file and analyze each file individually.

Analyzing a text snippet

You can submit text for analysis when you specify the text in JSON format by using syntax like this:

```
{
"text":"The text that you want to analyze."
}
```

The following example request shows how to analyze text that you specify in the request, not that you pass in a physical file.

```
curl --location --request POST \
'https://my-cloud-pak-for-data-cluster/discovery/zen-wd/instances/{instance-id}/api/v2/ \
projects/{project-id}/collections/{collection-id}/analyze?version=2020-08-30'  \
--header  'Authorization: Bearer ...'  \
--form  'file={"text": "ISO 9000 is a standard."}'
```

The response might look as follows.

```
{
   "result"  :  {
      "enriched_text"  :  [  {
         "entities"  :  [  {
            "text"  :  "ISO 9000",
            "type"  :  "my_iso_pattern",
            "mentions"  :  [  {
               "text"  :  "ISO 9000",
               "confidence"  :  1.0,
               "location"  :  {
                  "begin"  :  0,
                  "end"  :  8
               }
            }  ],
```

```
            "model_name" : "My ISO Pattern"
         }, {
            "text" : "9000",
            "type" : "Number",
            "mentions" : [ {
               "text" : "9000",
               "confidence" : 0.8,
               "location" : {
                  "begin" : 4,
                  "end" : 8
               }
            } ],
            "model_name" : "natural_language_understanding"
         } ]
      } ],
      "metadata" : { },
      "text" : [ "ISO 9000 is a standard." ]
   },
   "notices" : [ ]
}
```

Analyzing HTML content

You can analyze HTML when you submit the html in JSON format by using syntax like this:

```
{
"html":"<p>My html content.</p>"
}
```

The following example request shows how to analyze text that you specify in the request, not that you pass in a physical file.

The collection to which the request is made uses the following enrichments, which means these enrichments are applied to the content that you submit with the API request:

- Entities
- Keywords
- Table Understanding

Request example

The body of the request contains `form-data` with the name `file`. The value is the JSON content to be analyzed.

```
curl --location --request POST \
'https://cpd-abc.example.com/discovery/abc-wd/instances/1671204318684041/api/v2/projects/d457fcd9-a4ce-4637-a340-
33123b5cbe2c/collections/2d47dbcc-64c7-84e9-0000-01851bb9d998/analyze?version=2020-08-30' \
--header 'Authorization: Bearer ...' \
--form 'file={
 "html":"<html><head>This is my html file</head><body><p>My file contains a table.</p><table><tbody><tr><th>Holiday</th><th>Popular
greeting</th></tr><tr><td>Christmas</td><td>Merry Christma!s</td></tr></tbody></table></body></html>",
 "text":"This is a sentence that contains key words, such as George Washington and Boston, MA."
}'
```

Results

The results show the output of the Entities, Keywords, and Table Understanding enrichments on the `text` and `html` fields that were submitted.

```
{
      "result": {
            "text": [
                  "This is a sentence that contains key words, such as George Washington and Boston, MA."
            ],
            "enriched_text": [
                  {
                        "keywords": [
                              {
                                    "text": "George Washington",
                                    "mentions": [
                                          {
                                                "text": "George Washington",
                                                "location": {
                                                      "begin": 52,
                                                      "end": 69
                                                }
```

```
                        }
                    ],
                    "relevance": 0.952591
                },
                {
                    "text": "Boston",
                    "mentions": [
                        {
                            "text": "Boston",
                            "location": {
                                "begin": 74,
                                "end": 80
                            }
                        }
                    ],
                    "relevance": 0.578079
                },
                {
                    "text": "MA",
                    "mentions": [
                        {
                            "text": "MA",
                            "location": {
                                "begin": 82,
                                "end": 84
                            }
                        }
                    ],
                    "relevance": 0.146905
                }
            ],
            "entities": [
                {
                    "text": "George Washington",
                    "type": "Location",
                    "mentions": [
                        {
                            "text": "George Washington",
                            "confidence": 0.54922265,
                            "location": {
                                "begin": 52,
                                "end": 69
                            }
                        }
                    ],
                    "model_name": "natural_language_understanding"
                },
                {
                    "text": "Boston, MA",
                    "type": "Location",
                    "mentions": [
                        {
                            "text": "Boston, MA",
                            "confidence": 0.66049105,
                            "location": {
                                "begin": 74,
                                "end": 84
                            }
                        }
                    ],
                    "model_name": "natural_language_understanding"
                }
            ]
        }
    ],
    "metadata": {},
    "enriched_html": [
        {
            "tables": [
                {
                    "body_cells": [
                        {}
                    ],
                    "location": {
                        "begin": 99,
                        "end": 183
                    },
                    "row_headers": [],
```

```
                                "key_value_pairs":  [],
                                "section_title":  {},
                                "contexts":  [],
                                "text":  "Holiday Popular greeting Christmas Merry Christmas!",
                                "table_headers":  [],
                                "title":  {},
                                "column_headers":  []
                            }
                        ]
                    }
            ],
            "html":  [
                    "<html><head>This is my html file</head><body><p>My file contains a table.</p><table><tbody><tr><th>Holiday</th><th>Popular
greeting</th></tr><tr><td>Christmas</td><td>Merry Christmas!</td></tr></tbody></table></body></html>"
            ]
        },
        "notices":  []
}
```


Analyze API limits

The following table shows the file size and usage limits for the Analyze API.

Deployment type	File size limit	Concurrent collections limit	Concurrent queries per collection limit
Cloud Pak for Data installed deployment	Unlimited	Unlimited	Unlimited
Enterprise plan managed deployment	50 KB	5	5

Limits that are applied to the Analyze API usage

Use of the Analyze API from Discovery Cartridge for IBM Cloud Pak for Data affects license usage. For more information, see the license information.

Monitoring usage IBM Cloud Pak for Data

You can monitor the usage of the Analyze API from the *API usage* page.

Note: The *API usage* page is available from installed deployments only. For Enterprise plans, analyze method call information is combined with query method call information and is reported as part of the query metrics.

To access the **API usage** page, open the **Projects** page, select **Data usage**, then **API usage**.

Start date

The start date of the API call-monitoring period.

End date

The end date of the API call-monitoring period.

Thirty-day call total

Number of calls to the Analyze API in the 30-day time interval that is indicated by the **Start date** and **End date**. The time interval is determined by calculating the consecutive time period with the highest number of API calls. The 30-day window updates as the time interval with the highest number of API call changes.

Note: The **API usage** is not displayed until some time after API usage monitoring begins. A delay in displaying the final total number of the **30-day call total** might occur, even if the 30-day period that is listed includes the current date.

---

1. For the table understanding enrichment to produce any results, the input must contain a `<table>` HTML element to analyze. ↩

Searching Discovery data from watsonx Assistant

Your Discovery project can provide answers to questions that stump your assistant. Instead of answering with “I don't know”, your assistant can say, “I'm not sure, but I searched my knowledge base and found these answers which might help.”

For more information about how to search a Discovery project from an assistant, read the appropriate watsonx Assistant documentation for your situation.

- From the new experience user interface, see Search trigger.
- From an actions skill in the classic user interface, see Configuring the search for an answer.
- From a dialog skill, see Adding a search skill response type.

Tip: If you use the built-in web chat, you can use answer finding by enabling the *Emphasize the answer* feature. Answer finding highlights the word or phrase in the search result that is determined to be the exact answer to the customer's question.

For a more detailed look at the steps to take to connect to a Discovery project from watsonx Assistant, take a tutorial that walks you through them. For more information, see Power your assistant with answers from web resources.

Alternatively, you can add a generative language service named NeuralSeek between the Watson Discovery and watsonx Assistant services. For more information, see Use NeuralSeek to return polished answers from existing help content.

How the assistant calls Discovery

When a user asks your assistant a question that triggers a search, the following API request is sent to Discovery if *Emphasize the answer* is enabled.

Note: The *Emphasize the answer* feature is available from instances that are managed by IBM Cloud only.

```
{
      "aggregation":  "",
      "sort":  "",
      "count":  10,
      "return":  [],
      "filter": <custom_filter_specified_in_assistant>
      "passages":  {
   "enabled":  "true",
            "fields":  [
              <search_config_body_field_specified_in_assistant>
            ],
            "characters":  325,
            "per_document":  true,
            "max_per_document":  3,
             "find_answers":  true,
             "max_answers_per_passage":  1
      },
      "highlight":  false,
      "spelling_suggestions":  false,
      "table_results":  {
          "enabled":  false
      },
      "suggested_refinements":  {
          "enabled":  false
      }
}
```

When *Emphasize the answer* is used (`"find_answers": true`), Discovery rescores and reorders the documents to ensure that documents with the highest-quality answers are returned first.

Choosing a project type

If the *Conversational Search* project type isn't providing the best answers and you want to understand why, switch to using a *Document Retrieval* project type.

Most often, the *Conversational Search* project type is the right choice. You get great results from the start, and when you enable extra features like *Emphasize the answer*, the answers are clear and concise. However, for advanced use cases, or if you want to be able to troubleshoot issues, a *Document Retrieval* project type might be a better fit.

To help you choose the right Discovery project type, review the project type differences that are described in the following table.

Function	Conversational Search	Document Retrieval

Enrichment support	No default enrichments are applied.	The *Entities* enrichment is applied. The Entities enrichment is helpful for identifying important information and introduces more ways to filter query results.
Testing queries from the *Improve and customize* page in Discovery	You see only one of the responses that are returned from the chatbot. You cannot see all of the available responses and cannot analyze individual query results.	You can filter query results by enrichment-based facets. You can review details about fields that are indexed in the source documents that are returned for a query. Access to more information makes it easier to troubleshoot unexpected results.
Search triggers	Returns answers from the `text` field automatically. If answers are stored in another field, you must change the configuration.	You can apply a Smart Document Understanding (SDU) model or enrichments to your collections and retrieve useful information from fields other than `text` when search is triggered from the assistant.

Project type details

For both project types, the best way to test is to trigger search from the watsonx Assistant preview. When you configure search support for an assistant, you can fine-tune the experience in ways that aren't available in Discovery.

And settings that are available from the *Search results* tool for a *Document Retrieval* project type are replaced by configuration settings that you specify in watsonx Assistant. For example, the query response title and body are defined in watsonx Assistant. And a passage length of 325 characters is applied to responses regardless of what you specify in the **Max characters in a passage** field.

The way that you deploy search support in your chatbot is the same regardless of the project type. You enable search support in your assistant and then publish your assistant.

Note: If you decide that you want to use a *Document Retrieval* project type, you must create it *before* you add the search function to your virtual assistant. Otherwise, when you add search support to your assistant, a *Conversational Search* project is created for you automatically. When you have a service instance that contains a *Document Retrieval* project already, the watsonx Assistant user interface shows the existing instance, and you can choose to use it.

Deploying the built-in UI components

For Document Retrieval and custom projects, a set of user interface components are available for your use.

Work with a developer to use the pre-built UI components that are provided by IBM to deploy an application.

For more information about building your own app, see the Building custom applications with the API.

Several built-in UI components are available.

Search bar

A search box that uses a natural language understanding query to fetch the most relevant results.

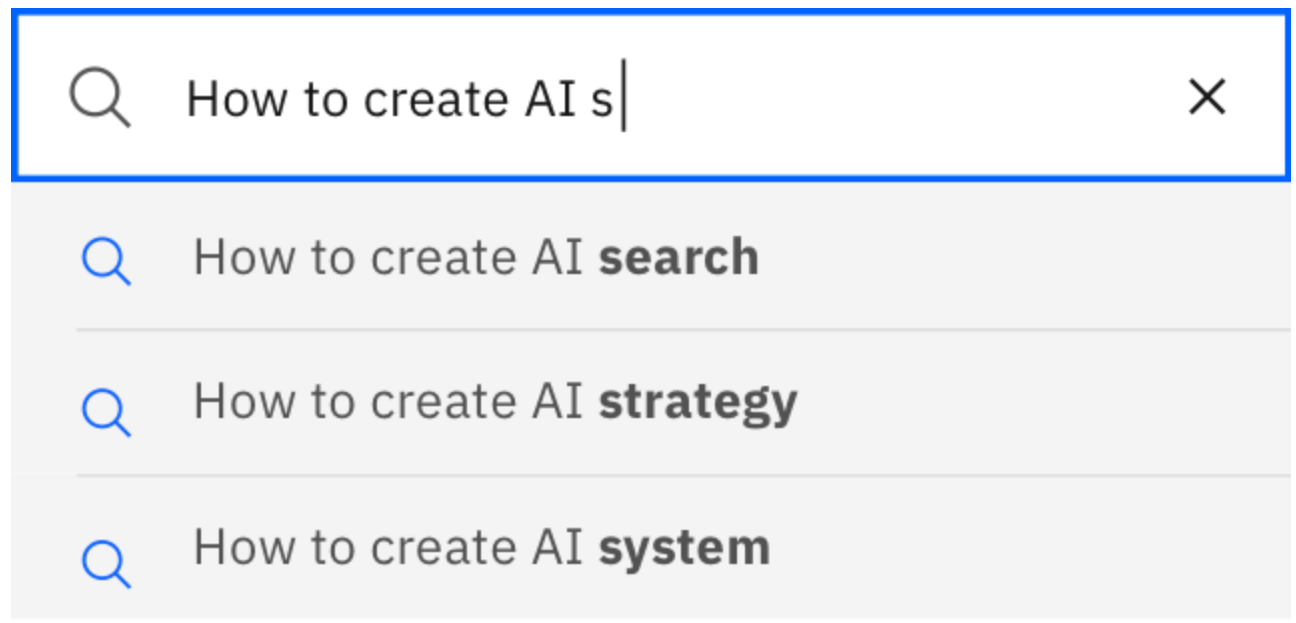


Figure 1. Search bar type ahead

Try it

Search results

A set of results that rank the most relevant passages and tables to a query.

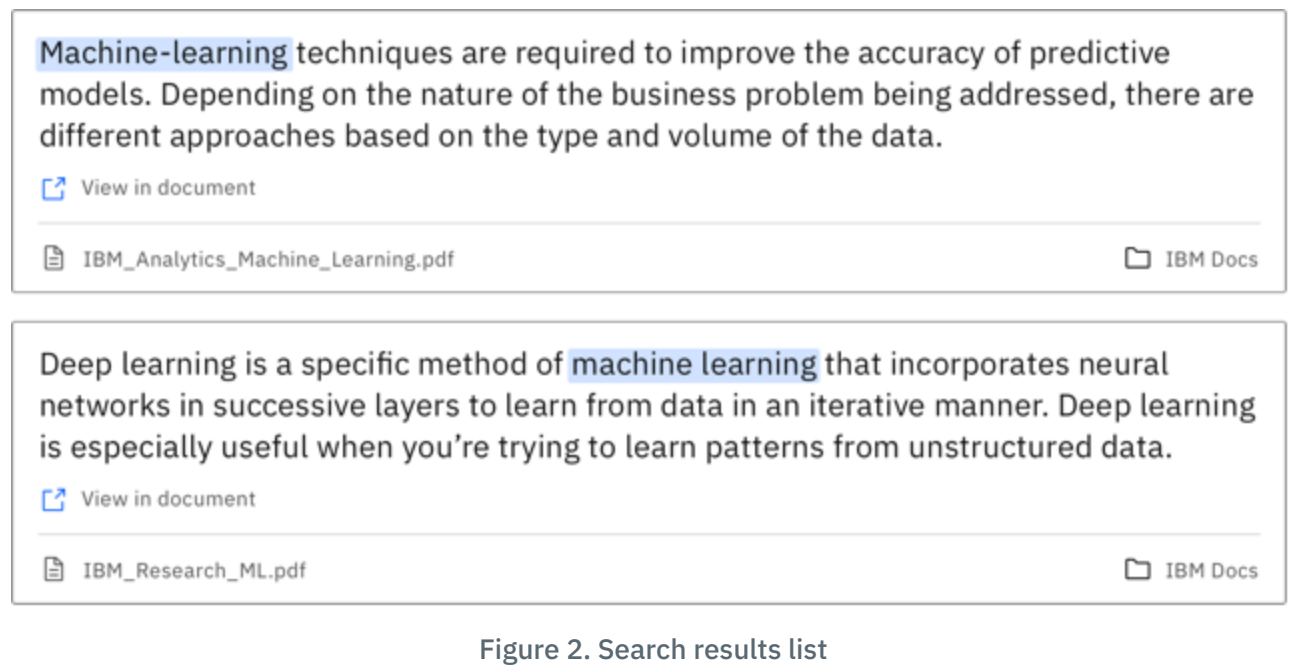


Figure 2. Search results list

Try it

Facets

Refine your results with facets that help users filter the search results by specific categories and domains.

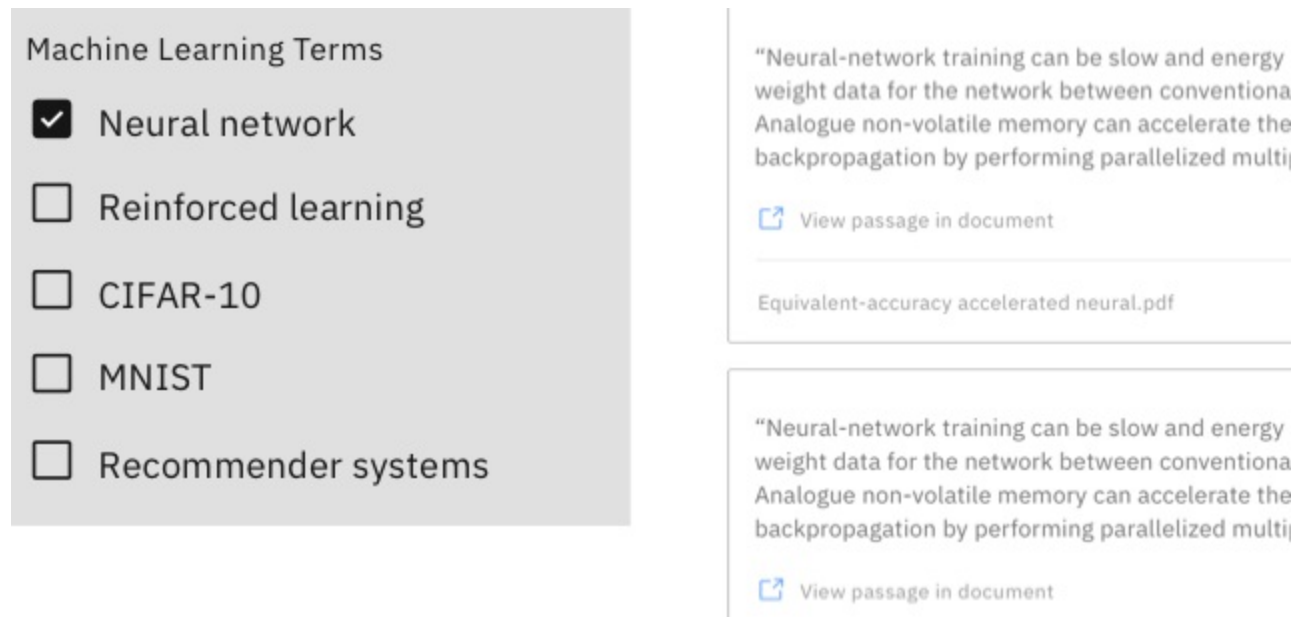


Figure 3. Facets

Try it

Rich document preview

Displays your results in a document preview. This view highlights result passages within the text of the original document. It also shows any enrichment mentions that are detected in the document. The rich preview is available with source documents where an SDU model is applied, such as PDF, Microsoft PowerPoint, and Microsoft Word files.

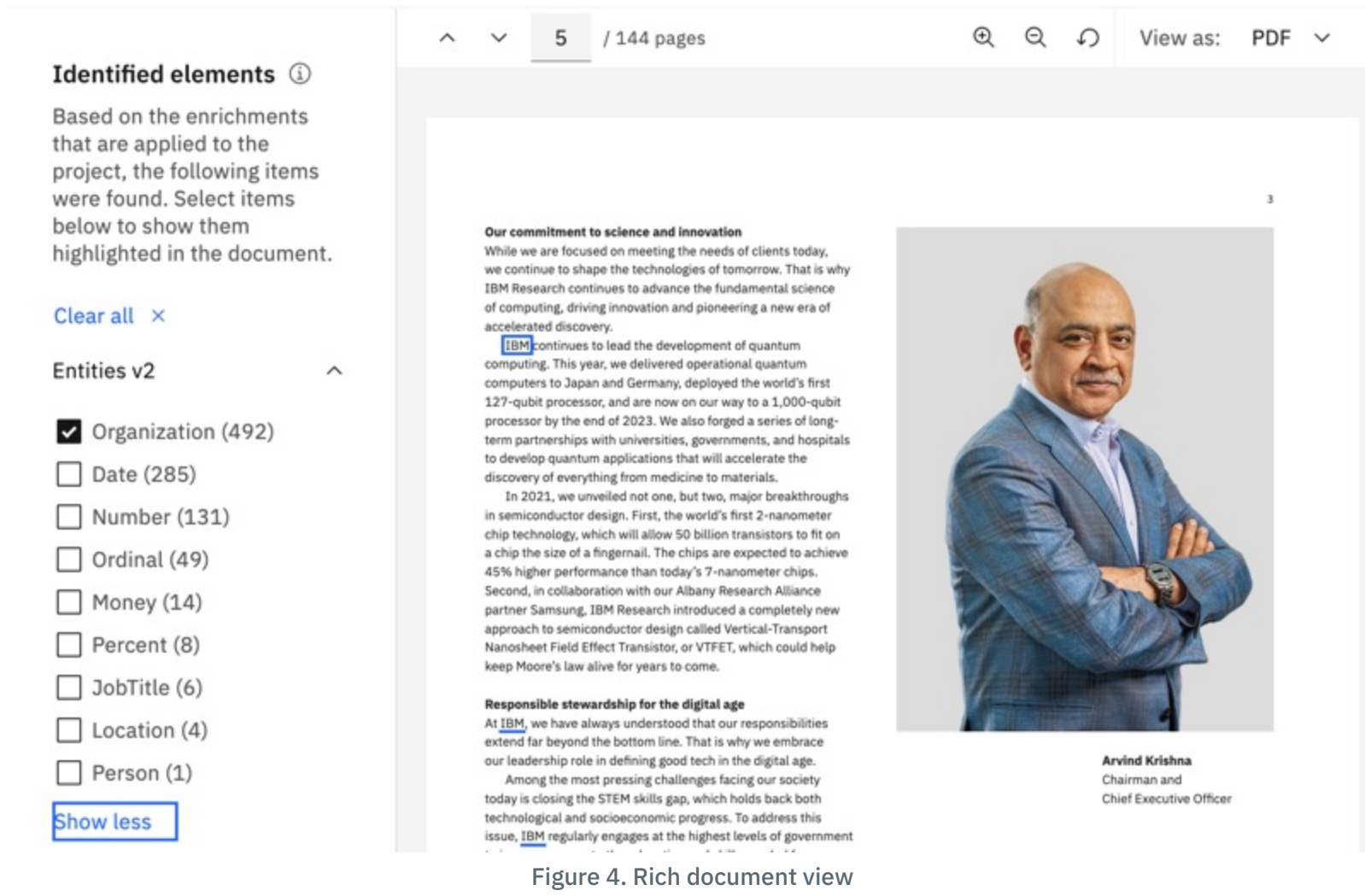


Figure 4. Rich document view

Try it

Deploying a project

To deploy your project, complete the following steps:

1. To use the API, you need to know the project ID for your project. Go to the **Integrate and Deploy** > **API Information** page.
2. From the **Integrate and Deploy** > **UI Components** page, find links to resources that a developer can use to get started.
    - GitHub
    - Storybook

Getting started with the GitHub sample app

From resources available in GitHub, you can run a script to start a sample app with prebuilt UI components. In fact, the sample app looks a lot like the *Improve and customize* page of the product because the product itself uses these UI components.

The script requires some prerequisite software to function. After you start the script, it checks whether you have the necessary software installed on your

system. If not, it lets you know what software you need to install. Install the following packages if they are not installed already:

- Node.js
- Yarn

The script needs information about your service instance and project to use the data and search settings that you configured for your project and apply them to the sample app. You must collect the following information so that you can share it with the script when it asks you for the information later:

Service credentials

The following information is used by the sample app script to construct an endpoint where it can send API requests and to authenticate with your service instance:

- URL
- API key

To get this information, complete the appropriate steps for the type of deployment you are using:

- IBM Cloud From the IBM Cloud Resource list, expand the *AI/Machine Learning* section, and then find the service instance that you created earlier. Click the instance to open its overview page. From the *Credentials* section, copy the URL and API key values and store them somewhere where you can access them later, such as a local text file.
- IBM Cloud Pak for Data From the IBM Cloud Pak for Data web client main menu, expand *Services*, and then click *Instances*. Find your instance, and then click it to open its summary page. Scroll to the *Access information* section of the page, and then copy the *URL* and bearer token. Store the values somewhere where you can access them later, such as a local text file. (The bearer token serves as the apikey for installed deployments.)

Project ID

The unique identifier for the project you created in this tutorial.

You can copy the project ID from the *API Information* tab of the *Integrate and deploy* page.

To run the script that starts the sample app, complete the following steps:

1. Do one of the following things:
   - If you downloaded the repo, extract the files from the archive to a workding directory on your system. Open a command terminal window, and then change to the directory where you downloaded the repository files.
   - If you cloned the repo, open a terminal window from the directory to which you cloned the repository.
2. Enter the following command to start the script:

   ```
   ./runExampleApp.sh
   ```

   Give the script time to set up the necessary resources to run the application.

   If any required prerequisite software packages are missing, the script lets you know what packages you need to install before you can use the script successfully.
3. When prompted to specify the `authType`, enter the type of authentication you use. The type differs based on how your service instance is deployed:
   - IBM Cloud Enter `iam`
   - IBM Cloud Pak for Data Enter `CP4D`.

     The `iam` value indicates that you are using Identity and Access Management, which is a service that is used by IBM Cloud to authenticate its managed services. For installed instances that are deployed on IBM Cloud Pak for Data, `CP4D` is specified instead.

   For the next three prompts, enter the information that you copied and saved earlier.
   - url
   - apikey
   - project_id

When the script is done, it asks if you want to start the sample app now. Enter `y` for yes. A new web browser window or tab is displayed and the sample app is rendered in the page. The URL for the sample app is `http://localhost:3000/`, which means that the app is running locally and cannot be accessed by anyone who is using a different computer.

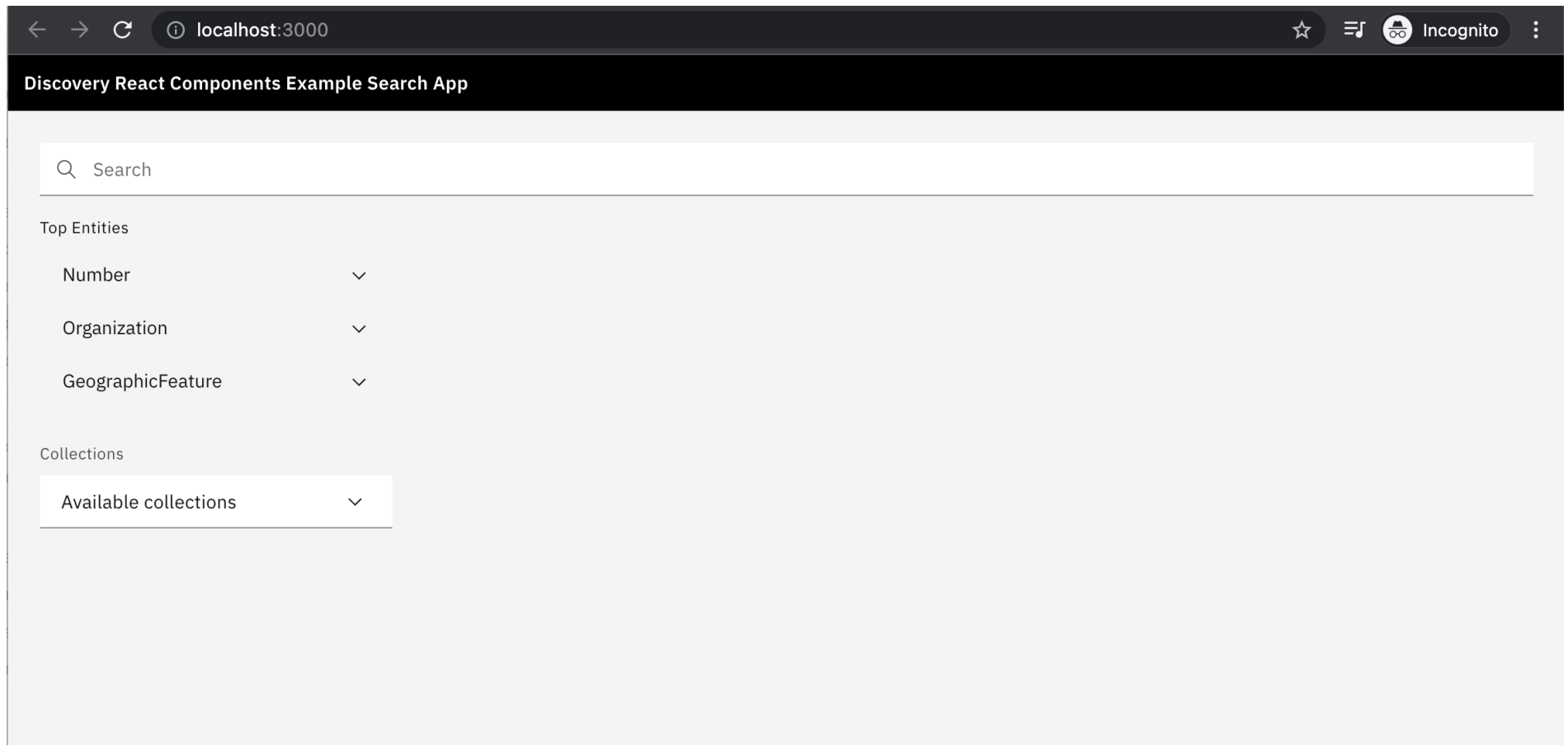


Figure 5. Sample app user interface

The sample app gives you a preview of your search project. Use it to test your search project and make any necessary adjustments.

When you're done testing with the sample app, you can stop it by returning to the terminal window where you ran the initial script, and pressing `Ctrl + C`.

Deploying other project types

To analyze data from a Content Mining project, click **Launch application** from the *Improve and customize* page. For more information, see Analyzing your data.

To analyze a contract from a Document Retreival for Contracts project, submit a query. For more information about understanding contract information, see Understanding contracts.

To deploy a Conversational Search project, connect this project to an assistant that is built with watsonx Assistant. The general steps to follow include:

1. Create an assistant.

   You can use a watsonx Assistant Trial plan for testing purposes.

2. Add a search skill to your assistant, and then connect it to this project.

3. Deploy your assistant.

   For more information about building a watsonx Assistant search skill, see the appropriate documentation for your deployment:

   - IBM Cloud From the new experience, see Adding a search integration.
   - IBM Cloud From the classic experience, see Embedding existing help content.
   - IBM Cloud Pak for Data Creating a search skill.

Choosing a deployment solution

Discovery is available both as a service that is hosted by IBM Cloud and as a service that you install on IBM Cloud Pak for Data. Learn about these deployment solutions and how they differ.

The product user interface and APIs are mostly equivalent regardless of whether you use the managed or installed version of the service. The few differences between the two solutions include:

- How you deploy and set up the service
- The underlying technology that is used to crawl data sources
- Limits for things like the maximum number of documents and enrichments or file sizes
- Who to contact for support and how you share product feedback

Although the documentation that describes how to use the product is the same (what you're reading now), more documentation about how to install and administer the service in IBM Cloud Pak for Data is available from the IBM Cloud Pak for Data product documentation that is hosted in IBM Documentation.

To keep up with product changes, check the following topics periodically:

- IBM Cloud Pak for Data Release notes for IBM Cloud Pak for Data service instances
- IBM Cloud Release notes for IBM Cloud service instances

Comparing features

The following table describes the feature support differences between the two deployment types.

Note: The features that are listed in the IBM Cloud column apply to service instances that are deployed from IBM Cloud Pak for Data as a Service also.

Feature	IBM Cloud	IBM Cloud Pak for Data
Crawl the local file system, Window file system, databases, LDAP directories, FileNet P8, and HCL Notes		
Schedule crawls with more precision		
Apply document-level security to crawled collections		
Enable JavaScript execution for web pages that you want to crawl		
Crawl IBM Cloud Object Storage		
Preview .pdf files that are crawled from external data sources		
Build custom crawlers		
Use App Connect to crawl other external data sources		
Apply answer finding to search queries		
Optical Character Recognition v2		
Patterns enrichment		
App switcher menu where you can get service instance information and usage statistics		
Import and apply UIMA text analysis models created in Watson Explorer Content Analytics Studio		
Monitor usage with activity tracker events		

Monitor usage of the Analyze API from the product user interface	

Feature support details

Installing

Installation overview

Find information about how to install IBM Watson® Discovery Cartridge for IBM Cloud Pak® for Data.

IBM Cloud Pak for Data

Note: This information applies only to installed deployments.

Full installation instructions

- 4.8.x
- 4.7.x
- 4.6.x
- 4.5.x
- 4.0.x
- 2.2.0, 2.2.1

Federal Information Security Management Act (FISMA) support is available for Discovery for Cloud Pak for Data offerings purchased on or after August 30, 2019. IBM Watson® Discovery is FISMA High Ready.

Support matrix

You install IBM Cloud Pak for Data, and then install the Discovery service.

- The 4.6.2 release is the last supported release on Red Hat OpenShift Container Platform 4.8.
- The 4.5.3 release is the last supported release on Red Hat OpenShift Container Platform 4.6.29 or later.

Discovery version	IBM Cloud Pak for Data version	Red Hat OpenShift version
4.8.0	4.8.0	4.12
4.8.0	4.8.0	4.10
4.7.3	4.7.3	4.12
4.7.3	4.7.3	4.10
4.7.1	4.7.1	4.12
4.7.1	4.7.1	4.10
4.7.0	4.7.0	4.12
4.7.0	4.7.0	4.10
4.6.5	4.6.6	4.12
4.6.5	4.6.6	4.10
4.6.5	4.6.5	4.12
4.6.5	4.6.5	4.10
4.6.3	4.6.4	4.12
4.6.3	4.6.4	4.10
4.6.3	4.6.3	4.10

4.6.2	4.6.2	4.10
4.6.2	4.6.2	4.8
4.6.2	4.6.1	4.10
4.6.2	4.6.1	4.8
4.6.0	4.6.0	4.10
4.6.0	4.6.0	4.8
4.5.3	4.5.3	4.10
4.5.3	4.5.3	4.8
4.5.3	4.5.3	4.6.29 or later
4.5.1	4.5.1	4.10
4.5.1	4.5.1	4.8
4.5.1	4.5.1	4.6.29 or later
4.5.0	4.5.0	4.10
4.5.0	4.5.0	4.8
4.5.0	4.5.0	4.6.29 or later
4.0.9	4.0.9	4.8
4.0.9	4.0.9	4.6.29 or later
4.0.8	4.0.8	4.8
4.0.8	4.0.8	4.6.29 or later
4.0.7	4.0.7	4.8
4.0.7	4.0.7	4.6.29 or later
4.0.6	4.0.6	4.8
4.0.6	4.0.6	4.6.29 or later
4.0.5	4.0.5	4.8
4.0.5	4.0.5	4.6.29 or later
4.0.4	4.0.4	4.8
4.0.4	4.0.4	4.6.29 or later
4.0.3	4.0.3	4.8

4.0.3	4.0.3	4.6.29 or later
4.0.2	4.0.2	4.8
4.0.2	4.0.2	4.6
4.0.0	4.0.0	4.6
2.2.1	3.5.0	4.5, 4.6
2.2.1	3.5.0	3.11.188
2.2.1	3.0.1	4.5, 4.6
2.2.1	3.0.1	3.11.188
2.2.0	3.5.0	4.5
2.2.0	3.5.0	3.11.188
2.2.0	3.0.1	4.5
2.2.0	3.0.1	3.11
2.1.4	3.0.1	3.11.188
2.1.4	2.5	3.11
2.1.3	3.0.1	3.11.188
2.1.3	2.5	3.11

Support matrix

The `3.11.188` version more precisely means 3.11.188 or a later 3.11 version.

Upgrading the service

Learn how to upgrade the version of your installed service deployment.

IBM Cloud Pak for Data

Note: This information applies only to installed deployments.

Upgrade your deployment

The steps to follow to upgrade your Discovery service instance are described in the IBM Cloud Pak for Data documentation. The following in-place upgrades are supported:

- From a 4.6.x release or 4.7.x release to the latest 4.8 release. For more information, see Upgrading Watson Discovery.
- From one 4.7.x release to a later 4.7.y release. For more information, see Upgrading Watson Discovery to a newer 4.7 refresh.
- From a 4.6.x release to the latest 4.7 release. For more information, see Upgrading Watson Discovery.
- From a 4.0.x, 4.5.x, or earlier 4.6.x release to the latest 4.6 release. For more information, see Upgrading Watson Discovery.
- From a 4.0.x release or from one 4.5.x release to a later 4.5.y release. For more information, see Upgrading Watson Discovery.
- From one 4.0.x release to a later 4.0.y release. For more information, see Upgrading Watson Discovery to a newer 4.0 refresh.

You cannot do an in-place upgrade from releases prior to version 4.x. For information about how to move from a 2.x deployment to IBM Watson® Discovery 4.5, see Backing up and restoring data in IBM Cloud Pak for Data.

Troubleshooting IBM Watson® Discovery Cartridge for IBM Cloud Pak® for Data

deployments

Learn ways to troubleshoot and address issues that you might encounter while using the product.

IBM Cloud Pak for Data

> **Note:** This information applies only to instances of IBM Watson® Discovery that are installed on IBM Cloud Pak® for Data. For troubleshooting tips about adding data to both installed and managed deployments, see Troubleshooting ingestion.

The information in this topic suggests steps you can take to investigate issues that might occur. For information about known issues and their workarounds per version, see Known issues.

Minio pods enter a reboot loop during installation or upgrade

- **Error**: `Cannot find volume "export" to mount into container "ibm-minio"` is displayed during an installation or upgrade of Discovery. When you check the status of the Minio pods by using the command, `oc get pods -l release=wd-minio -o wide`, and then check the `Minio operator` logs by using the commands, `oc get pods -A | grep ibm-minio-operator`, and then `oc logs -n <namespace> ibm-minio-operator-XXXXX`, you see an error similar to the following one in the logs:

  ```
  ibm-minio/templates/minio-create-bucket-job.yaml failed: jobs.batch "wd-minio-discovery-create-bucket"
  already exists) and failed rollback: failed to replace object"
  ```

- **Cause**: A job that creates a storage bucket for Minio and then is deleted after it completes, is not being deleted properly.
- **Solution**: Complete the following steps to check whether an incomplete `create-bucket` job for Minio exists. If so, delete the incomplete job so that the job can be recreated and can then run successfully.

  1. Check for the Minio job by using the following command:

     ```
     oc get jobs | grep 'wd-minio-discovery-create-bucket'
     ```

  2. If an existing job is listed in the response, delete the job by using the following command:

     ```
     oc delete job $(oc get jobs -oname | grep 'wd-minio-discovery-create-bucket')
     ```

  3. Verify that all of the Minio pods start successfully by using the following command:

     ```
     oc get pods -l release=wd-minio -o wide
     ```

A `No space left on device` message is displayed in the log

If the a `wd-ibm-elasticsearch-es-server-client` pod restarts repeatedly and then reports the `Crashloopbackoff` state with the message `No space left on device` written to the log for the pod, then the issue might be a lack of memory on the pod. Follow the steps to troubleshoot out of memory issues or contact IBM Support.

A `java.lang.OutOfMemoryError: Java heap space` message is displayed in the log

When indexing a large set of documents that have multiple enrichments applied to them, the worker node can run out of space. To address the issue, first determine which pod is out of memory by completing the following steps:

If the document status cannot be promoted to `Processing`, check the status of the `inlet`, `outlet`, and `converter` pods.

1. Run the following command:

   ```
   $ oc get pod -l 'tenant=wd,run in (inlet,outlet,converter)'
   ```

2. If any of the pods are not showing a `Running` status, restart the failing pod by using the following command:

   ```
   $ oc delete pod <pod_name>
   ```

3. Otherwise, open the *Manage collections>{collection name}>Activity* page. Check the *Warnings and errors at a glance* section for the message, `OutOfMemory happened during conversion. Please reconsider size of documents.` If shown, use the following command to raise the memory size for the converter:

   ```
   $ oc patch wd wd --type=merge \
   --patch='{"spec":{"ingestion":{"converter":b{"maxHeapMemory":"10240m","resources":{"limits":{"memory":"10Gi"}}}}}}'
   ```

> **Note:** Adjust the value of `maxHeapMemory` and the container memory according to your cluster resources.

4. After the `converter` pod restarts successfully, click **Reprocess** from the Activity tab.

If documents are stuck in `Processing` status and cannot be promoted to the `Available` status for a collection, complete the following steps:

1. Check the status of Hadoop by using the following command:

   ```
   $ oc get pod -l 'tenant=wd,run in (hdp-rm,hdp-worker)'
   ```

   where `l` is a lowercase L for list.

2. If any of the pods are not showing a `Running` status, restart the failing pod by using the following command:

   ```
   $ oc delete pod <pod_name>
   ```

3. Check whether any of the Hadoop worker nodes has insufficient memory by using the following command to look for the `OOM when allocating` message:

   ```
   $ oc logs -l tenant=wd,run=hdp-worker -c logger --tail=-1 \
   | grep "OOM when allocating"
   ```

4. If a match is found, use the following command to patch the resource:

   ```
   $ oc patch wd wd --type=merge \
   --patch='{"spec":{"orchestrator":{"docproc":{"pythonAnalyzerMaxMemory":"8g"}}}}'
   ```

   > **Note:** The maximum allowed value for `pythonAnalyzerMaxMemory` is `12g`. The default value is `6g`. Increase the value gradually, such as in increments of 2g at a time according to your cluster resources.

5. Check whether any of the Hadoop worker nodes has insufficient memory by using the following command to look for the `OutOfMemoryError` message:

   ```
   $ oc logs -l tenant=wd,run=hdp-worker -c logger --tail=-1 \
   | grep "OutOfMemoryError"
   ```

6. If a match is found, check the current environment variable values and the Hadoop worker node memory resources by using following commands:

   - To check the `"DOCPROC_MAX_MEMORY"` variable in the orchestrator container:

     ```
     $ oc exec `oc get po -l run=orchestrator -o 'jsonpath={.items[0].metadata.name}'` env \
     | grep DOCPROC_MAX_MEMORY
     ```

   - To check the `"YARN_NODEMANAGER_RESOURCE_MEMORY_MB"` variable in the Hadoop worker node container:

     ```
     $ oc exec `oc get po -lrun=hdp-worker -o 'jsonpath={.items[0].metadata.name}'` \
     -c hdp-worker -- env | grep YARN_NODEMANAGER_RESOURCE_MEMORY_MB
     ```

   - To check the memory resource of the `hdp-worker` container:

     ```
     $ oc get po -l run=hdp-worker -o 'jsonpath=requests are \
     {.items[*].spec.containers[?(.name=="hdp-worker")].resources.requests.memory}, \
     limits are {.items[*].spec.containers[?(.name=="hdp-worker")].resources.limits.memory}'
     ```

7. Patch the environment variable resources gradually by using the following command:

   ```
   $ oc patch wd `oc get wd -o 'jsonpath={.items[0].metadata.name}'` \
   --type=merge --patch='{"spec":{"orchestrator":{"docproc":{"maxMemory":"4g"}}, \
   "hdp":{"worker":{"nm":{"memoryMB":12000}, "resources":{"limits":{"memory":"20Gi"}, \
   "requests":{"memory":"20Gi"}}}}}}'
   ```

   The default values for the resources are as follows:

   - `docproc.maxMemory`: 2g

Increase in increments of 2g at a time.

- `nm.memoryMB`: 10,240

  Start from 12,000 and increase in increments of 2,000 at a time.

- `memory requests/limits`: 13Gi/18Gi

  Increase in increments of 2Gi at a time.

8. Check whether the Hadoop pods restart successfully by using the following command:

```
$ oc get pods -l 'tenant=wd,run in (orchestrator,hdp-worker)'
```

9. Confirm that the new configurations were applied after you patched the cluster.
10. If the pod is not restarted, check whether the resource got updated by using the following command:

```
$ oc get wd wd -o yaml
```

11. Check the status of Elasticsearch by checking the client and data nodes separately.
12. On the client node, run the following command to check whether an out-of-memory exception occurred:

```
$ oc logs -l tenant=wd,ibm-es-data=False,ibm-es-master=False \
-c elasticsearch --tail=-1 | grep "OutOfMemoryError"
```

13. If an error message is found, excluding INFO messages, increase the memory resource by using the following command:

```
$ oc patch wd wd --type=merge \
--patch='{"spec":{"elasticsearch":{"clientNode":{"maxHeap":"896m","resources":{"limits":{"memory":"1792Mi"}}}}}}'
```

14. After you run this command, the Elasticsearch client pod is restarted about 20 minutes later. Monitor the "AGE" of the pod by using the following command:

```
$ oc get pod -l tenant=wd,ibm-es-data=False,ibm-es-master=False
```

15. After the pod is restarted successfully, check the new value of `ES_JAVA_OPTS` and the container memory limit by using the following command:

```
$ oc describe $(oc get po -l tenant=wd,ibm-es-data=False,ibm-es-master=False -o name)
```

16. On the data node, run the following command to check whether an out-of-memory exception occurred:

```
$ oc logs -l tenant=wd,ibm-es-data=True,ibm-es-master=False \
-c elasticsearch --tail=-1 | grep "OutOfMemoryError"
```

17. If an error message is found, excluding INFO messages, increase the memory resource by using the following command:

```
$ oc patch wd wd --type=merge \
--patch='{"spec":{"elasticsearch":{"dataNode":{"maxHeap":"6g","resources":{"limits":{"memory":"10Gi"},"requests":{"memory":"8Gi"}}}}}}'
```

18. After you run this command, the Elasticsearch client pod is restarted about 20 minutes later. Monitor the "AGE" of the pod by using the following command:

```
$ oc get pod -l tenant=wd,ibm-es-data=True,ibm-es-master=False
```

19. After the pod is restarted successfully, check the new value of `ES_JAVA_OPTS` and the container memory requests/limit by using the following command:

```
$ oc describe $(oc get po -l tenant=wd,ibm-es-data=True,ibm-es-master=False -o name)
```

Note: For both nodes (client and data), set the `resources.limits.memory` equal to `2 * maxHeap`.

If the pod cannot be restarted after 30 mins by applying the `oc patch` command, collect logs to share with IBM Support by using the following command:

```
$ oc logs -l control-plane=ibm-es-controller-manager --tail=-1
```

For both issues, where document status cannot be promoted to `Processing` and where documents are stuck in `Processing` status, if you are using Portworx storage, you can check whether the Elasticsearch disk is full.

1. Run the following command to check whether the Elasticsearch disk is full:

   ```
   $ oc logs -l tenant=wd,run=elastic,ibm-es-master=True \
   -c elasticsearch --tail=100000|grep 'disk watermark'
   ```

2. If the log shows a message such as `watermark exceeded on x-data-1`, it means the disk on the node that is specified is full and you need to increase the disk size by using the following command:

   ```
   $ oc patch pvc $(oc get pvc -l tenant=wd,run=elastic,ibm-es-data=True,ibm-es-master=False \
   -o jsonpath='{.items[N].metadata.name}') \
   -p '{"spec": {"resources": {"requests":{"storage": "60Gi"}}}}'
   ```

   where `N` denotes the data node number that was reported from the log.

   For example, if the log mentions `data-1` in the node name, then the command to use is:

   ```
   $ oc patch pvc $(oc get pvc -l tenant=wd,run=elastic,ibm-es-data=True,ibm-es-master=False \
   -o jsonpath='{.items[1].metadata.name}') -p '{"spec": {"resources": {"requests":{"storage": "60Gi"}}}}'
   ```

Setting the shard limit in Discovery for Cloud Pak for Data

In Discovery version 2.2.0, there is a limit to the number of shards that can stay open on a cluster. In development instances, the limit is 1,000 open shards, and in production instances, the limit is two data nodes, which is equal to 2,000 open shards, or 1,000 open shards per data node. After you reach either limit, you cannot create any more projects and collections on your cluster, and if you try to create a new project and collection, you receive an error message.

This limit is due to the fact that, when you install Discovery version 2.2.0, Elasticsearch version 7.8.0 automatically runs on your clusters. Because this version of Elasticsearch runs on your clusters, a new cluster stability configuration becomes available that limits the number of open shards to 1,000 for each Elasticsearch data node.

If you are unable to create new projects and collections and you receive errors, first check the status of your Elasticsearch cluster and the number of shards on that cluster. Consider increasing the number of data nodes on your cluster to support more shards. This method is optimal for maximizing performance. However, an increased number of nodes uses more memory. If the number of shards reaches the limit, you can also increase the limit in a data node. For more information about increasing the shard limit in a node, see Increasing the shard limit.

> **Note:** This limit of 1,000 shards does not apply to versions of Discovery that are earlier than 2.2.0.

Increasing the shard limit

1. Log in to your Discovery cluster.
2. Access your data node.
3. Enter the following command:

   ```
   $ oc exec -it $(oc get pod \
   -l app=elastic,ibm-es-data=True -o jsonpath='{.items[0].metadata.name}') -- bash
   ```

4. Enter the following command, replacing the `<>` and the content inside with your port number:

   ```
   $ curl -X POST http://localhost:<port_number>/_cluster/health?pretty
   ```

   If you do not know what your port number is, enter the following command to find it:

   ```
   $ oc get pod -l app=elastic,ibm-es-data=True -o json \
   | jq .items[].spec.containers[].ports[0].containerPort | head -n 1`
   ```

   The curl `POST` command returns a value for `active_primary_shards`. If you have one data node that has a value larger than 1,000 or if you have two data nodes that have a value larger than 2,000, you must increase the shard limit to create new projects and collections in your cluster.

   > **Important:** If you increase this limit, the cluster becomes less stable because it contains an increased number of shards.

5. Enter the following command to increase the number of shards, replacing `<port_number>` with your port number and `<total_shards_per_node>` and `<max_shards_per_node>` with the new shard limit that you want to assign to a node:

```
$ curl -X POST http://localhost:<port_number>/_cluster/settings \
-d '{"persistent": {"cluster.routing.allocation.total_shards_per_node":<total_shards_per_node>, \
"cluster.max_shards_per_node":<max_shards_per_node>} }' \
-XPUT -H 'Content-Type:application/json'
```

After you increase the shard limit, you can create more projects and collections on your cluster.

Clearing a lock state

IBM Cloud Pak for Data **Installed only**: When the `gateway` pod restarts, it runs a database validation plug-in that checks for changes and applies the latest change sets to the shared database. If the pod is restarted while this check is in process, the plug-in might remain in a lock state, preventing the service from starting. Manual database intervention might be needed to clear the lock.

If the Discovery API does not come online or if the `gateway-0` pod looks like it is in a constant crash loop, you can try checking the Liberty server logs for the API service located here: `/opt/ibm/wlp/output/wdapi/logs/messages.log`

The logs would indicate if Liquibase is failing and unable to run. If the system is locked, you might see something similar to the following:

```
$ [11/7/19 5:07:51:491 UTC] 0000002f liquibase.executor.jvm.JdbcExecutor I SELECT LOCKED FROM
public.databasechangeloglock WHERE ID=1
[11/7/19 5:07:51:593 UTC] 0000002f liquibase.lockservice.StandardLockService I Waiting for changelog lock....
[11/7/19 5:08:01:601 UTC] 0000002f liquibase.executor.jvm.JdbcExecutor I SELECT LOCKED FROM
public.databasechangeloglock WHERE ID=1
[11/7/19 5:08:02:091 UTC] 0000002f liquibase.lockservice.StandardLockService I Waiting for changelog lock....
[11/7/19 5:08:12:097 UTC] 0000002f liquibase.executor.jvm.JdbcExecutor I SELECT LOCKED FROM
public.databasechangeloglock WHERE ID=1
[11/7/19 5:08:12:197 UTC] 0000002f liquibase.lockservice.StandardLockService I Waiting for changelog lock....
[11/7/19 5:08:22:203 UTC] 0000002f liquibase.executor.jvm.JdbcExecutor I SELECT ID,LOCKED,LOCKGRANTED,LOCKEDBY
FROM public.databasechangeloglock WHERE ID=1
[11/7/19 5:08:22:613 UTC] 0000002f com.ibm.ws.logging.internal.impl.IncidentImpl I FFDC1015I: An FFDC Incident
has been created: "org.jboss.weld.exceptions.DeploymentException: WELD-000049: Unable to invoke public void
liquibase.integration.cdi.CDILiquibase.onStartup() on liquibase.integration.cdi.CDILiquibase@7f02a07
com.ibm.ws.container.service.state.internal.ApplicationStateManager 31" at ffdc\_19.11.07\_05.08.22.0.log
```

It is possible to manually unlock the plug-in. If you have Discovery 2.1.4 or earlier, enter the following command on the postgres database that the `gateway-0` pod is looking at:

```
$ psql dadmin
UPDATE DATABASECHANGELOGLOCK SET LOCKED=FALSE, LOCKGRANTED=null, LOCKEDBY=null where ID=1;
```

If you have Discovery 2.2.0 or later, enter the following command on the postgres database that the `gateway-0` pod is looking at:

```
$ oc exec -it wd-discovery-postgres-0 -- bash -c 'env PGPASSWORD="$PG_PASSWORD" psql
"postgresql://$PG_USER@$STKEEPER_CLUSTER_NAME-proxy-service:$STKEEPER_PG_PORT/dadmin" -c "UPDATE DATABASECHANGELOGLOCK SET
LOCKE
D=FALSE, LOCKGRANTED=null, LOCKEDBY=null where ID=1"'
```

If you can then restart the `gateway` pod, everything should resume normally.

Environment variable settings for Smart Document Understanding

There are two environment variables that need to be adjusted for Smart Document Understanding in IBM Watson® Discovery version 2.1.0. This was resolved in version 2.1.1, see 2.1.1 release, 24 Jan 2020.

```
$ SDU_PYTHON_REST_RESPONSE_TIMEOUT_MS
SDU_YOLO_TIMEOUT_SEC
```

Both of these should be set to their respective `hour` value. These values must be set in the `<release-name>-watson-discovery-hdp` ConfigMap.

The values should be:

`SDU_PYTHON_REST_RESPONSE_TIMEOUT_MS` should be set to `3600000`

`SDU_YOLO_TIMEOUT_SEC` should be set to `3600`

These values should be set when the software is installed or reinstalled.

Troubleshooting error messages

If you receive error messages that are related to timeouts and insufficient memory for your enrichments, you can enter the following commands to change timeout and memory settings to potentially resolve these error messages:

- Notice message: `<enrichment_name>: Document enrichment timed out`

  Suggested action: Increase the document processing timeout. Enter the following command to increase the default timeout from 10 to 20 minutes:

  ```
  $ oc patch wd wd --type=merge \
  --patch='{"spec": {"orchestrator": {"docproc": {"defaultTimeoutSeconds": 1200 } } } }'
  ```

- Notice message: `<enrichment_name>: Document enrichment failed due to lack of memory`

  Suggested action: Increase the orchestrator container memory limit. Enter the following command to increase the memory limit from 4 Gi to 6 Gi:

  ```
  $ oc patch wd wd --type=merge \
  --patch='{"spec": {"orchestrator": {"resources": {"limits": {"memory": "6Gi"} } } } }'
  ```

- Notice messsage: `Indexing request timed out`

  Suggested action: Increase the timeout for pushing documents to Elasticsearch. Enter the following command to increase the default timeout from 10 to 20 minutes:

  ```
  $ oc patch wd wd --type=merge \
  --patch='{"spec": {"shared": {"elastic": {"publishTimeoutSeconds": 1200 } } } }'
  ```

Known issues

Known issues are listed by the release in which they were identified.

- For the list of release notes, see Release notes.
- For troubleshooting information, see Troubleshooting.

IBM Cloud Pak for Data

Note: The known issues that are described in this topic apply to installed deployments only.

Important: Known issues are cumulative. Issues from previous releases persist in later releases unless otherwise noted.

4.8.x releases

See Limitations and known issues in Watson Discovery

4.7.x releases

See Limitations and known issues in Watson Discovery

4.6.x releases

See Limitations and known issues in Watson Discovery

4.5.x releases

See Limitations and known issues in Watson Discovery.

4.0.x releases

For more information about known issues, see the IBM Cloud Pak for Data documentation.

4.0.9, 25 May 2022

- Discovery generates a partial failure status message for the IBM Cloud Pak for Data Red Hat OpenShift APIs for Data Protection (OADP) backup and restore utility.
  - **Error**: When you check the status of the OADP backup utility after using it to backup a cluster where Discovery is installed, a `Phase:`

`PartiallyFailed` message is displayed. One or more Discovery components are included in the `Failed` list.

- **Cause**: Discovery cannot be backed up and restored by using the OADP backup and restore utility. When the Discovery service is present, and an administrator backs up an entire IBM Cloud Pak for Data instance, a status message is displayed that indicates a partial failure. This status is displayed because the persistent volume claims (PVCs) for Discovery are not backed up. However, the message does not impact the back up of the rest of the services.
- **Solution**: No action is required to resolve the status message. You can remove the persistent volume claims that are associated with the Discovery service separately. After using the scripts to back up your Discovery service data, you can follow the step that is documented in the uninstall instructions for the Discovery service to delete the PVCs. For more information about how to remove the PVC associated with Discovery, see Uninstalling the Discovery service.

4.0.8, 27 April 2022

- The wd-discovery-multi-tenant-migration job fails if anyone besides a system administrator performs the upgrade.
  - **Error**: When you upgrade with a user ID other than admin, the migration job fails.
  - **Cause**: The migration script assumes that the script is run by a user with the admin user ID.
  - **Solution**: Apply a patch that allows the migration to be successful. Complete the following steps:
    1. From the Cloud Pak for Data web client, get the user ID of the owner of the instance that you want to upgrade.
    2. Download the `wd-migration-uid-patch.zip` patch file from the Watson Developer Cloud GitHub repository.
    3. Extract the **wd-migration-uid-patch.yaml** file from the archive file, and then open it in a text editor.
    4. Replace the `<user_id>` variable with the user ID of the owner of the instance that you want to upgrade.
    5. Run the following command in a terminal that is logged in to the cluster:

       ```
       oc create -f wd-migration-uid-patch.yaml
       ```

    6. Delete the previous migration job by using the following command:

       ```
       oc delete job wd-discovery-multi-tenant-migration
       ```

    After the job is deleted, the migration job restarts and the migration resumes.

    The issue is fixed with the 4.0.9 release.

- Discovery generates a partial failure status message for the IBM Cloud Pak for Data OpenShift® APIs for Data Protection (OADP) backup and restore utility.
  - **Error**: When you check the status of the OADP backup utility after using it to backup a cluster where Discovery is installed, a `Phase: PartiallyFailed` message is displayed. One or more Discovery components are included in the `Failed` list.
  - **Cause**: Discovery cannot be backed up and restored by using the OADP backup and restore utility. When the Discovery service is present, and an administrator backs up an entire IBM Cloud Pak for Data instance, a status message is displayed that indicates a partial failure. This status is displayed because the persistent volume claims (PVCs) for Discovery are not backed up. However, the message does not impact the back up of the rest of the services.
  - **Solution**: No action is required to resolve the status message. You can remove the persistent volume claims that are associated with the Discovery service separately. After using the scripts to back up your Discovery service data, you can follow the step that is documented in the uninstall instructions for the Discovery service to delete the PVCs. For more information about how to remove the PVC associated with Discovery, see Uninstalling the Discovery service.

4.0.7, 30 March 2022

- Discovery generates an error in the IBM Cloud Pak for Data OpenShift® APIs for Data Protection (OADP) backup and restore utility.
  - **Error**: The utility does not complete successfully and the following message is written to the log: `preBackupViaConfigHookRule on backupconfig/watson-discovery in namespace cpd (status=error)`
  - **Cause**: Discovery cannot be backed up and restored by using the OADP backup and restore utility. When the Discovery service is present, and an administrator attempts to backup an entire IBM Cloud Pak for Data instance, Discovery prevents the utility from completing successfully.
  - **Solution**: Apply a patch that stops Discovery from preventing the utility from completing successfully.
    1. Download the `wd-aux-br-patch.zip` file from the Watson Developer Cloud Github repository.
    2. Extract the `wd-aux-br-patch.yaml` file from the ZIP file.
    3. Run the following command in a terminal that is logged in to the cluster:

       ```
       $ oc create -f wd-aux-br-patch.yaml
       ```

The issue is fixed with the 4.0.8 release. (You still cannot back up the Discovery service by using the OADP utility, but the OADP utility can back up other services when Discovery is installed.)

- `Deployed` status of resources fluctuates after the 4.0.7 upgrade is completed.
  - **Error**: When you check the status by submitting the `oc get WatsonDiscovery` command, the ready status of the resources toggles between showing `23/23` and `20/23` components as being ready for use.
  - **Cause**: The readiness state of the resources is not reported consistently after a migration.
  - **Solution**: Typically, the instance is ready for use despite the ready state instability. To manually refresh the status information, run the following commands in a terminal that is logged in to the cluster:

    ```
    $ oc proxy &
    curl -ksS -X PATCH -H "Accept: application/json, */*" -H "Content-Type: application/merge-patch+json"
    http://127.0.0.1:8001/apis/discovery.watson.ibm.com/v1/namespaces/<namespace>/watsondiscoveries/wd/statu
    s --data '{"status": null}'
    ```

  This issue is fixed with the 4.0.8 release.

- The wd-discovery-multi-tenant-migration job fails if anyone besides a system administrator performs the upgrade.
  - **Error**: When you upgrade with a user ID other than admin, the migration job fails.
  - **Cause**: The migration script assumes that the script is run by a user with the admin user ID.
  - **Solution**: Apply a patch that allows the migration to be successful. Complete the following steps:
    1. From the Cloud Pak for Data web client, get the user ID of the owner of the instance that you want to upgrade.
    2. Download the `wd-migration-uid-patch.zip` patch file from the Watson Developer Cloud GitHub repository.
    3. Extract the **wd-migration-uid-patch.yaml** file from the archive file, and then open it in a text editor.
    4. Replace the `<user_id>` variable with the user ID of the owner of the instance that you want to upgrade.
    5. Run the following command in a terminal that is logged in to the cluster:

       ```
       oc create -f wd-migration-uid-patch.yaml
       ```

    6. Delete the previous migration job by using the following command:

       ```
       oc delete job wd-discovery-multi-tenant-migration
       ```

  After the job is deleted, the migration job restarts and the migration resumes.

  The issue is fixed with the 4.0.9 release.

4.0.6, 1 March 2022

- Upgrade to 4.0.6 fails if no Discovery instance is provisioned in the existing cluster before you begin the upgrade process.
  - **Error**: The 4.0.6 upgrade process assumes that a Discovery instance is provisioned in the existing cluster. For example, if you are upgrading from 4.0.5 to 4.0.6, you must have an instance provisioned in the 4.0.5 cluster before you begin the migration.
  - **Cause**: The current code returns an error when no instance exists because it cannot find a document index to migrate.
  - **Solution**: Verify that an instance of Discovery has been provisioned in the existing IBM Cloud Pak for Data cluster before you start the upgrade to 4.0.6. If you tried to upgrade to 4.0.6, but no instances were provisioned and the migration failed, remove the existing installation and install 4.0.6 from scratch.

- `Deployed` status of resources fluctuates after the 4.0.6 upgrade is completed.
  - **Error**: When you check the status by submitting the `oc get WatsonDiscovery` command, the ready status of the resources toggles between showing `23/23` and `20/23` components as being ready for use.
  - **Cause**: The readiness state of the resources is not reported consistently after a migration.
  - **Solution**: Typically, the instance is ready for use despite the ready state instability. The ready state settles after approximately 5 hours. You can wait for the readiness state to consistently show `23/23` or you can manually refresh the status information by running the following commands in a terminal that is logged into the cluster:

    ```
    $ oc proxy &
    curl -ksS -X PATCH -H "Accept: application/json, */*" -H "Content-Type: application/merge-patch+json"
    http://127.0.0.1:8001/apis/discovery.watson.ibm.com/v1/namespaces/<namespace>/watsondiscoveries/wd/statu
    s --data '{"status": null}'
    ```

This issue is fixed with the 4.0.8 release.

- Discovery generates an error in the IBM Cloud Pak for Data OpenShift® APIs for Data Protection (OADP) backup and restore utility.
    - **Error**: The utility does not complete successfully and the following message is written to the log: `preBackupViaConfigHookRule on backupconfig/watson-discovery in namespace cpd (status=error)`
    - **Cause**: Discovery cannot be backed up and restored by using the OADP backup and restore utility. When the Discovery service is present, and an administrator attempts to backup an entire IBM Cloud Pak for Data instance, Discovery prevents the utility from completing successfully.
    - **Solution**: Apply a patch that stops Discovery from preventing the utility from completing successfully.
        1. Download the `wd-aux-br-patch.zip` file from the Watson Developer Cloud Github repository.
        2. Extract the `wd-aux-br-patch.yaml` file from the ZIP file.
        3. Run the following command in a terminal that is logged in to the cluster:

```
$ oc create -f wd-aux-br-patch.yaml
```

    This issue was fixed with the 4.0.8 release. (You still cannot back up the Discovery service by using the OADP utility, but the OADP utility can back up other services when Discovery is installed.)

- The wd-discovery-multi-tenant-migration job fails if anyone besides a system administrator performs the upgrade.
    - **Error**: When you upgrade with a user ID other than admin, the migration job fails.
    - **Cause**: The migration script assumes that the script is run by a user with the admin user ID.
    - **Solution**: Apply a patch that allows the migration to be successful. Complete the following steps:
        1. From the Cloud Pak for Data web client, get the user ID of the owner of the instance that you want to upgrade.
        2. Download the `wd-migration-uid-patch.zip` patch file from the Watson Developer Cloud GitHub repository.
        3. Extract the **wd-migration-uid-patch.yaml** file from the archive file, and then open it in a text editor.
        4. Replace the `<user_id>` variable with the user ID of the owner of the instance that you want to upgrade.
        5. Run the following command in a terminal that is logged in to the cluster:

```
oc create -f wd-migration-uid-patch.yaml
```

        6. Delete the previous migration job by using the following command:

```
oc delete job wd-discovery-multi-tenant-migration
```

    After the job is deleted, the migration job restarts and the migration resumes.

    The issue is fixed with the 4.0.9 release.

4.0.5, 26 January 2022

- Discovery generates an error in the IBM Cloud Pak for Data OpenShift® APIs for Data Protection (OADP) backup and restore utility.
    - **Error**: The utility does not complete successfully and the following message is written to the log: `preBackupViaConfigHookRule on backupconfig/watson-discovery in namespace cpd (status=error)`
    - **Cause**: Discovery cannot be backed up and restored by using the OADP backup and restore utility. When the Discovery service is present, and an administrator attempts to backup an entire IBM Cloud Pak for Data instance, Discovery prevents the utility from completing successfully.
    - **Solution**: Apply a patch that stops Discovery from preventing the utility from completing successfully.
        1. Download the `wd-aux-br-patch.zip` file from the Watson Developer Cloud Github repository.
        2. Extract the `wd-aux-br-patch.yaml` file from the ZIP file.
        3. Run the following command in a terminal that is logged in to the cluster:

```
$ oc create -f wd-aux-br-patch.yaml
```

    This issue was fixed with the 4.0.8 release. (You still cannot back up the Discovery service by using the OADP utility, but the OADP utility can back up other services when Discovery is installed.)

4.0.4, 20 December 2021

- Discovery generates an error in the IBM Cloud Pak for Data OpenShift® APIs for Data Protection (OADP) backup and restore utility.
    - **Error**: The utility does not complete successfully and the following message is written to the log: `preBackupViaConfigHookRule on backupconfig/watson-discovery in namespace cpd (status=error)`
    - **Cause**: Discovery cannot be backed up and restored by using the OADP backup and restore utility. When the Discovery service is present, and an administrator attempts to backup an entire IBM Cloud Pak for Data instance, Discovery prevents the utility from completing successfully.
    - **Solution**: Apply a patch that stops Discovery from preventing the utility from completing successfully.
        1. Download the `wd-aux-br-patch.zip` file from the Watson Developer Cloud Github repository.
        2. Extract the `wd-aux-br-patch.yaml` file from the ZIP file.
        3. Run the following command in a terminal that is logged in to the cluster:

```
$ oc create -f wd-aux-br-patch.yaml
```

This issue was fixed with the 4.0.8 release. (You still cannot back up the Discovery service by using the OADP utility, but the OADP utility can back up other services when Discovery is installed.)

4.0.3, 18 November 2021

- The guided tours are not available in this release.
- Discovery generates an error in the IBM Cloud Pak for Data OpenShift® APIs for Data Protection (OADP) backup and restore utility.
    - **Error**: The utility does not complete successfully and the following message is written to the log: `preBackupViaConfigHookRule on backupconfig/watson-discovery in namespace cpd (status=error)`
    - **Cause**: Discovery cannot be backed up and restored by using the OADP backup and restore utility. When the Discovery service is present, and an administrator attempts to backup an entire IBM Cloud Pak for Data instance, Discovery prevents the utility from completing successfully.
    - **Solution**: Apply a patch that stops Discovery from preventing the utility from completing successfully.
        1. Download the `wd-aux-br-patch.zip` file from the Watson Developer Cloud Github repository.
        2. Extract the `wd-aux-br-patch.yaml` file from the ZIP file.
        3. Run the following command in a terminal that is logged in to the cluster:

```
$ oc create -f wd-aux-br-patch.yaml
```

This issue was fixed with the 4.0.8 release. (You still cannot back up the Discovery service by using the OADP utility, but the OADP utility can back up other services when Discovery is installed.)

4.0.0, 13 July 2021

- Machine learning model enrichments that you apply by using the Analyze API can fail.
    - **Error**: `[WKSML_MODEL_NAME]: Enrichment of a document failed`
    - **Cause**: There is a known issue in Watson Knowledge Studio that can cause a timeout in enrichment processing.
    - **Solution**: When you use the Analyze API to apply a Watson Knowledge Studio model enrichment to a collection, keep the size of the input document under 50 KB.

2.2.1 issues that were fixed in subsequent releases

- [Fixed in version 4] If you add an IBM Watson® Knowledge Studio machine learning enrichment to a collection, the ingestion process might run very slowly but will eventually complete. If ingestion processes slowly, you might see the following error message in **Warnings and errors**:

```
$ [WKSML_MODEL_NAME]: Document analysis timed out
```

For additional timeout details, you can check your Knowledge Studio machine learning logs, which might look similar to the following:

```
{
    "message":  "Analysis failed due to:
   org.apache.uima.analysis_engine.AnalysisEngineProcessException
   at c.i.n.b.SIREAnnotator.process(_:454)
 ... ",
    "level":  "SEVERE",
```

```
}
```

Documents that time out during processing are indexed without Knowledge Studio enrichment results.

2.2.1, 26 February 2021

- Deployment timing issue:
  - **Error**: After installing patch 7, when you try to provision a service instance, a `404 Not Found` error is displayed. The following message might be logged for the `nginx` pods: `open() "/usr/local/openresty/nginx/html/watson/common/discovery/auth" failed (2: No such file or directory)`
  - **Solution**: Restart the `zen-watcher` pod.
- If you perform an air-gapped installation that pulls container images from an external container registry, you might experience the following issue:
  - **Error**: Some Discovery pods might report an `ImagePullBackoff` error.
  - **Cause**: The wrong image pull secret is being used.
  - **Solution**: Complete the following steps during the installation:
    - Start installing Watson Discovery.
    - After watson-discovery-operator module completes, check if a WatsonDiscovery custom resource is created by running the following command:

      ```
      $ oc get WatsonDiscovery wd
      ```

    - After the custom resource is created, run the following commands to point the correct image pull secret to pull images from the external registry:

      ```
      pull_secret=$(oc get secrets | grep 'docker-pull-.*-watson-discovery-registry-registry' | cut -d '' -f 1)
      cat << EOS > discovery-patch.yaml
      spec:
        shared:
          imagePullSecret: $pull_secret
      EOS
      oc patch wd wd --type=merge --patch "$(cat discovery-patch.yaml)"
      ```

    - If the `RabbitMQ` pods are still in ImagePullBackoff state, remove the RabbitMQ CR to enable the rabbitmq-operator to re-create the RabbitMQ clusters. You can use the following command:

      ```
      $ oc delete IbmRabbitmq wd-rabbitmq
      ```

- In IBM Watson® Discovery, the `Content Mining` project only supports one collection per project. If you create more than one `Content Mining` collection, you might experience errors. If you experience errors, delete additional `Content Mining` collections so that each `Content Mining` project has only one associated collection.
- If you are preparing your Discovery for Cloud Pak for Data clusters for an in-place upgrade of your instance from 2.2.0 to 2.2.1, occasionally, the `cpd-cli adm` command fails, showing the following error message: `Error from server (UnsupportedMediaType): error when applying patch`. If you receive this error message, enter `oc delete scc cpd-zensys-scc cpd-user-scc cpd-noperm-scc edb-operator-scc admin-discovery-scc` to delete the related resources, and re-enter the `cpd-cli adm` command.
- If you are upgrading your Discovery for Cloud Pak for Data instance from 2.2.0 to 2.2.1, occasionally, the `cpd-cli upgrade` command completes before rolling updates complete. For information about verifying that your upgrade completed successfully, see Verifying that your upgrade completed successfully.
- Model-train images are not updated after upgrading from Discovery 2.2.0 to 2.2.1. To work around this issue, delete the deployments that the model-train operator creates, and wait for the operator to recreate the deployments. Enter the following command to delete the deployments:

  ```
  $ oc delete deploy -l 'app.kubernetes.io/managed-by=ibm-modeltrain'
  ```

  After you run this command, the model-train operator creates new deployments.
- If you upgrade Discovery for Cloud Pak for Data from 2.2.0 to 2.2.1, you might receive the following error message:

  ```
  $ [ERROR] [2021-03-04 05:12:44-0657] Exiting due to error (Storage class is immutable. Module ibm-watson-gateway-operator x86_64 from Assembly portworx-shared-gp3 was installed with ibm-watson-gateway-operator x86_64, but new install/upgrade command is requesting portworx-db-gp3-sc. If you installed the assembly with a different storage class, please upgrade it individually.). Please check /ibm/cpd-cli-workspace/logs/CPD-2021-03-04T05-12-04.log for details
  ```

```
[ERROR] 2021-03-04T05:12:44.659615Z Execution error:  exit status 1
```

This error message is generated because the storage class that was used for installation is different than the one that was used during the upgrade. This discrepancy results from a different add-on installing the dependency operators because the storage class dependency operators of the different add-on were recorded as the ones that were used for installation. To work around this issue, you must upgrade the following subassemblies individually:

- Upgrade the Watson gateway operator:

```
$ ./cpd-cli upgrade \
--repo ./repo.yaml \
--assembly ibm-watson-gateway-operator \
--arch Cluster_architecture \
--namespace <Project> \
--transfer-image-to <Registry_location> \
--cluster-pull-prefix <Registry_from_cluster> \
--ask-pull-registry-credentials \
--ask-push-registry-credentials
```

- Upgrade Minio operator:

```
$ ./cpd-cli upgrade \
--repo ./repo.yaml \
--assembly ibm-minio-operator \
--namespace <Project> \
--transfer-image-to <Registry_location> \
--cluster-pull-prefix <Registry_from_cluster> \
--ask-pull-registry-credentials \
--ask-push-registry-credentials
```

- Upgrade RabbitMQ operator:

```
$ ./cpd-cli upgrade \
--repo ./repo.yaml \
--assembly ibm-rabbitmq-operator \
--namespace <Project> \
--transfer-image-to <Registry_location> \
--cluster-pull-prefix <Registry_from_cluster> \
--ask-pull-registry-credentials \
--ask-push-registry-credentials
```

- Upgrade etcd operator:

```
$ ./cpd-cli upgrade \
--repo ./repo.yaml \
--assembly ibm-etcd-operator \
--namespace <Project> \
--transfer-image-to <Registry_location> \
--cluster-pull-prefix <Registry_from_cluster> \
--ask-pull-registry-credentials \
--ask-push-registry-credentials
```

- Upgrade model train classic operator:

```
$ ./cpd-cli upgrade \
--repo ./repo.yaml \
--assembly modeltrain-classic \
--arch Cluster_architecture \
--namespace <Project> \
--transfer-image-to <Registry_location> \
--cluster-pull-prefix <Registry_from_cluster> \
--ask-pull-registry-credentials \
--ask-push-registry-credentials
```

- Upgrade Elasticsearch operator:

```
$ ./cpd-cli upgrade \
--repo ./repo.yaml \
--assembly ibm-cloudpakopen-elasticsearch-operator \
--namespace <Project> \
--transfer-image-to <Registry_location> \
--cluster-pull-prefix <Registry_from_cluster> \
--ask-pull-registry-credentials \
--ask-push-registry-credentials
```

where `<Project>` is the namespace where your Discovery for Cloud Pak for Data 2.2.0 instance is installed, where `<Registry_location>` is the location of the images that you pushed to the registry server, and where `<Registry_from_cluster>` is the location from which pods on the cluster can pull images.

- When you install on IBM Cloud Pak for Data 3.5, you might encounter the following issue:
  - **Error**: If you try to provision the Discovery service on a cluster where Planning Analytics is running, some of the Discovery pods don't start and installation fails. The logs for the pod show messages such as, `java.lang.NumberFormatException: For input string`.
  - **Cause**: An environment variable named `COUCHDB_PORT` is added to the Kubernetes cluster by the couchdb service that is installed with Planning Analytics. Discovery does not use couchdb, and therefore does not specify a value for this environment variable. However, some pods attempt to parse the variable, which results in the error.
  - **Solution**: Install patch cpd-watson-discovery-2.2.1-patch-1, which fixes this issue.

Also, see the issues in all previous releases.

2.2, 8 December 2020

- When a small CSV file (generally a CSV with 99 lines or fewer) is uploaded, the header and/or first row may not be ingested correctly. If this happens, in the tooling, navigate to the CSV Settings tab and update the settings. After reprocessing, navigate to the **Manage fields** tab and update the field types if needed.

- If you have set up your collections using a custom crawler built with the IBM Cloud Pak for Data custom connector, and then remove the custom crawler deployment, the Processing Settings page will not display the crawler configuration. This is because the underlying crawler is not available. To work around this issue, confirm that the custom crawler is deployed when there are collections using it.

- When using a IBM Cloud Pak for Data custom connector with Discovery for IBM Cloud Pak for Data 2.2, the script `scripts/manage_custom_crawler.sh` used to deploy and remove the deployment of the custom crawler fails. To work around this issue, replace line 37 `podname="gateway"` with `podname="wd-discovery-gateway"` in `scripts/manage_custom_crawler.sh`, and then rerun the deploy command.

- When you create a custom enrichment in the tooling, you must choose a field the enrichment should be applied to and click **Apply**. If no field is selected, then the **Apply and reprocess** button will be disabled for enrichments changes until the new enrichment has a field.

- If you apply the Contracts enrichment or the Understanding tables enrichment to a collection, you might receive the following error message when that collection is ingesting documents: `The number of nested documents has exceeded the allowed limit of [X].` Contact the IBM Support Center to adjust the limit.

- When text is enriched with a custom dictionary, the output of `entities.type` should be the full facet path for the Dictionary enrichment. However, in this release, the full facet path will not be displayed. To work around this, reprocess the collection. For example, if the facet path is `sample1.sample2`, it will look like this before reprocessing:

```
{
   "result" : {
      "enriched_text" : [
         {
            "entities" : [
               {
                  "text" : "capital",
                  "type" : "sample2",
                  ...
                  "model_name" : "Dictionary:.sample1.sample2"}
                  ...
```

And this after:

```
{
   "result" : {
      "enriched_text" : [
         {
            "entities" : [
               {
                  "text" : "capital",
                  "type" : "sample1.sample2",
                  ...
                  "model_name" : "Dictionary:.sample1.sample2"}
                  ...
```

- When a CSV file is uploaded with the converter settings set to `auto_detection=true`, the **CSV settings** tab in the tooling will display the incorrect settings. If you update the settings on the **CSV settings** tab, `auto_detection` will no longer be set to `true`.

- In Office documents ('.doc', '.docx', '.odf', '.xls', '.xlsx', '.ods', '.ppt', '.pptx', '.odp') converted using a Smart Document Understanding (SDU) custom model, the `publicationdate` may not display in `extracted_metadata` field in the JSON response. It will instead appear in the `html` field

of the JSON response. The `publicationdate` in the `html` field will be the date the document was ingested and not the document's original publication date.

- The Analyze API uses an in-memory cache to hold the enrichment models associated with the collection used to run the documents. If the collection contains many large enrichments or multiple of these collections are used at the same time, the cache may run out of memory. When this happens, the Analyze API returns null results (see example) and the stateless api rest proxy will display this message in its log: `RESOURCE_EXHAUSTED: stateless.Analysis/analyze: RESOURCE_EXHAUSTED`.

```
{
   "result":  null,
   "notices":  null
}
```

To work around this issue:

1. Review the enrichments used in the collection and remove those that are not necessary for your application. In particular, remove the *Part of Speech* enrichment.
2. Reduce the number of collections used concurrently with the Analyze API.
3. Increase the cache memory:
   - Increase the memory limit of `container model-runtime` in `deployment core-discovery-stateless-api-model-runtime` to `10` GB or more
   - Edit the environment variable `CAPACITY_MB` in `deployment core-discovery-stateless-api-model-runtime`, set it to `10240` or more

- If the model runtime container is restarted but the model mesh runtime container is not, the Analyze API can run into problems.
  - **Error**: The Analzye API call returns 500 error on a specific collection and the log contains the following entry:

```
"message":  "error occurred in analyzer
 java.lang.NullPointerException
 at c.i.e.a.a.s.r.ModelManager$2.analyze(ModelManager.java:112)
```

  - **Cause**: The model runtime container and model mesh runtime container are out of sync.
  - **Solution**: Delete the `wd-stateless-api-model-runtime` pods to restart both the model mesh and model runtime containers.

Also see the issues identified in all previous releases.

2.1.4, 2 September 2020:

- When configuring a Web crawl using FORM authentication, if you specify a URL without a trailing slash, for example: `https://webcrawlurl.com`, the web crawl will only crawl the login page. To work around this issue, add a trailing slash to the URL, for example: `https://webcrawlurl.com/`.
- The Guided Tours do not run on Firefox. For the list of other supported browsers, see Browser support.
- Ingesting documents into a collection that uses a custom Advanced Rules model built in Watson Knowledge Studio may fail if multiple extractors in the model internally use the same names for one or more output views.
- If you delete a large number of documents, then immediately ingest a large number of documents, it may take longer for all the documents to become available.
- The Classifier enrichment doesn't work when FIPS (Federal Information Processing Standards) is enabled.

Also see the issues identified in all previous releases.

2.1.4 issues that were fixed in subsequent releases

- [Fixed in version 2.2] In the deployed Content Mining application, if you include the tilde (~) symbol in a search query to enable fuzzy matching or include an asterisk (*) symbol to represent a wildcard, the search customizations function properly, but the matching string is not highlighted in the query result.
- [Fixed in version 2.2] A conversion error may occur when the **Include in index** field on the **Manage fields** tab in the tooling is changed. The document will not be indexed if this error occurs. To work around the issue:
  1. `oc edit sts core-discovery-converter`
  2. Edit between `containers` and `- name: INGESTION_POD_NAME` as follows:

```
containers:
   -  command:
      -  bash
      -  -c
      -  |
    FILE=/opt/ibm/wex/zing/bin/converter.sh &&
    sed -i "/choreo_2.11-9.1.1.jar/d" $FILE &&
    sed -i "/disco-doc-conversion-commons_2.11-1.0.4.jar/d" $FILE &&
```

```
sed -i "/jackson-module-scala_2.11-2.10.4.jar/d" $FILE &&
sed -i "/macro-compat_2.11-1.1.1.jar/d" $FILE &&
sed -i "/pureconfig-core_2.11-0.12.2.jar/d" $FILE &&
sed -i "/pureconfig-generic-base_2.11-0.12.2.jar/d" $FILE &&
sed -i "/pureconfig-generic_2.11-0.12.2.jar/d" $FILE &&
sed -i "/pureconfig-macros_2.11-0.12.2.jar/d" $FILE &&
sed -i "/pureconfig_2.11-0.12.2.jar/d" $FILE &&
sed -i "/scala-guice_2.11-4.1.1.jar/d" $FILE &&
sed -i "/scala-logging_2.11-3.7.2.jar/d" $FILE &&
sed -i "/scalactic_2.11-3.0.5.jar/d" $FILE &&
sed -i "/scalaj-http_2.11-2.3.0.jar/d" $FILE &&
sed -i "/service-commons_2.11-22.1.0.jar/d" $FILE &&
sed -i "/shapeless_2.11-2.3.3.jar/d" $FILE &&
/opt/ibm/wex/zing/bin/entrypoint.sh /opt/ibm/wex/zing/bin/controller.sh
  env:
  - name:  INGESTION_POD_NAME
```

Added lines from `- command:` to `/opt/ibm/wex/zing/bin/entrypoint.sh` `/opt/ibm/wex/zing/bin/controller.sh` and removed `-` before `env:`

3. Save the changes. It will restart the `converter` pod.

2.1.3, 19 June 2020:

- `Entity Subtypes` in IBM Watson® Knowledge Studio Machine Learning models are not supported in Discovery for Cloud Pak for Data 2.1.3 or later. For instructions on converting existing models, contact the Support center.
- You cannot upload CSV files that include a space in the file name (for example: `file 1.csv`) to a Content Mining project. Rename the file to work around the issue.
- When performing Project level relevancy training, if you have multiple collections, and two or more of those collections contains a duplicate `document_id`, then project level relevancy training will fail. Example of duplicate `document_ids`: `Collection A` contains a document with the id of `1234`, and `Collection B` also contains a document with the id of `1234`.
- Only the first facet using a field with the prefix `extracted_metadata` is saved correctly after creation. Others with that prefix will appear but after a screen refresh will be gone. This only happens once per project, so the workaround is to refresh and add the facet again.
- IBM Cloud Pak for Data During installation on IBM Cloud Pak® for Data 2.5.0.0, some Kubernetes Jobs may incorrectly report their status as `OOMKilled`, causing the install to timeout. To resolve this, once a Job returns `OOMKilled` verify the logs of the Pod associated with that Job. There should be no obvious error messages in the logs and the resources are reported in the logs as created. Manually verify these resources exist in the namespace and then delete the Job. This will cause the install to continue.
- Some documents may show two `html` fields when applying an enrichment. Both `html` fields shown are the same and operate as such.
- When creating a data source in Firefox, you may not see the entire list of options, including the **More processing settings** settings. To work around the issue, zoom out, increase the browser height, or use another supported browser.
- When customizing the display of search results, the changes made sometimes do not save after clicking the `Apply` button. To work around this issue, refresh the browser and try to make the changes again.
- When setting up a data source or web crawler for your collection, if you enter an incorrect configuration, then try to update it on the **Processing settings** page, the data source update or crawl may not start when you click the `Apply changes and reprocess` button. You can confirm this issue by opening the **Activity** page for your collection to see if processing has started. If you see that processing has not started for your data source, click the `Recrawl` button, then the `Apply changes and reprocess` button. If you see that processing has not started for your web crawl, click the `Stop` button, then the `Recrawl` button.
- IBM Cloud Pak for Data When running Helm tests on the `core` deployment using `helm test core`, the `core-discovery-api-post-install-test` will return a `FAILED` status. This is due to a bug within the `test` pod's image. The test result can be ignored as the failure is not related to anything within the deployment.
- By default, Optical Character Recognition (OCR) is set to `off` when you create any **Project type** with the tooling. However, if you create a Project using the API, OCR is set to `on`. To work around this issue, open the Tooling and change the **Project setting** to `off`.
- When Optical Character Recognition (OCR) is set to `on` for a Collection AND no trained Smart Document Understanding (SDU) model is applied, PNG, TIFF, and JPG files will not be processed for text recognition. Images embedded in PDF, Word, PowerPoint, and Excel documents will not be processed - only the non-image portion of these documents will be processed for text recognition. To work around this issue, import or train an SDU model and reprocess the collection. This will allow text to be extracted from the images.
- After you create a Search Skill in Watson Assistant and are directed to the Watson Discovery tooling, the screen is blank. This happens because the URL is missing the Discovery instance ID. To work around this issue:
  1. From the IBM Cloud Pak for Data web client menu, choose **My Instances**. For example: `https://mycluster.com/zen/#/myInstances`.
  2. Select the Discovery instance you are using and click **Launch Tool**.
  3. Once the tooling is loaded, the URL should have the following structure: `https://mycluster.com/discovery/core/instances/00000000-0000-0000-0001-597165341876/projects`

4. Copy the entire path, excluding `/projects`. For example: `https://mycluster.com/discovery/core/instances/00000000-0000-0000-0001-597165341876`
5. Go back to the browser tab that is displaying the blank Discovery screen. That URL structure will look like this: `https://mycluster.com/discovery/core/collections/new?redirect_uri=...`
6. Replace `https://mycluster.com/discovery/core` with the URL you copied previously, so the new URL should look like this: `https://mycluster.com/discovery/core/instances/00000000-0000-0000-0001-597165341876/collections/new?redirect_uri=...`
7. Press enter to open updated URL. You should now be on the Watson Discovery **Manage collections** page.

Also see the issues identified in all previous releases.

2.1.2, 31 March 2020

- When using passage retrieval with Korean, Polish, Japanese, Slovak or Chinese you may encounter much slower response times in this version. To resolve this, either disable passage retrieval, or upload a custom stopword list with words that are common in your documents (for example, prepositions and pronouns). See Defining stopwords for example stopword lists in several languages. Also see Stopwords ISO) on GitHub.

- [Update: fixed in version 2.1.3] In versions 2.1.2, 2.1.1, and 2.1.0, PNG, TIFF, and JPG individual image files are not scanned, and no text is extracted from those files. PNG, TIFF, and JPEG images embedded in PDF, Word, PowerPoint, and Excel files are also not scanned, and no text is extracted from those image files.

- Smart Document Understanding does not support `.doc`, `.docx`, `.odf`, `.xls`, `.xlsx`, `.ods`, `.ppt`, `.pptx`, and `.odp` conversion when FIPS (Federal Information Processing Standards) is enabled.

- In a Content Mining application, any document flags set will disappear if the index is rebuilt for that collection.

- Beginning with the 2.1.2 release, uploading and managing relevancy training data using the v1 APIs will not train a relevancy training model. The v1 APIs have been superseded by the Projects relevancy training v2 APIs. If your training data needs to be preserved, it can be listed using the v1 API, then added to a project with the v2 API.

- Multiple Regular expressions cannot be applied to a collection at the same time.

- IBM Cloud Pak for Data There were two small changes to the installation instructions README included with the download of IBM Watson® Discovery for IBM Cloud Pak® for Data. For the updated version of the README, see the Discovery Helm chart README.md.

  - A change to the description of the `--cluster-pull-prefix PREFIX` argument.
  - The language extension pack name has been updated from `ibm-watson-discovery-pack1-2.1.2.tar.xz.` to `ibm-wat-dis-pack1-prod-2.1.2.tar.xz`.

Also see the issues identified in all previous releases.

2.1.1, 24 January 2020

- When creating a dictionary, suggested dictionary terms are normalized to lowercase by default (for example, Watson Assistant will be normalized to watson assistant). To ensure matching on uppercase terms, they should be explicitly included as part of the `Other terms` list or as the `Base term`.
- When backing up and restoring data, training data does not restore successfully. If the documents in your collection were added by crawl using a connector or web crawl, your training data can be separately retrieved for backup from an existing project and uploaded to a new restored project. For more information, see List training queries and Create training queries) in the API reference.
- When crawling SharePoint Online or SharePoint OnPrem documents, JSON documents may not be indexed correctly and the `title` returned may be `errored`. This is because SharePoint web services use the `ows_FileRef` property to retrieve JSON files, which will return an error page. To fix this issue, contact your SharePoint Administrator and Microsoft Support.
- If you migrate a collection created in version 2.0.1 to either version 2.1.0 or 2.1.1, that collection will not have a **Project type** assigned and the collection will not be available to be queried. To assign a **Project type**, open the **Projects** page by selecting **My Projects**. Name your project and choose one of the Project types: `Document Retrieval`, `Conversational Search`, `Content Mining`, or `Custom`.

Also see the issues identified in all previous releases.

2.1.1 issues that were fixed in subsequent releases

- [Fixed in version 2.1.2] When installing Discovery for Cloud Pak for Data on OpenShift, the `ranker-rest` service might intermittently fail to startup, due to an incompatible jar in the `classpath`. To fix the issue:

  1. Open the `ranker-rest` editor with this command: `kubectl edit deployment {release-name}-{watson-discovery}-ranker-rest`
  2. In the editor, search for the `ranker-rest image` (for example: `{docker-registry}/{namespace}/discovery-ranker-rest-service:20200113-150050-2-d1527c2`)
  3. Add the following command below `{docker-registry}/{namespace}/discovery-ranker-rest-service:20200113-150050-2-d1527c2`:

     ```
     $ command: ["/tini"]
     args: ["-s", "-v", "--", "java", "-Dkaryon.ssl=true", "-Dkaryon.port=9081", "-Dkaryon.ssl.port=9090", "-Dkaryon.ssl.certificate=/opt/bluegoat/karyon/ssl/karyon-cert.pem", "-Dkaryon.ssl.privatekey=/opt/bluegoat/karyon/ssl/karyon-private-
     ```

```
key.pem", "-Djavax.net.ssl.trustStore=/opt/bluegoat/karyon/ssl/keystore.jks", "-
Djavax.net.ssl.keyStore=/opt/bluegoat/karyon/ssl/keystore.jks", "-Dlog4j.debug=false", "-Dlitelinks.threadcontexts=log4j_mdc", "-
Dwatson.ssl.truststore.path=/opt/bluegoat/karyon/ssl/litelinks-truststore.jks", "-Dwatson.ssl.truststore.password=watson15qa", "-
Dlitelinks.delay_client_close=false", "-Drxnetty.http.maxcontentlength=314572800", "-cp", "lib/logback-classic-1.2.3.jar:*:lib/*",
"com.ibm.watson.raas.rest.Runner"]
```

2.1.0, 27 November 2019

- When you apply an enrichment to a collection, the enrichment language must match the collection language, or it will fail. The tooling displays all the collections, regardless of language.
- On the Manage Fields tab, you can edit system-generated fields. The following fields should not be edited by changing the field type or turning off indexing: `document_id`, `extracted_metadata`, `metadata`.
- When you delete a Collection and select the option `Don't delete underlying data`, any incomplete document ingestion crawls will continue running in the background, which will impact the new crawl start times, until the existing crawls are completed.
- IBM Cloud Pak for Data Discovery can fail to start up correctly due to components getting into a lock state. Manual database intervention may be needed to clear the lock. For more information on identifying and resolving this issue, see Clearing a lock state.
- If you upload a document with the Upload Data function, delete that document, and then try to upload either the same document or another document with the same document ID,the upload will fail and the message `Error during creating a document` will be displayed.
- Documents that produce an `html` field when processed can not be used with relevancy training. html is produced for documents processed with Smart Document Understanding or Content Intelligence. The `html` field must be removed before relevancy training can complete successfully.
- If the *Part of Speech* enrichment is not turned on: Dynamic facets will not be created, Dictionary suggestions cannot be used, Content Miner "extracted facets" will not generate.
- [Update: fixed in version 2.1.1] Discovery for Content Intelligence and Table Understanding enrichments are configured out of the box to be applied on a field named `html`. When a user uploads a JSON document without a root-level field named `html`, these enrichments will not yield results in the index. To run the enrichments on this kind of JSON documents, users must re-configure the enrichments to run on an existing field (or fields) in the JSON document.
- When viewing the Content Miner deploy page, sometimes the full application URL is not displayed for copying. To fix, refresh the page.
- [Update: fixed in version 2.1.2] Deprovisioning a IBM Watson® Discovery for IBM Cloud Pak® for Data Instance will not delete the underlying data. Delete the collections and documents manually.
- [Update: fixed in version 2.1.3] On the Improvement tools panel, the enrichment `Sentiment of phrases` is listed, but is not currently available.
- In Content Mining projects, the `dates` fields may not be parsed properly for display in facets.
- The Dynamic facets toggle should not appear in Content Mining projects.
- A minimum of 50-100 documents should be ingested to see valid dynamic facets generated.
- If you click **Stop** to stop a crawler and the converter processes slowly or has errors, you might see a status of the crawler running.
- The total size limit of all non-HTML fields in uploaded and crawled documents is 1MB, which is equivalent to 1,048,576 bytes, and the total size limit of all HTML fields in these documents is 5MB. If you exceed either limit, you receive an error message stating `The document has fields/HTML fields that exceed the 1 MB/5 MB limit.`, and the document is not ingested. For assistance on increasing either size limit, contact the IBM Support Center.

Also see the issues identified in all previous releases.

2.0.1, 30 August 2019

- After you create a Machine Learning enrichment using a IBM Watson® Knowledge Studio model, two identically named enrichments may display on the **Enrich fields** page. This will not affect the enrichments, but it is best to use only one of them to select and apply the enrichment to one or more fields.
- If a web crawl appears to be stuck processing at a fixed number of documents, and the message displayed on the **Logs** page is `The ingestion job <jobid> is terminated incorrectly`, contact IBM support for assistance restarting the crawl.
- If one or more of your collections is trained, the training data from one of those collection may display on the **Train** page of an untrained collection. Refresh the page to clear that training data.
- The following types of documents will not be processed if they do not have the proper file extension: .docx, .pptx, .xlsx.

Also see the issues identified in the previous release.

2.0.1 issues that were fixed in subsequent releases

- [Fixed in version 2.1.2] When you upload documents to a collection with existing documents, a `Documents uploaded!` message displays on the **Activity** page, but no further processing status displays until the number of documents increases.

General Availability (GA) release, 28 June 2019

- If you are working in the Discovery for Cloud Pak for Data tooling, and your IBM Cloud Pak® for Data session expires, you will receive a blank page. To return to the tooling, refresh the browser and log back in.
- All JSON files ingested into Discovery should include the .json file extension.
- When querying on the `collection_id` of a trained collection, the `training_status.notices` value may occasionally display as `0` instead of the correct value.
- Not all query limitations are enforced in this release. See query limitations for the complete list of banned fields.
- In JSON source documents, you should not duplicate the following system-generated fields: `document_id`, `parent_document_id`, `filename`, and `title`. This will cause the duplicate fields to nest within arrays and break certain features, such as ranker training.
- Do not include a root-level `metadata` property in your JSON documents. If you upload a JSON document that already contains a root-level `metadata` property, then the `metadata` property of the indexed document will be converted to an array in the index.

- Do not use metadata for column names in your CSV files. If you upload a CSV file that uses metadata for the column names in the header, then the `metadata` property of the indexed document will be converted to an array in the index.
- CSV files must use commas (`,`) or semicolons (`;`) as delimiters; other delimiters are not supported. If your CSV file includes values containing either commas or semicolons, you should surround those values in double quotation marks so they are not separated. If header rows are present, the values within them are processed in the same manner as values in all other rows. The last row of CSV files will not be processed if not followed by a CRLF (carriage return).
- Currently, unique collection names are not enforced. Using duplicate collection names is not recommended and should be avoided

Advanced development

Configuring a Cloud Pak for Data custom connector

Building a Cloud Pak for Data custom connector

Discovery provides connectors to many popular data sources, as described in Configuring Cloud Pak for Data data sources. If you need to connect to a different data source, you can write and deploy a *custom connector*.

IBM Cloud Pak for Data **IBM Cloud Pak for Data only**

> **Note:** This information applies only to installed deployments.

> **Note:** Any custom code that is used with IBM Watson® Discovery is the responsibility of the developer and is not covered by IBM support.

Example code and configuration files for a basic custom connector are included.

Related topics:

- Developing custom Cloud Pak for Data connector code
- Assembling, compiling, and packaging a custom Cloud Pak for Data connector
- Installing and uninstalling a custom Cloud Pak for Data connector
- Using a custom Cloud Pak for Data connector from the Discovery user interface

Custom connector requirements

A custom connector is a component that uses the SDK and crawler framework that is documented here to connect to and crawl a specific data source. Custom connectors have the same general requirements as provided connectors. For more information, see Data source requirements.

Before you implement a custom connector, you need to know the following information about the data source:

- The data source's network location (server name or address, including port, or URL, including port)
- The data source's authentication method and security credentials
- The path or paths on the data source that the connector needs to crawl
- The connection method or protocol that the data source supports

Designing a custom connector

A custom connector needs the following capabilities:

- Configuring a crawler.
  - Configuring all settings that are required to connect to the data source.
  - Discovering a *crawl space* on the data source. At least one crawl space is required.
- Crawling documents.
  - Crawling the documents on each data set.
  - Adding Access Control List (ACL) information to each document.
- Retrieving ACL information for the username that authenticates to the data source.

These capabilities can be implemented by using the interfaces and methods that are described in Developing custom connector code.

Custom connector limitations

Observe the following notes and warnings when you implement a custom connector.

- Custom connectors do *not* support the following features:
  - Synchronization settings
  - Filtering documents based on user access at query time. (At crawl and index time, only documents that the current user has the right to access are returned.)
  - The `required` and `hidden` validation settings. They are ignored when the connector is displayed in Discovery
  - The use of `<condition />` tags in the definition file. These tags are currently ignored.
- When you use the example connector code in the current release, Discovery does not collapse and group authentication settings for the custom connector's properties. For example, even when the `{connector_name}_DATASOURCE_SETTINGS_USE_KEY_LABEL` toggle is set to `Off`, the user interface shows the fields for `{connector_name}_DATASOURCE_SETTINGS_KEY_LABEL` and `{connector_name}_DATATSOURCE_SETTINGS_PASSPHRASE_LABEL`.

- The `list` parameter type is not supported.
- If a custom connector fails to connect to its source for any reason, it issues a generic error message such as `Failed to create connector` or `Timed out`, or a `500` HTTP error. Specific failure information is not currently provided.

  See the Release notes for more possible issues.

Developing custom Cloud Pak for Data connector code

The custom connector example includes a Java package named `com.ibm.es.ama.custom.crawler`. The package includes the following Java interfaces that you can use when you write your own custom connector.

IBM Cloud Pak for Data **IBM Cloud Pak for Data only**

> **Note:** This information applies only to installed deployments.

Interfaces and Javadoc

The interfaces that are listed in this document are available in the JAR package file that is included with the custom connector compressed file. After you download and expand the `custom-crawler-docs.zip` file as described in Downloading the `custom-crawler-docs.zip` file in Discovery 2.2.1 and later and Downloading the `custom-crawler-docs.zip` file in Discovery 2.2.0 and earlier, the interface JAR file is available as `wexlib/ama-zing-custom-crawler-{version_numbers}.jar` from the root level of the expanded compressed file. Javadoc for the JAR file is available as `wexlib/ama-zing-custom-crawler-{version_numbers}-javadoc.jar` at the same level.

Initialization interface

`CustomCrawler`

Use the `com.ibm.es.ama.custom.crawler.CustomCrawler` interface to start or stop a custom crawler or to crawl documents from a path. The interface has the following methods.

Method	Description
`init`	Start a custom crawler
`term`	Stop a custom crawler
`crawl`	Crawl documents from a specified path

CustomCrawler methods

Configuration interfaces

`CustomCrawlerConfiguration`

Use the `com.ibm.es.ama.custom.crawler.CustomCrawlerConfiguration` interface to validate the configuration and to discover available crawl spaces on the data source. The interface has the following methods.

Method	Description
`validate`	Validate configuration
`getFieldsFor`	List known fields and their types
`discoverySubspaces`	Discover crawl spaces on the data source

CustomCrawlerConfiguration methods

`ConfigProvider`

Use the `com.ibm.es.ama.custom.crawler.CustomCrawlerConfiguration.ConfigProvider` interface to map the settings of the data source and to list the crawl-space settings on the data source. The interface has the following methods:

Method	Description

`get`	Get a map of the settings in a section
`getCrawlspaceSettings`	Get a list of crawl-space settings

ConfigProvider methods

`SubspaceConsumer`

Use the `com.ibm.es.ama.custom.crawler.CustomCrawlerConfiguration.SubspaceConsumer` interface to add a path to a crawl space. The interface has the following method:

Method	Description
`add`	Add a path to the crawl space

SubspaceConsumer methods

Crawler interface

`RecordKeeper`

Use the `com.ibm.es.ama.custom.crawler.CustomCrawler.RecordKeeper` interface to keep records of crawls and to publish crawled documents. The interface has the following methods:

Method	Description
`canContinue`	Boolean that lists whether the crawler can continue. The custom crawler must poll this value periodically and terminate if it returns `false`.
`check`	Get metadata fields from the last crawled document
`upsert`	Publish a document for further processing
`delete`	Delete a document

RecordKeeper methods

Security interface

`CustomCrawlerSecurityHandler`

Use the `com.ibm.es.ama.custom.crawler.CustomCrawlerSecurityHandler` interface to implement security for your custom crawler. The interface has the following methods:

Method	Description
`term`	Terminate a security handler
`getUserAndGroups`	Get the ACLs of a given user

CustomCrawlerSecurityHandler methods

⚠ **Important:** When the `getUserAndGroups` logic of a connector is updated, it can take up to 10 minutes after the connector is redeployed for the change to take effect.

Custom connector example

The example connector is a Secure File Transfer Protocol (SFTP) connector that crawls files that are located on an SFTP server.

The example connector includes three components:

- Java source code for the connector
- An XML definition file that defines the parameters that the connector uses to connect to and crawl the data source
- A properties file that defines optional behaviors for the connector

Requirements

The Java source code for the example connector has the following dependencies:

- Java SE Development Kit (JDK) 1.8 or higher.
- The `custom-crawler-docs.zip` file from an installed Discovery instance as described at Downloading the `custom-crawler-docs.zip` file in Discovery 2.2.1 and later and Downloading the `custom-crawler-docs.zip` file in Discovery 2.2.0 and earlier.
- The JSch Java package, as described Downloaded JSch. You can download the package in ZIP format or JAR format.

Downloading the `custom-crawler-docs.zip` file in Discovery 2.2.1 and later

In Discovery version 2.2.1 and later, perform the following steps to download the `custom-crawler-docs.zip` file to your local machine. You need root access to an installed Discovery instance:

1. Log in to your Discovery cluster.
2. Enter the following command to obtain your `crawler` pod name:

   ```
   $ oc get pods | grep crawler
   ```

   You might see output that looks like this:

   ```
   wd-discovery-crawler-57985fc5cf-rxk89     1/1     Running     0          85m
   ```

3. Enter the following command to obtain the `custom-crawler-docs.zip` file, replacing `{crawler-pod-name}` with the `crawler` pod name that you obtained in step 2:

   ```
   $ oc exec {crawler-pod-name} -- ls -l /opt/ibm/wex/zing/resources/ \
   | grep custom-crawler-docs
   ```

   You might see output that is similar to the following:

   ```
   -rw-r--r--. 1 dadmin dadmin 59451 Jan 19 03:50 custom-crawler-docs-${build-version}.zip
   ```

4. Enter the following command to copy the `custom-crawler-docs.zip` file to the host server, replacing `{build-version}` with the build version number in step 3:

   ```
   $ oc cp {crawler-pod-name}:/opt/ibm/wex/zing/resources/custom-crawler-docs-${build-version}.zip custom-crawler-
   docs.zip
   ```

5. Enter the following command to expand the `custom-crawler-docs.zip` file:

   ```
   $ unzip custom-crawler-docs.zip -d custom-crawler-docs-primary
   ```

   Note: If necessary, copy the `custom-crawler-docs.zip` file to the development server.

   Note: If your local machine does not have the `unzip` utility, try using the `gunzip` command instead, or see the documentation of the operating system of your local machine for other alternatives to expand compressed files.

   Tip: If you are using a version of Discovery that is earlier than 2.1.2 and you want to access the `custom-crawler-docs.zip` file, enter the following command: `scp root@{instance_name}:/root/bob/sdk/custom-crawler-docs.zip {local_directory}`.

For information about downloading the `custom-crawler-docs.zip` file to Discovery 2.2.0 and earlier, see Downloading the custom-crawler-docs.zip file to Discovery 2.2.0 and earlier.

Downloading the `custom-crawler-docs.zip` file to Discovery 2.2.0 and earlier

In Discovery version 2.2.0 and earlier, perform the following steps to download the `custom-crawler-docs.zip` file to your local machine. You need root access to an installed Discovery instance:

1. Obtain the entitlement key by navigating to your container software library.
2. Enter the following command to log in to the Docker registry where your Discovery images are available. Include your entitlement key in the following command:

```
$ docker login cp.icr.io -u cp -p {entitlement_key}
```

3. Enter the following command to pull the `custom-crawler-sdk` image:

```
$ docker pull cp.icr.io/cp/watson-discovery/custom-crawler-sdk:2.1.3
```

4. Enter the following command to run the `custom-crawler-sdk` image:

```
$ docker run cp.icr.io/cp/watson-discovery/custom-crawler-sdk:2.1.3
```

5. Enter the following command to copy `custom-crawler-docs.zip` from the container where the image is running:

```
$ docker cp {container_name}:/crawler/custom-crawler-docs.zip .
```

   To find the image, enter `docker ps -a | grep custom-crawler-sdk`.

6. Expand the `custom-crawler-docs.zip` file:

```
$ cd {local_directory}
```

   where `{local_directory}` is the directory on your local machine to which you downloaded the `custom-crawler-docs.zip` file.

```
$ unzip custom-crawler-docs.zip
```

> **Note:** If your local machine does not have the `unzip` utility, try using the `gunzip` command instead, or see the documentation of the operating system of your local machine for other alternatives to expand compressed files.

> **Tip:** If you are using a version of Discovery that is earlier than 2.1.2 and you want to access the `custom-crawler-docs.zip` file, enter the following command: `scp root@{instance_name}:/root/bob/sdk/custom-crawler-docs.zip {local_directory}`.

For information about downloading the `custom-crawler-docs.zip` file on Discovery version 2.2.1 and later, see Downloading the `custom-crawler-docs.zip` file in Discovery 2.2.1 and later.

Understanding the `custom-crawler-docs.zip` file

The `custom-crawler-docs.zip` file expands into a directory named `custom-crawler-docs-primary` that includes the following contents:

```
custom-crawler-docs-primary
├── README.md
├── build.gradle
├── config
│   ├── README.md
│   ├── messages.properties
│   └── template.xml
├── scripts
│   └── manage_custom_crawler.sh
├── settings.gradle
├── src
│   └── main
│       └── java
│           └── com
│               └── ibm
│                   └── es
│                       └── ama
│                           └── custom
│                               └── crawler
│                                   └── sample
│                                       └── sftp
│                                           ├── SftpCrawler.java
│                                           └── SftpSecurityHandler.java
└── wexlib
    ├── META-INF
    │   └── MANIFEST.MF
    ├── README.md
    ├── ama-zing-custom-crawler-{version_numbers}-javadoc.jar
    └── ama-zing-custom-crawler-{version_numbers}.jar

15 directories, 12 files
```

Downloading JSch

JSch is a Java implementation of the Secure Shell protocol version 2 (SSH2) protocol and, by extension, `sftp`. It is derived from the Java Cryptography Extension (JCE). You can find specifications for SSH2 at www.openssh.com/specs.html.

The current version of JSch is 0.1.55 and is supported by the example connector.

Download JSch to your development directory (`{local_directory}`). You can download the package in ZIP format or JAR format. If you download the package in .zip format, extract it as described in the previous section.

Files for the example connector

The example custom connector includes three files that get built together:

- Java source files that are named `SftpCrawler.java` and `SftpSecurityHandler.java`
- An XML definitions file named `template.xml`
- A properties file named `message.properties`

You can locate and examine these files by referencing the directory tree listing in Understanding the `custom-crawler-docs.zip` file.

For more information

For detailed documentation of all of the interfaces and methods that are available in the `com.ibm.es.ama.custom.crawler` package, see the Javadoc, which is available as indicated in Interfaces and Javadoc.

Assembling and compiling a custom Cloud Pak for Data connector

You package a number of component files together to create a custom connector.

IBM Cloud Pak for Data **IBM Cloud Pak for Data only**

Note: This information applies only to installed deployments.

Custom connector components

A custom connector package is a compressed file that contains the following components:

Path	Description
`config/template.xml`	A configuration template
`config/messages.properties`	A properties file for UI messages
`lib/*.jar`	JAR files required by the custom connector, not including the connector code that you write

Connector components

Configuration template

The configuration template is an XML file that is divided into sections. Each section contains related settings. The XML snippets are taken from the example `template.xml` file whose location is listed in Understanding the `custom-crawler-docs.zip` file.

Declaration settings

Declared settings are represented by the `<declare />` element. The element has the following attributes:

Attribute name	Description
`type`	Data type; one of `string`, `long`, `boolean`, `list` of strings, or `enum`
`name`	The name of the setting
`initial-value`	The initial value of the setting
`enum-value`	A list of `enum` values separated by vertical bars (`|`)

`required`	Indicates that the setting is required
`hidden`	Indicates whether to hide the setting from the UI. Specify a value of `true` to hide the setting.

Declare element attributes

Note: In the current release, the `required` and `hidden` attributes are not applied in the Discovery product user interface.

Declaration setting examples

To declare an `enum` type, use code similar to the following snippet:

```
<declare type="enum" name="type" enum-values="PROXY|BASIC|NTLM" initial-value="BASIC"/>
```

To declare a hidden `string` with an initial value, use code similar to the following snippet:

```
<declare type="string" name="custom_config_class" hidden="true" initial-value="com.example.ExampleCrawlerConfig" />
```

To declare a required `long`, use code similar to the following snippet:

```
<declare type="long" name="port" required="required" initial-value="22"/>
```

Conditional settings

Conditional settings are represented by the `<condition />` element. A conditional setting is displayed only if the condition is satisfied. The element has the following attributes:

Attribute name	Description
`name`	The name of the setting
`enable`	Enable the setting if the value of the `name` attribute equals the value of the `enable` attribute
`in`	Enable the setting if the value of the `name` attribute is included in a specified list of values

Condition element attributes

Note: In the current release, conditional settings are not applied in the Discovery product user interface.

Conditional setting examples

To enable a section by using a `boolean` condition, use code similar to the following snippet:

```
<declare type="boolean" name="use_key" initial-value="true" />
<!-- Enable setting only if use_key is true -->
<condition name="use_key" enabled="true">
   <declare type="string" name="key" hidden="false" />
</condition>
```

To enable a section by using an `enum` condition, use code similar to the following snippet:

```
<declare type="enum" name="type" enum-values="PROXY|BASIC|NTLM" initial-value="BASIC"/>
<!-- Enable setting for BASIC, NTLM or PROXY -->
<condition name="type" in="BASIC|NTLM|PROXY">
</condition>
<!-- Enable setting for PROXY -->
<condition name="type" in="PROXY">
```

Template sections

Each section includes one `<declare />` element for each of its settings.

XPath expression	Description

`/function/@name`	The name (type) of the crawler. Not a display name for the UI. Cannot contain spaces.
`/function/prototype/proto-section`	A section of the configuration.

Template sections

Section: `general_settings`

The XPath expression is `/function/prototype/proto-section[@section="general_settings"]`. It includes common settings for all crawlers, including the following settings:

```
<declare type="string" name="crawler_name" />
<declare type="string" name="description" />
<declare type="long" name="fetch_interval" initial-value="0" />
<declare type="long" name="number_of_max_threads" initial-value="10" />
<declare type="long" name="number_of_max_documents" initial-value="2000000000" />
<declare type="long" name="max_page_length" initial-value="32768" />
```

The custom crawler is initialized with the following settings in the `general_settings` section. For information about the interfaces, see Developing custom connector code.

Name	Value
`custom_config_class`	The name of a class that implements the `com.ibm.es.ama.custom.crawler.CustomCrawlerConfiguration` interface
`custom_crawler_class`	The name of a class that implements the `com.ibm.es.ama.custom.crawler.CustomCrawler` interface
`custom_security_class`	The name of a class that implements the `com.ibm.es.ama.custom.crawler.CustomCrawlerSecurityHandler` interface
`document_level_security_supported`	Specifies whether document-level security is enabled (`true`) or disabled (`false`)

General settings section defaults

To specify the interfaces, use code similar to the following snippet:

```
<!-- Configuration class -->
   <declare type="string" name="custom_config_class" hidden="true" initial-value="com.ibm.es.ama.custom.crwler.sample.sftp.SftpCrawler" />
<!-- Crawler class -->
   <declare type="string" name="custom_crawler_class" hidden="true" initial-value="com.ibm.es.ama.custom.crwler.sample.sftp.SftpCrawler" />
<!-- Document level security class -->
   <declare type="string" name="custom_security_class" hidden="true" initial-value="com.ibm.es.ama.custom.crwler.sample.sftp.SftpCrawler" />
<!-- Document level security is enabled or not -->
   <declare type="boolean" name="document_level_security_supported" initial-value="true" hidden="true"/>
```

Note: If you built a custom connector with an SDK package that was bundled with version 2.2.1 or earlier, `document_level_security_supported` must be disabled (set to `false`). Document-level security is not supported in 2.2.1 and earlier releases. However, the **Enable Document Level Security** option is displayed in Discovery even when document-level security is not supported. Do not select this option when you create a new collection.

To hide the **Enable Document Level Security** option from Discovery if the custom connector was built with an SDK package that was bundled with version 2.2.1 or earlier, complete the following steps:

1. Change the `document_level_security_supported` parameter in the `config/template.xml` file to read as follows:

   ```
   <declare type="boolean" name="document_level_security_supported" hidden="true" initial-value="false"/>
   ```

2. Rebuild the connector package, and then upload it again.

Section: `datasource_settings`

The XPath expression is `/function/prototype/proto-section[@section="datasource_settings"]`. It includes settings specific to the data source.

```
<!-- Data source settings change on each server -->
   <proto-section section="datasource_settings">
      <!-- Sample: SFTP server settings -->
         <declare type="string" name="host" required="required" initial-value="localhost"/>
         <declare type="long" name="port" required="required" initial-value="22"/>
         <declare type="string" name="user" required="required" />
      <!-- Sample: Use key file or password -->
         <declare type="boolean" name="use_key" initial-value="true" />
      <!-- Sample: If use key, input key and passphrase -->
         <condition name="use_key" enabled="true">
            <declare type="string" name="key" hidden="false" />
            <declare type="password" name="passphrase" hidden="false" />
         </condition>
      <!-- Sample: If use password, input password -->
         <condition name="use_key" enabled="false">
            <declare type="password" name="secret_key" hidden="false" />
         </condition>
   </proto-section>
```

Section: `crawlspace_settings`

The XPath expression is `/function/prototype/proto-section[@section="crawlspace_settings"]`. The section contains only one `<declare />` element to specify the path. The value of the path is provided by the connector code.

```
<!-- Do not modify, must be here -->
   <proto-section section="crawlspace_settings" cardinality="multiple">
      <declare type="string" name="path" hidden="true" />
   </proto-section>
```

Properties file

For an example of a properties file, see the example `messages.properties` file whose location is listed in Understanding the `custom-crawler-docs.zip` file.

JAR files

The JAR files for any interfaces used by your custom connector code, including the `ama-zing-custom-crawler-{version_numbers}.jar` file whose location is listed in Understanding the `custom-crawler-docs.zip` file. The `ama-zing-custom-crawler-{version_numbers}.jar` file includes the `com.ibm.es.ama.custom.crawler` Java package that is described in Developing custom connector code.

Compiling and packaging the custom connector

After you write the source code and configuration files for your custom connector, you need to compile and package it.

Prerequisites

To compile a custom connector, you need to have the following items on your local system. See Custom connector example for details.

- Java SDK 1.8 or higher
- Gradle
- The `custom-crawler-docs.zip` file from an installed Discovery instance
- The JSch package
- The following files for the example custom connector:
  - Java source code (`SftpCrawler.java` and `SftpSecurityHandler.java`)
  - XML definition file (`template.xml`)
  - Properties file (`messages.properties`)

> ⚠ **Important:** Do not change the names or paths of the example custom connector files. Doing so can result in problems, including build failures.

Compiling and packaging the source code

1. Ensure you are in the custom connector development directory on your local system:

```
$ cd {local_directory}
```

2. Use Gradle to compile your Java source code and to create a compressed file that includes all of the required components for the custom connector:

```
$ gradle build packageCustomCrawler
```

Gradle creates a file in `{local_directory}/build/distributions/{built_connector_zip_file}`, where the name of the `{built_connector_zip_file}` is based on the `rootProject.name` value of `settings.gradle`. For example, if the line reads as follows, Gradle generates a file that is named `{local_directory}/build/distributions/my-sftp-connector.zip`.

```
rootProject.name = 'my-sftp-connector'
```

Next step

Proceed to Installing and uninstalling a custom connector to install the custom connector to your Discovery instance.

Installing a custom Cloud Pak for Data connector

After you have compiled and packaged your custom connector, you need to install it to your Discovery instance.

IBM Cloud Pak for Data **IBM Cloud Pak for Data only**

Note: This information applies only to installed deployments.

Discovery provides a script named `manage_custom_crawler.sh` for installing and uninstalling custom connectors. The script is located in the `scripts` directory of the expanded `custom-crawler-docs.zip` file as described in Understanding the `custom-crawler-docs.zip` file.

Installing a connector

You can install your custom connector to your Discovery instance by performing the following steps.

1. Ensure that you have completed all steps to create a custom connector up to and including the steps listed in Compiling and packaging the example connector.

2. Run the following command from the directory on your local machine where you created and compiled your custom connector:

```
$ bash scripts/manage_custom_crawler.sh --endpoint {endpoint} --token {access token} deploy -n {crawler
name} -f {built_connector_zip_file}
```

   where you specify values for the following variables:

   - endpoint: URL for your service instance. You can get this value from the *Access information* section of the service instance overview page in the IBM Cloud Pak for Data administrative console.
   - access token: Bearer token that is required to access the endpoint. You can get this value from the same page as the endpoint.
   - crawler name: (Optional) Name that you specified for the crawler.
   - `{built_connector_zip_file}` is the name of the file you created in Compiling and packaging the example connector.

   For example:

```
$ bash scripts/manage_custom_crawler.sh --endpoint https://mycpd.wd40.example.com/discovery/zen40-
wd/instances/1638165624521059/api --token
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImVKcV9HY29NcHF5WUFJcVByZ0x0cERRZDNQcmRiTWo5TGg0X09WOEU4MlkifP.eyJ1aXQiOiI
-tKyznA_wjk_G698fbx1Zl73KZKyEWcTKtyX7IJ1Px5DPdophcqS9i3bPJowHy-ioVp6DML02mscZImhvZPra-e6gwUdhSB64KArmMClo1-
kZG20EclNh6-
oxR447Bjdsgp7IYpkmynmw0K6vPIqmzwEhr9gAK1vWLOoVd4EoiYNuxZaSFL5byJ0mnQxXzM14w3lKQHZ91WYVKc4JnuJiSVsdpGqVz1JNFmT8D9FBqJQ4
-USKCbJmMPXicU8cDtJIfheBejwenfvejUTz5rgZgymYWrGvw3G2oOx_L1Yg-Q deploy -n awesome_crawler -f
awesome_crawler.zip
```

   Instead of specifying an access token for authentication, you can specify username and password parameters. For more information, see Understanding the `manage_custom_crawler.sh` script.

When the custom crawler is deployed, a resource ID is assigned to the connector.

Verifying an installed connector

Verify that the connector has been deployed to the Discovery instance by logging into the Discovery tooling and ensuring that your connector is displayed as an option on the **Configure collection** page.

Using an installed connector on Discovery

To use the installed custom connector, follow the steps listed in Creating a collection. The custom connector appears in the list of connectors provided at Configuring Cloud Pak for Data data sources. For more information, see Using a custom connector with the Discovery tooling.

Uninstalling a connector

To uninstall a custom connector from a Discovery instance, complete the following steps:

1. Optional: If you don't know the resource ID, run the following command to list the custom connectors. The resource IDs of the connectors are returned.

   ```
   $ scripts/manage_custom_crawler.sh --endpoint {endpoint} --token {token} list
   ```

2. Run the following command from the directory where you extracted your custom connector ZIP file to uninstall the connector:

   ```
   $ scripts/manage_custom_crawler.sh --endpoint {endpoint} --token {token} undeploy --id
   {crawler_resource_id}
   ```

   where `{crawler-resource-id}` is the ID that is generated for the crawler when it is deployed.

   ```
   $ scripts/manage_custom_crawler.sh --endpoint {endpoint} --token {token} undeploy --id
   {crawler_resource_id}
   ```

Instead of specifying an access token for authentication, you can specify username and password parameters. For more information, see Understanding the `manage_custom_crawler.sh` script.

Understanding the `manage_custom_crawler.sh` script

The `manage_custom_crawler.sh` script has the following internal documentation:

```
Watson Discovery Custom Crawler Manager

This script will help you deploy, manage, and undeploy your custom crawler for
Watson Discovery.

Subcommands:
  deploy        Add a new Custom Crawler to your Watson Discovery instance.
  undeploy      Undeploy your Custom Crawler by name.
  list          List all Custom Crawlers for your Watson Discovery instance.

Options:
  -e --endpoint         The endpoint URL for your cluster and add-on service instance
                        (`https://{cpd_cluster_host}:{port}/discovery/{release}/instances/{instance_id}/api`)
  -t --token            The authorization token of your Cloud Pak instance
  -u --user             The user name of your Cloud Pak instance
  -p --password         The user password of your Cloud Pak instance
                        If the password is not specified, the command line prompts to input
  -n --name             The name of the custom crawler to upload (deploy only)
  -f --file             The path of the custom crawler package to upload (deploy only)
  -i --id               The crawler_resource_id value to delete the custom crawler (undeploy only)
  --help                Show this message.
```

4.0.5 and earlier releases only

Installing a connector in 4.0.5 and earlier releases

You can install your custom connector to your Discovery instance by performing the following steps.

1. Ensure that you have completed all steps to create a custom connector up to and including the steps listed in Compiling and packaging the example connector.

2. Run the following command from the directory on your local machine where you created and compiled your custom connector:

   ```
   $ bash scripts/manage_custom_crawler.sh deploy -z {built_connector_zip_file}
   ```

   where `{built_connector_zip_file}` is the name of the file you packaged in Compiling and packaging the example connector.

   ⚠ **Important:** If your Discovery instance is running on Red Hat OpenShift, specify the `-o` or `--openshift` parameter with the script.

   For example:

   ```
   $ bash scripts/manage_custom_crawler.sh deploy -z myCrawler.zip -o true
   ```

Uninstalling a connector in 4.0.5 and earlier releases

To uninstall a custom connector from a Discovery instance, run the following command at the root of the unzipped `custom-crawler-docs.zip` directory:

```
$ bash scripts/manage_custom_crawler.sh undeploy -n {built_connector_name}
```

where `{build_connector_name}` is the name, not the zip file, of the installed connector.

> ⚠ **Important:** If your IBM Watson® Discovery instance is running on Red Hat OpenShift, specify the `-o` or `--openshift` parameter with the script.

```
$ bash scripts/manage_custom_crawler.sh undeploy -n {built_connector_name} -o true
```

Understanding the `manage_custom_crawler.sh` script in 4.0.5 and earlier releases

The `manage_custom_crawler.sh` script has the following internal documentation:

```
Usage: ${BASH_SOURCE[0]} [--pathToZip PATH] [--properties PROPERTIES] [--xml XML]

Watson Discovery Custom Crawler Manager

This script will help you deploy, manage, and undeploy your custom crawler for
Watson Discovery.

Subcommands:
  deploy        Add a new Custom Crawler to your Watson Discovery instance.
  properties    Generate the properties file for your crawler.
  undeploy      Undeploy your Custom Crawler by name.
  list          List all Custom Crawlers for your Watson Discovery instance.

Options:
  -d --discovery        The name of the Watson Discovery instance
  -z --zipfile          The path to the zip file to be uploaded.
                        For deploy only.
  -x --xml              The path to the XML file to be uploaded.
                        For deploy only.
  -n --name             The name of the Custom Crawler to undeploy.
  -m --messages         The path to the properties file, used when doing a two part deploy.
                        For properties only.
  -o --openshift        Set flag to true if this is an OpenShift Cluster
  --help                Show this message.
```

Using a custom Cloud Pak for Data connector from the Discovery user interface

After you build and deploy a custom connector, you can configure and run it in the Discovery user interface to create a collection.

IBM Cloud Pak for Data **IBM Cloud Pak for Data only**

> **Note:** This information applies only to installed deployments.

You create and manage a collection as described in Creating and managing collections. You can use a successfully deployed custom connector during this process as follows. Follow these instructions to use a custom connector instead of one of the pre-built connectors that are listed in Configuring Cloud Pak for Data data sources.

1. After you create a project, look for your custom connector to connect to a data source.
2. Select the custom connector and then click **Next**.

   The **Configure collection** page opens.

   > **Note:** The following steps apply specifically to the example custom connector that is included with the `custom-crawler-docs.zip` file.

3. Enter values for the following fields on the **Configure collection** page. If a field is already populated with a value, verify and change the value if needed. A prepopulated value indicates that a value was specified in the custom connector's `template.xml` or `message.properties` file.

   General

   Complete the following fields

- Collection name
- Collection language
- Crawl schedule

Crawler properties

Complete the following fields

- Crawler name
- Crawler description
- Time to wait between retrieval requests (milliseconds)

  The default value is `0`.
- Maximum number of active crawler threads

  The default value is `10`.
- Maximum number of documents to crawl

  The default value `2000000000`.
- Maximum document size (KB)

  The default value is `32768`.

Data source properties

Complete the following fields

- Host name

  The default value is `localhost`.
- Port

  The default value is `22`.
- User name
- Use key file (or input password)

  The default value is `On`.
- Key file location
- passphrase
- Password

Crawl Space Properties

If the custom crawler supports document-level security and the `document_level_security_supported` value in the `template.xml` is set to `true`, then an **Enable Document Level Security** switch is displayed in a *Security* section of the data source connection setup page. To enable document-level security, set the **Enable Document Level Security** switch to **On**. If the switch is set to Off, then the collection that is created cannot support document-level security even if the custom crawler can support document-level security.

4. Click **Finish** to create the collection.

Configuring a Cloud Pak for Data custom crawler plug-in

Building a Cloud Pak for Data custom crawler plug-in

Discovery features the option to build your own crawler plug-in with a Java SDK. By using crawler plug-ins, you can now quickly develop relevant solutions for your use cases. You can download the SDK from your installed Discovery cluster. For more information, see Obtaining the crawler plug-in SDK package.

IBM Cloud Pak for Data **IBM Cloud Pak for Data only**

Note: This information applies only to installed deployments.

> **Note:** Any custom code that you use with IBM Watson® Discovery is the responsibility of the developer; IBM Support does not cover any custom code that the developer creates.

The crawler plug-ins support the following functions:

- Update the metadata list of a crawled document
- Update the content of a crawled document
- Exclude a crawled document
- Reference crawler configurations, masking password values
- Show notice messages in the Discovery user interface
- Output log messages to the `crawler` pod console

However, the `crawler` plug-ins cannot support the following functions:

- Split a crawled document into multiple documents
- Combine content from multiple documents into a single document
- Modify access control lists

Crawler plug-in requirements

Make sure that the following items are installed on the development server that you plan to use to develop a `crawler` plug-in by using this SDK:

- Java SE Development Kit (JDK) 1.8 or higher
- Gradle
- cURL
- sed (stream editor)

Obtaining the crawler plug-in SDK package

1. Log in to your Discovery cluster.
2. Enter the following command to obtain your `crawler` pod name:

   ```
   $ oc get pods | grep crawler
   ```

   The following example shows sample output.

   ```
   wd-discovery-crawler-57985fc5cf-rxk89     1/1     Running     0          85m
   ```

3. Enter the following command to obtain the SDK package name, replacing `{crawler-pod-name}` with the `crawler` pod name that you obtained in step 2:

   ```
   $ oc exec {crawler-pod-name} -- ls -l /opt/ibm/wex/zing/resources/ | grep wd-crawler-plugin-sdk
   ```

   The following example shows sample output.

   ```
   -rw-r--r--. 1 dadmin dadmin 35575 Oct  1 16:51 wd-crawler-plugin-sdk-${build-version}.zip
   ```

4. Enter the following command to copy the SDK package to the host server, replacing `{build-version}` with the build version number from the previous step:

   ```
   $ oc cp {crawler-pod-name}:/opt/ibm/wex/zing/resources/wd-crawler-plugin-sdk-${build-version}.zip wd-crawler-plugin-sdk.zip
   ```

5. If necessary, copy the SDK package to the development server.

Building a crawler plug-in package

1. Extract the SDK compressed file.
2. Implement the plug-in logic in `src/`. Ensure that the dependency is written in `build.gradle`.
3. Enter `gradle packageCrawlerPlugin` to create the plug-in package. The package is generated as `build/distributed/wd-crawler-plugin-sample.zip`.

Developing and implementing a Cloud Pak for Data custom crawler plug-in

The `crawler` plug-in includes a file that is called `com.ibm.es.ama.plugin.CrawlerPlugin`. This file is the initialization interface that has methods you can use when you work with your `crawler` plug-in.

IBM Cloud Pak for Data **IBM Cloud Pak for Data only**

> **Note:** This information applies only to installed deployments.

Interfaces and Javadoc

The interface library is stored in the `lib/ama-zing-crawler-plugin-${build-version}.jar` directory of the SDK directory. The Javadoc for the JAR file is available in the `lib/ama-zing-crawler-plugin-${build-version}-javadoc.jar` file in the same directory.

Initialization interface

Use the `com.ibm.es.ama.plugin.CrawlerPlugin` interface to manage the `crawler` plug-in. The interface has the following methods:

Method	Description
`init`	Start a crawler plug-in
`term`	Stop a crawler plug-in
`updateDocument`	Update crawled documents

Supported methods

Dependency management

The file `build.gradle` manages the Java dependencies.

Crawler plug-in sample

A sample `crawler` plug-in is available that illustrates how to add, update, and delete metadata. The plug-in example also updates and deletes documents that are crawled by the local file system connector. The Java source code file is named `src/main/java/com/ibm/es/ama/plugin/sample/SampleCrawlerPlugin.java`.

Logging messages

The custom `crawler` plug-in supports the `java.util.logging.Logger` package for logging messages.

Any log messages that you add must meet the following requirements:

- The log level must be `INFO` or higher.
- The logger name must start with `com.ibm.es.ama`.

Messages are written to the log file of the `crawler` pod where the plug-in is running. A logging sample is available in the `crawler` plug-in sample.

Assembling and compiling a custom crawler plug-in

After you write the source code for your crawler plug-in, you must assemble and compile it.

IBM Cloud Pak for Data **IBM Cloud Pak for Data only**

> **Note:** This information applies only to installed deployments.

Prerequisites

You must have the following items to compile a crawler plug-in:

- Java SE Development Kit 1.8 or higher
- Gradle
- cURL
- sed (stream editor)
- Crawler plug-in SDK package, see Obtaining the crawler plug-in SDK package

Assembling and compiling the crawler plug-in

1. Specify the class name of the crawler plug-in by opening the `config/template.xml` file and modifying the `initial-value` of the `crawler_plugin_class` element.
2. Ensure that you are in the crawler plug-in SDK directory on your development server.

3. Enter `gradle packageCrawlerPlugin` to use Gradle to compile your Java source code and to create a compressed file that includes all of the required components for the crawler plug-in.
4. Confirm that you have access to the crawler plug-in package, which is in the `build/distributions/wd-crawler-plugin-sample.zip` file.

Managing custom crawler plug-ins on your Watson Discovery cluster

You can use the `scripts/manage_crawler_plugin.sh` script to perform common plug-in management actions. The `scripts/manage_crawler_plugin.sh` script is located in the crawler plug-in SDK package. When you use the script in a command, you must have the endpoint URL of your Discovery cluster and the username and password of your IBM Cloud Pak® for Data instance.

IBM Cloud Pak for Data **IBM Cloud Pak for Data only**

> **Note:** This information applies only to installed deployments.

Commands and options for managing your crawler plug-ins

You can enter `scripts/manage_crawler_plugin.sh --help` to view the following help messages in the script:

```
Usage: scripts/manage_crawler_plugin.sh --endpoint endpoint --user username [--password password] command

Watson Discovery Crawler Plug-in Manager

This script will help you deploy, undeploy, and list your crawler plug-ins for Watson Discovery.

Commands:
deploy        Add a new crawler plug-in to your Watson Discovery instance
undeploy      Undeploy your crawler plug-in by ID
list          List all crawler plug-ins for your Watson Discovery instance (default)

Options:
-e --endpoint         The endpoint URL for your cluster and add-on service instance
                      (https://{cpd_cluster_host}{: port}/discovery/{release}/instances/{instance_id}/api)
-u --user             The user name of your Cloud Pak instance
-p --password         The user password of your Cloud Pak instance
                      If the password is not specified, the command line prompts to input
-n --name             The name of the crawler plug-in to upload (deploy only)
-f --file             The path of the crawler plug-in package to upload (deploy only)
--id                  The crawler_resource_id value to delete the crawler plug-in (undeploy only)
--help               Show this message
```

You can use the following commands to deploy, undeploy, and list your crawler plug-ins. Replace the variable references `{variable}` with the required information:

- Deploy crawler plug-in: `scripts/manage_crawler_plugin.sh --endpoint {endpoint_URL} --user {username} deploy --name {plugin_name}`
- Undeploy crawler plug-in: `scripts/manage_crawler_plugin.sh --endpoint {endpoint_URL} --user {username} undeploy --id {crawler_resource_id}`
- List deployed crawler plug-in: `scripts/manage_crawler_plugin.sh --endpoint {endpoint_URL} --user {username} list`

After you use the `scripts/manage_crawler_plugin.sh` script to deploy a crawler plug-in, you can select the plug-in in the Discovery tooling when you create a collection. For more information about the crawler plug-in settings in Discovery, see Crawler plug-in settings.

Using a Cloud Pak for Data custom crawler plug-in from the Discovery user interface

After you build and deploy a crawler plug-in, you can configure your Discovery collection to use your plug-in to process documents.

IBM Cloud Pak for Data **IBM Cloud Pak for Data only**

> **Note:** This information applies only to installed deployments.

- You can create and manage a collection as described in Creating and managing collections.
- You can select a successfully deployed crawler plug-in when you create and manage a collection.

  For more information, see Crawler plug-in settings.
- You can also deploy a crawler plug-in package to a testing environment.

Glossary

Term	Definition
Classifier	A resource that you can train to recognize document types and categorize them in your collection. You can create two types of classifiers, a *text classifier* and a *document classifier*. A *text classifier* can classify documents based on words and phrases that are extracted from the body text with their part of speech information taken into account. A *document classifier* can classify documents based on words and phrases that are extracted from the body text fields with information from their part of speech and the other enrichments that are applied to the body text taken into account. The information from the other nonbody fields are also used. Learn more
Collection	A set of documents that you can enrich and later search for meaningful information. Learn more.
Content Intelligence	A feature that you can use to enrich documents in a Document Retrieval project such that the project can recognize information that is relevant to business contracts. A Document Retrieval project with this feature enabled is referred to as a *Document Retrieval for Contracts* project type. Learn more
Data source	An external application or service where valuable knowledge resources are stored. Connect to a service where your data is stored so you can crawl the data without having to move it. Learn more.
Domain-specific	Relates to terms and concepts that have special meaning to an industry or business. In tennis, for example, the term *love* has a special meaning. It represents a zero score. If you built a tennis-related application, you would teach Discovery that the term *love* has a meaning in the domain of tennis that is different from the generally understood meaning.
Enrichment	What you add to documents in your collection to identify or *tag* terms in the document that are significant. For example, when you apply the Entity enrichment, terms that mention city names or famous people are tagged as locations or people of interest.
Facet	A category by which you can filter search query results. Automatically, facets based on entity types are applied to the query results for Document Retrieval projects and facets based on the parts of speech are applied to Content Mining projects. You can define your own facet categories based on document fields, including fields generated by enrichments, or based on dictionaries or patterns. Learn more
Index	As you upload data or connect to data that is stored in an external repository, the data is crawled and ingested. As part of the processing, an index is created to keep track of important information that is recognized from the source. The main difference between a data source and a collection is that content in the data source is crawled, normalized, and indexed as it is added to a collection.
Project	A container for the collections of data that fuel your research or search applications. Learn more.
Regular expression	A regular expression, also known as a *regex*, is a standardized format for defining search patterns. You can define patterns with special significance to your application. For example, the bill of materials (BOM) numbers for parts that you manufacture might have a standard syntax of two uppercase letters followed by four numbers (**GT2345**). You can teach Discovery to recognize BOM mentions by adding a regular expression that can recognize and tag occurrences of the pattern in text. Learn more.
Stopword	Words to filter out of queries because they are not useful in a search, such as `a`, `an`, and `the`. You can add words that are common and not useful to your use case as stopwords to improve the relevance of results for natural language queries. Learn more.

Definitions of Discovery terms

Watson SDKs

SDKs abstract much of the complexity associated with application development. By providing programming interfaces in languages that you already know, they can help you get up and running quickly with IBM Watson services.

Supported SDKs

The following Watson SDKs are supported by IBM:

- Java SDK
- Node.js SDK
- Python SDK
- .NET SDK

Tip: The API reference for each service includes information and examples for these SDKs.

Community SDKs

The following SDKs are available from the Watson community of developers:

- ABAP SDK for IBM Watson, using SAP NetWeaver
- Android SDK
- Go SDK
- Ruby SDK
- Salesforce SDK
- Swift SDK
- Unity SDK

SDK updates and deprecation

The supported Watson SDKs are updated according to the following guidelines.

Semantic versioning

Supported Watson SDKs adhere to semantic versioning with releases labeled as `{major}.{minor}.{patch}`.

Release frequency

SDKs are released independently and might not update on the same schedule.

- The current releases of the Watson SDKs are updated on a 2- to 6-week schedule. These releases are either minor updates or patches that do not include breaking changes. You can update to any version of the SDK with the same major version number.
- Major updates that might include breaking changes are released approximately every 6 months.

Deprecated release

When a major version is released, support continues on the previous major release for 12 months in a deprecation period. The deprecated release might be updated with bug fixes, but no new features will be added and documentation might not be available.

Obsolete release

After the 12-month deprecation period, a release is obsolete. The release might be functional but is unsupported and not updated. Update to the current release.

Query reference

The full list of Discovery query parameters, operators, and aggregations. You can refer to this information for help when you write queries with the Discovery Query Language.

- For more information, see the Discovery API reference.
- For an overview of query concepts, see the Query overview.

Tip: In the Discovery user interface, you can write and test natural language queries on the *Improve and customize* page.

Parameters descriptions

Use query parameters to search your collection, identify a result set, and analyze result sets.

Parameter	Description	Example
`aggregation`	A statistical query of the results set	`aggregation=term(enriched_text.entities.type)`
`filter`	An unranked query language search for matching documents.	`filter=bees`
`natural_language_query`	A ranked natural language search for matching documents	`natural_language_query="How do bees fly"`
`query`	A ranked query language search for matching documents.	`query=bees`

Search parameters

Parameter	Description	Example
`count`	The number of `result` documents to return.	`count=15`
`highlight`	Highlight query matches	`highlight=true`
`offset`	The number of results to ignore before returning `result` documents from the results set	`offset=100`
`return`	List of fields to return	`return=title,url`
`sort`	Field to sort results set by	`sort=enriched_text.sentiment.document.score`
`spelling suggestions`	Spelling suggestions returned for natural language queries	`spelling_suggestions=true`

Structure parameters

Query limitations

You cannot query on field names that contain the following characters:

- Numbers (`0 - 9`) in the suffix of the field name. For example, `extracted-content2`.
- The characters `_`, `+`, and `-` in the prefix of the `field_name`. For example, `+extracted-content`.
- The characters `.`, `,`, and `:` in the `field_name`. For example, `new:extracted-content`.

Operators

Operators are the separators between different parts of a query. The following table lists the available operators.

Operator	Description	Example
`.`	JSON delimiter	`enriched_text.concepts.text`

`:`	Includes	`text:computer`
`::`	Exact match	`title::"Query building"`
`:!`	Does not include	`text:!computer`
`::!`	Not an exact match	`text::!winter`
`\`	Escape character	`title::"Dorothy said: \"There's no place like home.\""`
`""`	Phrase query	`enriched_text.concepts.text:"IBM Watson"`
`(), []`	Nested groups	`filter-entities:(text:Turkey,type:Location)`
`|`	or	`query-enriched.entities.text:Google|IBM`
`,`	and	`query-enriched.entities.text:Google,IBM`
`<=, >=, >, <`	Numerical comparisons	`enriched_text.sentiment.document.score>0.679`
`^x`	Score multiplier	`text:IBM^3`
`*`	Wildcard	`query-enriched_text.concepts.text:pre*`
`~n`	String variation	`query-enriched_text.entities.text:cat~1`
`:*`	Exists	`title:*`
`!*`	Does not exist	`title!*`

Query operators

Aggregations

Aggregations return a set of data values. The following table lists the available aggregations.

Aggregation	Description	Example
`average`	Mean value for the specified field in the results set.	`average(product.price)`
`filter`	Filter results based on the specified pattern	`filter(enriched_text.concepts.text:cloud computing)`
`histogram`	Interval-based distribution	`histogram(product.price,interval:1)`
`max`	Maximum value for the specified field in the results set.	`max(product.price)`
`min`	Minimum value for the specified field in the results set.	`min(product.price)`
`nested`	Restrict aggregation	`nested(enriched_text.entities)`
`sum`	Sum of all fields in the results set.	`sum(product.price)`
`term`	Count of identical values	`term(enriched_text.concepts.text,count:10)`
`timeslice`	Time-based distribution	`timeslice(last_modified,2day,America/New York)`

top_hits	Top-ranked result documents for the current aggregation	`term(enriched_text.concepts.text).top_hits(10)`
unique_count	Count of unique values for a field within an aggregation	`unique_count(enriched_text.entities.type)`

Query aggregations

Language support

When you create a collection, you specify the language of the collection. All of the documents that you add to a collection must be written in the same language.

Note: Discovery is not optimized for multilingual search. Although you can add several collections, each one with documents in a separate language, into one project, the query results from the project will be unpredictable. The results might include irrelevant passages from a document in a language that is different from the language of the user's query.

The following table describes the product features that are supported in each language.

Language	Supported features
Arabic (`ar`)	Advanced rules models, Built-in entities, Classifier (Document and Text), Custom entities, Dictionary, Document sentiment, Keywords, Machine Learning, Optical character recognition v1, Parts of speech, Regular expressions, Smart Document Understanding, Stemmer, Table Understanding
Bosnian (`bs`)	Classifier (Document and Text), Custom entities, Dictionary, Parts of speech, Regular expressions
Chinese, simplified (`zh-CN`)	Advanced rules models, Built-in entities, Classifier (Document and Text), Custom entities, Dictionary, Document sentiment, Keywords, Machine Learning, Optical character recognition v1, Parts of speech, Phrase sentiment, Regular expressions, Smart Document Understanding, Table Understanding
Chinese, traditional (`zh-TW`)	Advanced rules models, Classifier (Document and Text), Custom entities, Dictionary, Regular expressions, Machine Learning, Optical character recognition v1, Parts of speech, Phrase sentiment, Smart Document Understanding, Table Understanding
Croatian (`hr`)	Classifier (Document and Text), Custom entities, Dictionary, Regular expressions, Parts of speech
Czech (`cs`)	Classifier (Document and Text), Custom entities, Dictionary, Optical character recognition v1, Parts of speech, Phrase sentiment, Regular expressions, Smart Document Understanding, Stemmer, Table Understanding
Danish (`da`)	Classifier (Document and Text), Custom entities, Dictionary, Optical character recognition v1, Parts of speech, Regular expressions, Smart Document Understanding, Stemmer, Table Understanding
Dutch (`nl`)	Advanced rules models, Built-in entities, Classifier (Document and Text), Custom entities, Dictionary, Document sentiment, Keywords, Machine Learning, Optical character recognition v2, Parts of speech, Phrase sentiment, Regular expressions, Smart Document Understanding, Stemmer, Table Understanding
English (`en`)	Advanced rules models, Built-in entities, Classifier (Document and Text), Contracts, Custom entities, Dictionary, Document sentiment, Keywords, Machine Learning, Optical character recognition v2, Parts of speech, Phrase sentiment, Regular expressions, Smart Document Understanding, Stemmer, Table Understanding
Finnish (`fi`)	Classifier (Document and Text), Custom entities, Dictionary, Parts of speech, Regular expressions, Smart Document Understanding, Stemmer, Table Understanding
French (`fr`)	Advanced rules models, Built-in entities, Classifier (Document and Text), Custom entities, Dictionary, Document sentiment, Keywords, Machine Learning, Optical character recognition v2, Parts of speech, Regular expressions, Smart Document Understanding, Stemmer, Table Understanding
German (`de`)	Advanced rules models, Built-in entities, Classifier (Document and Text), Custom entities, Dictionary, Document sentiment, Keywords, Machine Learning, Optical character recognition v2, Parts of speech, Regular expressions, Smart Document Understanding, Stemmer, Table Understanding
Hebrew (`he`)	Classifier (Document and Text), Custom entities, Dictionary, Optical character recognition v2, Parts of speech, Regular expressions, Smart Document Understanding, Table Understanding. The optical character recognition (OCR) feature for Hebrew language text in images is a beta feature in Discovery. For more information, see Release notes for Discovery for IBM Cloud.
Hindi (`hi`)	Classifier (Document and Text), Custom entities, Dictionary, Parts of speech, Regular expressions, Stemmer

Italian (`it`)	Advanced rules models, Built-in entities, Classifier (Document and Text), Custom entities, Dictionary, Document sentiment, Keywords, Machine Learning, Optical character recognition v1, Parts of speech, Regular expressions, Smart Document Understanding, Stemmer, Table Understanding
Japanese (`ja`)	Advanced rules models, Built-in entities, Classifier (Document and Text), Custom entities, Dictionary, Document sentiment, Keywords, Machine Learning, Optical character recognition v1, Parts of speech, Phrase sentiment, Regular expressions, Smart Document Understanding, Table Understanding
Korean (`ko`)	Advanced rules models, Built-in entities, Classifier (Document and Text), Custom entities, Dictionary, Document sentiment, Keywords, Machine Learning, Optical character recognition v1, Parts of speech, Regular expressions, Smart Document Understanding, Table Understanding
Norwegian (Bokmål) (`nb`)	Classifier (Document and Text), Custom entities, Dictionary, Optical character recognition v1, Parts of speech, Regular expressions, Smart Document Understanding, Stemmer, Table Understanding
Norwegian (Nynorsk) (`nn`)	Classifier (Document and Text), Custom entities, Dictionary, Optical character recognition v1, Parts of speech, Regular expressions, Smart Document Understanding, Stemmer, Table Understanding
Polish (`pl`)	Classifier (Document and Text), Custom entities, Dictionary, Optical character recognition v1, Parts of speech, Regular expressions, Smart Document Understanding, Table Understanding
Portuguese, Brazilian (`pt-br`)	Advanced rules models, Built-in entities, Classifier (Document and Text), Custom entities, Dictionary, Document sentiment, Keywords, Machine Learning, Optical character recognition v2, Parts of speech, Regular expressions, Smart Document Understanding, Stemmer, Table Understanding
Romanian (`ro`)	Classifier (Document and Text), Custom entities, Dictionary, Optical character recognition v1, Parts of speech, Phrase sentiment, Regular expressions, Smart Document Understanding, Stemmer, Table Understanding
Russian (`ru`)	Classifier (Document and Text), Custom entities, Dictionary, Optical character recognition v1, Parts of speech, Phrase sentiment, Regular expressions, Smart Document Understanding, Stemmer, Table Understanding
Serbian (`sr`) [1]	Classifier (Document and Text), Custom entities, Dictionary, Parts of speech, Regular expressions
Slovak (`sk`)	Classifier (Document and Text), Custom entities, Dictionary, Optical character recognition v1, Parts of speech, Regular expressions, Smart Document Understanding, Table Understanding
Spanish (`es`)	Advanced rules models, Built-in entities, Classifier (Document and Text), Custom entities, Dictionary, Document sentiment, Keywords, Machine Learning, Optical character recognition v2, Parts of speech, Phrase sentiment, Regular expressions, Smart Document Understanding, Stemmer, Table Understanding
Swedish (`sv`)	Classifier (Document and Text), Custom entities, Dictionary, Optical character recognition v1, Parts of speech, Regular expressions, Smart Document Understanding, Stemmer, Table Understanding

Feature support per language

Note: Optical character recognition (OCR) v2 was introduced in Cloud-managed service instances on 2 November 2022. OCR v2 was introduced in IBM Cloud Pak for Data instances with version 4.7.1.

English-only support

The following features are currently supported in English only:

- Document Retrieval for Contract project type
- IBM Cloud Patterns (beta)

1. Serbian supports Latin script only. ↩

IBM Cloud security

Information security

IBM is committed to providing our clients and partners with innovative data privacy, security, and governance solutions.

IBM Cloud **IBM Cloud only**

> **Note:** This information applies only to managed deployments.

Notice: Clients are responsible for ensuring their own compliance with various laws and regulations, including the European Union General Data Protection Regulation. Clients are solely responsible for obtaining advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulations that may affect the clients' business and any actions the clients may need to take to comply with such laws and regulations.

The products, services, and other capabilities that are described herein are not suitable for all client situations and might have restricted availability. IBM does not provide legal, accounting, or auditing advice or represent or warrant that its services or products ensure that clients are in compliance with any law or regulation.

If you need to request GDPR support for IBM Cloud® Watson resources that are created, see GDPR Subject Access Request.

European Union General Data Protection Regulation (GDPR)

IBM is committed to providing our clients and partners with innovative data privacy, security, and governance solutions to assist them on their journey to GDPR compliance.

Learn more about IBM's own GDPR readiness journey and our GDPR capabilities and offerings to support your compliance journey here.

Labeling and deleting data in Discovery

Discovery includes an API to label data per call. For more information about how to label data by using either the API or from the Discovery product user interface, see Labeling data.

Customer data can be deleted by using the API. For more information about deleting customer data, see Deleting labeled data.

> **Note:** Experimental and beta features are not intended for use with a production environment and are not guaranteed to function as expected when you label and delete thier associated data. Do not use experimental and beta features when you implement a solution that requires the labeling and deletion of data.

Methods that support labeling data

The following stored information can be deleted by using a `customer_id` if the `customer_id` was specified when the information was originally added by using the associated method:

- Curations (`/v2/projects/{project_id}/curations`) Only available for `natural_language_query` query types.
- Documents (`/v2/projects/{project_id}/collections/{collection_id}/documents`)
- Notices (`/v2/projects/{project_id}/notices`) Only ingestion `notices` are labeled.
- Training data (`/v2/projects/{project_id}/training_data/queries`)
- Dictionaries (Only when created in the Discovery product user interface)
- Exported documents (Only when created in the Content Mining application)
- Reports (Only when created in the Content Mining application)

Exported documents and reports can be viewed in the *Repository* and *Report* pages of the Content Mining application. They are not available by using the API.

Discovery does not log query request data.

For more information about the options for labeling data in Discovery, see Labeling data.

The following stored information is not explicitly labeled and cannot be deleted by specifying the `customer_id`. Personal Data is not supported in these fields.

Any string fields (including but not limited to `name` and `description`) of the following stored items:

- Collections
- Projects

Labeling data

Data can be labeled by using the API, or by using the Discovery product user interface. For more information about labeling with the product user interface, see Labeling data in the product user interface .

> ⚠ **Important:** You cannot label data that is added by crawling external data sources.

Data is labeled by adding the `customer_id` of your choice to the optional `X-Watson-Metadata` header. Discovery can then delete it by `customer_id`.

You can label data with the API in different ways:

- When you ingest documents by using the `POST /v2/projects/{project_id}/collections/{collection_id}/documents` or `POST /v2/projects/{project_id}/collections/{collection_id}/documents/ID` operations, send an optional header `X-Watson-Metadata`. The `X-Watson-Metadata` header must include either of the following items:
  - Semicolon separated `field=value` pairs (for example: `customer_id=123`)
  - The `customer_id` field. By adding the `customer_id` in `X-Watson-Metadata` header, the request indicates that it contains data that belongs to this `customer_id`.

Optionally, you can include the `customer_id` field with the `metadata` multipart form part instead of including the `X-Watson-Metadata` header.

> **Note:** If you specify a `customer_id` in the `metadata` multipart form part and the `X-Watson-Metadata` header for the same document, then the `customer_id` in the `X-Watson-Metadata` header is used.

This example adds the `customer_id` to both the `X-Watson-Metadata` header and the `metadata`:

```
$ curl -k -u "apikey:$API_KEY" \
   -H "x-watson-userinfo:instance-id=asdf" \
   -H "x-watson-metadata:customer_id=customer_header_123" \
   -H "x-watson-discovery-next:true" \
   -F "file=@$FILENAME" \
   -F "metadata={\"customer_id\": \"new123\"}" \
  -X POST "$API_URL/v2/projects/$PROJECT_ID/collections/$COLLECTION_ID/documents?version=2020-03-08" \
```

Example output:

```
$ {
   "document_id":"8b152926-e9f5-4f34-940a-c02da7ef3af4",
   "result_metadata":{
      "collection_id":"24265c0b-2a55-3ccf-0000-017334467b6e"
   },
   "metadata":{
      "date":1594319812384,
      "parent_document_id":"8b152926-e9f5-4f34-940a-c02da7ef3af4",
      "customer_id":"customer_header_123"
   },
   "extracted_metadata":{
      "sha1":"CEC7C1D3423C7D4ED58FC448F52681ECA93CED8A",
      "numPages":"1",
      "filename":"Simple.pdf",
      "author":[
         "Simple Man"
      ],
      "subject":"Simple Metadata",
      "file_type":"pdf",
      "title":"Simple Title",
      "publicationdate":"2016-10-05"
   }
}
```

> ⚠ **Important:** If your documents are already ingested, you must reingest them to add the `X-Watson-Metadata` header and `customer_id`.

Restrictions:

- The value of the `X-Watson-Metadata` header cannot exceed 4 KB of text.
- The `X-Watson-Metadata` header must contain a semicolon-separated list of `field=value` pairs. The `field` and `value` must not contain semicolons (`;`) or equals signs (`=`).
- `customer_id`s are unique within each Discovery instance. They are NOT unique per project or collection.
- A `customer_id` cannot be more than 256 characters in length.
- If a `customer_id` contains only white space or is empty, it is treated as though the `customer_id` was not provided at all, and no error messages are returned.

Labeling data in the product user interface

Data can be labeled by using the Discovery product user interface, or by using the API. For more information about labeling with the API, see Labeling data.

To label data with the product user interface:

1. Open the **Projects** page by selecting **My Projects**.
2. Select **Data usage and GDPR**.
3. Choose the **GDPR data label** tab.
4. Set the **Label data with customer ID** toggle to `on`. The **Customer ID** field appears.
5. Enter a unique ID for the customer in the **Customer ID** field. Do not include personal data in a **Customer ID**.
6. Click **Save ID**.

After the **Customer ID** (`customer_id`) field is set, all data that is uploaded during the current browser session is labeled with the specified **Customer ID**. (You cannot label data that is added by crawling external data sources.)

Adding a **Customer ID** labels the documents, notices, dictionaries, and training data within that URL domain from that point forward, including each instance under that domain. Any actions, including document uploads, that occurred in the Discovery product user interface before the **Customer ID** field was added are not labeled.

Note: If you switch domains or browsers, empty the browser cache, or start an incognito session after you specify your **Customer ID** by using the Discovery product user interface, the **Customer ID** is not retained, and your data is not labeled. If you must switch domains or browsers, after the switch, open the **GDPR data label** tab, enter the **Customer ID** again, and then click **Save ID**.

If an existing **Customer ID** needs to be changed:

1. Delete the data associated with that **Customer ID**. For instructions, see Deleting labeled data.
2. Follow the instructions to label data with the Discovery product user interface, or by using the API.
3. Upload or crawl the data.

Deleting labeled data

Customer data that is labeled with a `customer_id` can be deleted by using the API. For more information about how to label data by using either the API or from the Discovery product user interface, see Labeling data.

Note: You cannot delete labeled data from the Discovery product user interface.

1. Use the `DELETE /v2/user_data` operation and provide the `customer_id` of the data you want to delete.
   - `DELETE /v2/user_data` deletes all data that is associated with a particular `customer_id` within that service instance, as specified in Methods that support labeling data. Also, see **Delete labeled data** in the API reference
2. To ensure all labeled content is correctly removed, run the `DELETE /v2/user_data` operation after the `processing` and `pending` counts for your collections return `0`.

Notes on deleting labeled data:

- Deletions happen asynchronously. You cannot track the progress of deletions.
- If a nonexistent `customer_id` is provided, nothing is deleted, but a `202 - Accepted` response is returned.
- Projects and collections are not labeled with a `customer_id`, even if a `X-Watson-Metadata` header is included in the request to create the project or collection. Only the individual documents within a collection are labeled. Therefore, when data is deleted, individual projects and collections are NOT deleted.

Managing IAM access for Discovery

Share a preview of your search application or build a team to work on a project. You can give team members access to your Discovery service instance through IBM Cloud.

IBM Cloud

Important: The information in this topic applies to managed deployments only.

Access to IBM Cloud service instances is controlled by IBM Cloud® Identity and Access Management (IAM). Every person who accesses Discovery in your account must be assigned an access policy with an IAM role.

Only an owner of the service account can add users. For more information, see Managing access to resources.

Platform roles

A platform role controls a person's ability to access a service instance in IBM Cloud.

To give someone access to your service instance, assign them any platform role other than *Viewer*. A person with the *Viewer* role cannot access the product user interface. All of the other roles allow users to perform all tasks.

Service roles

A service role controls what a person can do in Discovery.

With Discovery, anyone who can access a Discovery instance can perform all actions. The only limitation is with the *Reader* role; anyone with reader-level access cannot submit API POST requests.

Activity Tracker events

As a security officer, auditor, or manager, you can use the Activity Tracker service to track how users and applications interact with the Discovery service in IBM Cloud®.

IBM Cloud

> **Note:** This information applies only to managed deployments.

IBM Cloud Activity Tracker records user-initiated activities that change the state of a service in IBM Cloud. You can use this service to investigate abnormal activity and critical actions and to comply with regulatory audit requirements. In addition, you can be alerted about actions as they happen. The events that are collected comply with the Cloud Auditing Data Federation (CADF) standard. For more information, see the getting started tutorial for IBM Cloud Activity Tracker.

List of events

The following table lists the Discovery actions that generate an event.

Action	Description
`discovery.analyze-api.read`	Process text by using the Analyze API.
`discovery.autocompletion.read`	Suggest complete queries based on documents.
`discovery.collection-notices.read`	Get notices for a collection.
`discovery.collection-training-status.read`	Get the training status of a single-collection training.
`discovery.collections.read`	Read collection annotations.
`discovery.content-miner-csv.create`	Import a CSV file to a Content Mining project.
`discovery.content-miner-csv.delete`	Delete a CSV file from a Content Mining project.
`discovery.content-miner-csv.read`	Get a CSV file from a Content Mining project.
`discovery.content-miner-export.create`	Create a set of exported documents from a Content Mining project.
`discovery.content-miner-export.download`	Download exported documents from a Content Mining project.
`discovery.content-miner-export.search`	List sets of exported documents from a Content Mining project.
`discovery.content-miner-report.create`	Create a report from a Content Mining project.
`discovery.content-miner-report.delete`	Delete a report from a Content Mining project.

`discovery.content-miner-report.read`	Get report content from a Content Mining project.
`discovery.content-miner-report.update`	Update report content from a Content Mining project.
`discovery.credential.create`	Create a credential.
`discovery.credential.delete`	Delete a credential.
`discovery.credential.read`	Get a credential, Salesforce objects, or a list of Cloud Object Storage buckets.
`discovery.credential.update`	Update a credential.
`discovery.curations.create`	Create a curated query.
`discovery.curations.delete`	Delete specified curation.
`discovery.curations.read`	List currently configured curation queries.
`discovery.curations.update`	Update existing curated results documents for a specified query.
`discovery.dataset.create`	Create a data set.
`discovery.dataset.update`	Update a data set.
`discovery.dataset-notices.read`	Get notices for a data set.
`discovery.dictionary.create`	Create a dictionary.
`discovery.dictionary.delete`	Delete a dictionary.
`discovery.dictionary.read`	Read a dictionary.
`discovery.dictionary.update`	Update a dictionary.
`discovery.document.add`	Add one document.
`discovery.document.create`	Create a document.
`discovery.document.delete`	Delete a document by ID.
`discovery.document.read`	Download a PDF version of a document.
`discovery.document.update`	Update one document by ingesting new or modified content for a document given a document ID, or by changing the label for a specified document ID.
`discovery.document-annotation.read`	Read document annotations.
`discovery.document-results.read`	Search collections for relevant documents.
`discovery.enrichment.create`	Create an enrichment.
`discovery.enrichment.delete`	Delete an enrichment.
`discovery.enrichment.read`	Read an enrichment.

`discovery.enrichment.update`	Update an enrichment.
`discovery.entity-extractor.create`	Create an entity extractor.
`discovery.entity-extractor.delete`	Delete an entity extractor.
`discovery.entity-extractor.read`	Read an entity extractor.
`discovery.entity-extractor.update`	Update an entity extractor.
`discovery.entity-extractor.download`	Download an entity extractor model or labeled data.
`discovery.event.create`	Add a click event to a query.
`discovery.expansions.create`	Add synonyms to a collection.
`discovery.expansions.delete`	Delete synonyms from a collection.
`discovery.expansions.read`	Get synonyms for a collection.
`discovery.fields.read`	Get fields for a collection.
`discovery.label.create`	Create a collection label.
`discovery.label.delete`	Delete one or multiple documents from one or multiple collections based on a label.
`discovery.label.read`	Read a collection label.
`discovery.label.update`	Update a collection label.
`discovery.logs.read`	Get logs for a collection.
`discovery.metric.read`	Request a metric.
`discovery.model.export`	Export a model.
`discovery.model.import`	Import a model.
`discovery.multi-collection-training-status.read`	Get the training status of a multi-collection training.
`discovery.notices.read`	Get notices for a collection.
`discovery.page.read`	Read document pages.
`discovery.page-annotation.add`	Add document page annotation.
`discovery.page-prediction.read`	Read document page predictions.
`discovery.page-view.read`	Read document page view.
`discovery.project.create`	Create a project.
`discovery.project.delete`	Delete a project.

`discovery.project-notices.read`	Get notices for a project.
`discovery.sentence-labeling.create`	Create a sentence labeling workspace.
`discovery.sentence-labeling.delete`	Delete a sentence labeling workspace.
`discovery.sentence-labeling.read`	Read a sentence labeling workspace.
`discovery.sentence-labeling.update`	Update a sentence labeling workspace.
`discovery.sentence-labeling.download`	Download a sentence classifier model or labeled data.
`discovery.stopwords.create`	Add stopwords to a collection.
`discovery.stopwords.delete`	Delete stopwords from a collection.
`discovery.stopwords.read`	Get stopwords for a collection.
`discovery.table-annotation.read`	Read document page table annotations.
`discovery.table-cell.read`	Read document page table cell.
`discovery.user-data.delete`	Delete all data associated with a customer ID.

Table 1. Actions that generate events

Viewing events

Events that are generated by an instance of the Discovery service are automatically forwarded to the IBM Cloud Activity Tracker service instance that is available in the same location.

IBM Cloud Activity Tracker can have only one instance per location. To view events, you must access the web UI of the IBM Cloud Activity Tracker service in the same location where your service instance is available.

To open the IBM Cloud dashboard, click the user icon in the page header, and then click **IBM Cloud dashboard**.

For more information about how to open IBM Cloud Activity Tracker from the dashboard, see Navigating to the UI.

Public and private network endpoints

IBM Cloud

IBM Cloud® supports both public and private network endpoints for certain plans. Connections to private network endpoints do not require public internet access.

Private network endpoints support routing services over the IBM Cloud private network instead of the public network. A private network endpoint provides a unique IP address that is accessible to you without a VPN connection.

Enabling your account

⚠ **Important:** Private network endpoints are supported for paid plans. Check the plan information for your service to learn about the plans that support private network endpoints.

Your account must be configured before you can use private endpoints. To use private network endpoints, the following account features must be enabled for your account.

- Virtual routing and forwarding (VRF).
- Service endpoints. Enabling service endpoints means that all users in the account can connect to private network endpoints.

To enable VRF, you create a support case. To enable service endpoints, you use the IBM Cloud CLI. For more information about how to enable your

account, see Enabling VRF and service endpoints.

Setting a private endpoint

After your account is enabled for VRF and service endpoints, you can add a private network endpoint to a service instance.

A service instance can have a private network endpoint, a public network endpoint, or both.

- Public: A service endpoint on the IBM Cloud public network.
- Private: A service endpoint that is accessible only on the IBM Cloud private network with no access from the public internet.
- Both public and private: Service endpoints that allow access over both networks.

Adding a private network endpoint

You add a private endpoint to a paid service instance from the service details page if you have a Manager or Writer service access role.

1. Go to your Resource list.
2. Click the name of a service instance that is on a paid plan. Lite plans do not support private network endpoints.
3. In the service details page, click the **Manage** tab.
4. Click **Add private network endpoint**.

Viewing your endpoint URL

The service endpoint URLs are different for private and public network endpoints. You can view the URL for an endpoint from the service details page.

1. Go to your Resource list.
2. Click the name of a service instance that has a private network endpoint.
3. In the service details page, click the **Manage** tab, and then click **Private Network Endpoint**.

What to do next

- Configure your account for VRF and Service endpoints.
- Modify your applications to use the new service endpoint URL.
- Read more about service endpoints.

Virtual Private Endpoints

IBM Cloud

IBM Cloud® Virtual Private Endpoints (VPE) for VPC enables you to connect to supported IBM Cloud® services from your VPC network by using the IP addresses of your choosing, allocated from a subnet within your VPC. See more details here.

Note: This document applies to watsonx Assistant, Discovery, Speech to Text, and Text to Speech. Virtual Private Endpoints (VPEs) are available for these services in the Dallas, Washington, Frankfurt, London, Sydney, and Tokyo locations.

Prerequisites

- Have a Virtual Private Cloud (VPC)
- Have private endpoints enabled for your service instance

Instructions

- Create a VPE Gateway (VPEG) through the UI, CLI, or API.
- Creation of the VPEG is confirmed when an IP address is set in the details view of the VPEG page in the UI, CLI output of the `endpoint-gateway` command, or API details call.

You can verify by running `nslookup <endpoint>` on the private service endpoint of the Watson service from your VPC, for example:

```
# nslookup api.private.us-south.assistant.watson.cloud.ibm.com
Server:   127.0.0.53
Address:  127.0.0.53#53

Non-authoritative answer:
Name:   api.private.us-south.assistant.watson.cloud.ibm.com
Address: 10.240.0.9   <---- your VPE IP address
```

To make requests using the assigned IP address instead of just the private service endpoint as suggested here, you must do your own hostname resolution. For example:

```
$ curl -X POST "https://api.private.us-south.assistant.watson.cloud.ibm.com/v2/assistants" --connect ::10.240.0.9
```

```
$ curl -X POST "https://api.private.us-south.assistant.watson.cloud.ibm.com/v2/assistants" --resolve api.private.us-
south.assistant.watson.cloud.ibm.com:443:10.240.0.9
```

Additional links

- IBM Cloud VPC
- VPE FAQ
- Accessing the VPE after setup
- Viewing Details of a VPE Gateway
- VPE Limitations
- Full VPC CLI reference

Backup and restore

Backing up and restoring data in Cloud Pak for Data

Use the following procedures to back up and restore data in your IBM Watson® Discovery for IBM Cloud Pak® for Data instance.

IBM Cloud Pak for Data

Note: This information applies only to installed deployments.

You use the same set of backup and restore scripts to back up and restore data in any of the supported upgrade paths. The backup script stores the version number of the service with data to back up from the existing deployment. The restore script detects the version of the service that is installed on the new IBM Cloud Pak for Data deployment, and then follows the appropriate steps to restore data to the detected version.

The following table lists the upgrade paths that are supported by the scripts.

Version in use	Version that you can upgrade to
4.7.0	4.7.x
4.6.x	4.7.x
4.5.x	4.7.x except 4.7.0
4.0.x	4.7.x except 4.7.0

Supported upgrade paths

If you are upgrading from 4.5.x to 4.7.x, a simpler way to complete the upgrade is described in the following topics:

- Upgrading Watson Discovery from Version 4.7.
- Upgrading Watson Discovery from Version 4.6.
- Upgrading Watson Discovery from Version 4.5.x.

If you use IBM Cloud Pak for Data Red Hat OpenShift APIs for Data Protection (OADP) backup and restore utility to back up and restore an entire cluster to 4.6, a few extra steps are required. For more information, see Using OADP to back up a cluster where Discovery is installed.

You can do an in-place upgrade from one 4.7.x version to a later 4.7.y version. For more information, see Upgrading Watson Discovery from Version 4.7.x to a later 4.7 refresh.

You can do an in-place upgrade from one 4.6.x version to a later 4.6.y version. For more information, see Upgrading Watson Discovery from Version 4.6.x to a later 4.6 refresh.

You can do an in-place upgrade from one 4.5.x version to a later 4.5.y version. For more information, see Upgrading Watson Discovery to the latest Version 4.5 refresh.

You can do an in-place upgrade from one 4.0.x version to a later 4.0.y version. For more information, see Upgrading Watson Discovery to a newer 4.0 refresh.

Process overview

At a high level, the process includes the following steps:

1. Back up your Discovery data by using the backup script.
2. Install the latest version of IBM Cloud Pak for Data.
3. Install the latest version of the Discovery service on the cluster.
4. Restore the backed-up Discovery data by using the restore script.

Back up and restore limitations

You cannot migrate the following data:

- Dictionary suggestions models. These models are created when you build a dictionary. The dictionary is included in the backup, but the term suggestions model is not. Reprocess the migrated collections to enable dictionary term suggestions.
- You cannot back up and restore curations or migrate them because curations are a beta feature.

You can back up and restore some data by using the backup and restore scripts, but you must back up and restore other data manually. The following data must be backed up manually:

- Local file system folders and documents that you can crawl by using the Local file system data source.

The following updates are made when your collections are restored:

- Any collection that contains documents that were created by uploading data are automatically recrawled and reindexed when restored. These documents are assigned new document ID numbers in the restored collections.
- Collections that were used in *Content Mining* projects are automatically recrawled and reindexed when restored. Only documents that are added by uploading data are assigned new document ID numbers in the restored collections.

Back up and restore methods

You can back up and restore your instance of Discovery manually or by using scripts.

- Using the backup scripts
- Using the restore scripts
- Backing up data manually
- Restoring data manually

You must have Administrative access to the Discovery instance on your Discovery cluster (where the data to be backed up is stored) and administrative access to the new instance (where the data will be restored to).

⚠ **Important:** The backup and restore scripts complete many operations and can take quite a bit of time to run. To avoid timeout issues, run a tool that prevents timeouts, such as `nohup`.

Using the backup scripts

Because changes to the data stored in IBM Watson® Discovery during a backup can cause the backup to become corrupted and unusable, no in-flight requests are allowed during the backup period.

An in-flight request is any IBM Watson® Discovery action that processes data, including the following actions:

- Source crawl (scheduled or unscheduled)
- Ingesting documents
- Training a trained query model

The amount of storage that is available in the node where you run the backup script must be 3 times as large as the largest backup file in the data store that you plan to back up. If your data store is large, consider using a persistent volume claim instead of relying on the node's ephemeral storage. For more information, see Configuring jobs to use PVC.

Complete the following steps to back up IBM Watson® Discovery data by using the backup scripts:

1. Enter the following command to set the current namespace where your Discovery instance is deployed:

   ```
   $ oc project <namespace>
   ```

2. Get the backup script from the GitHub repository.

   ⚠ **Important:** You need all of the files in the repository to complete a backup and restore. Follow the instructions in GitHub Help to clone or download a compressed file of the repository.

3. Make each script an executable file by running the following command:

   ```
   $ chmod +x <name-of-script>
   ```

   Replace `<name-of-script>` with the name of the script.

4. Run the `all-backup-restore.sh` script.

   ```
   $ ./all-backup-restore.sh backup [ -f backup_file_name ] [--pvc]
   ```

   The `-f backup_file_name` parameter is optional. The name `watson_discovery_<timestamp>.backup` is used if you don't specify a name.

   The `--pvc` parameter is optional. For more information about when to use it, see Configuring jobs to use PVC. By default, the backup and restore scripts create a `tmp` directory in the current directory that the script uses for extracting or compressing backup files.

   If you run into issues with the backup, rerun the backup command and include the `--use-job` parameter. This parameter instructs the backup script to use a Kubernetes job to back up ElasticSearch and MinIO in addition to Postgres, which uses a Kubernetes job by default. If the size of the data in ElasticSearch and MinIO is large and ephemeral storage is insufficient, include the `--pvc` option. When you do so, the script uses the

persistent volume claim that is specified with the `--pvc` option instead of the `emptyDir` ephemeral storage as the temporary working directory for the job.

Extracting files from the backup archive file

The scripts generate an archive file, including the backup files of the services that are listed in Step 1.

1. You can extract files from the archive file by running the following command:

```
$ tar xvf <backup_file_name>
```

Configuring jobs to use PVC

The backup and restore process uses Kubernetes jobs. The jobs use ephemeral volumes that use ephemeral storage. It is a temporary storage mount on the pod that uses local storage of a node. In rare cases, the ephemeral storage is not large enough. You can optionally instruct the job to mount a Persistent Volume Claim (PVC) on its pod to use for storing the backup data. To do so, specify the `--pvc` option when you run the script. The scripts use `emptyDir` of Kubernetes otherwise.

In most cases, you don't need to use a persistent volume. If you choose to use a persistent volume, the volume must be 3 times as large as the largest backup file in the data store. The size of the data store's backup file depends on usage. After you create a backup, you can extract files from the archive file to check the file sizes.

Also, you must have 2 times as much disk space available on the local system as the size of the data store because the archive of the data is split and then recombined to prevent issues that might otherwise occur when you copy large files from the cluster node to the local system.

Mapping multitenant clusters

When you restore data that was backed up from a version earlier than 4.0.6 to any later release and the backed-up deployment had more than one instance of the service provisioned, an extra step is required. You must create a JSON file that maps the service instance IDs between the backed-up cluster and the cluster where the data is being restored.

This mapping step is not required if the instance IDs did not change between the back up and restore steps. For example, you can skip this step if you are restoring data to the same cluster where it was backed up from or if you are restoring data to a brand new cluster that has no Discovery instances.

To create a mapping, complete the following steps:

1. Extract the mapping template file from the backup archive file.

```
$ tar xf <backup_file_name> tmp/instance_mapping.json -O > <mapping_file_name>
```

2. Make a list of the names and instance IDs of the service instances that are provisioned to the cluster where the data is being restored.

   The instance ID is part of the URL that is specified in the instance summary page. From the IBM Cloud Pak for Data web client main menu, expand Services, and then click Instances. Find your instance, and then click it to open its summary page. Scroll to the *Access information* section of the page, and look for the instance ID in the *URL* field.

   For example, `https://<host_name>/wd/<namespace>-wd/instances/<instance_id>/api`.

   Repeat this step to make a note of the instance ID for every instance that is provisioned.

3. Edit the mapping file.

   Add the instance IDs for the destination service instances that you listed in the previous step. The following snippet is an example of a mapping file.

```
{
   "instance_mappings":  [
      {
         "display_name":  "discovery-1",
         "source_instance_id":  "1644822491506334",
         "dest_instance_id":  "<new_instance_id>"
      },
      {
         "display_name":  "discovery-2",
         "source_instance_id":  "1644822552830325",
         "dest_instance_id":  "<new_instance_id>"
      }
   ]
}
```

When you run the restore script, include the optional `--mapping` parameter to apply this mapping file when the data is restored.

Using the restore scripts

⚠ **Important:** If you are restoring data from a version earlier than 4.0.6 and you are restoring a multitenant cluster to a multitenant cluster, you must take an extra step before you begin. For more information, see Mapping multitenant clusters.

Complete the following steps to restore data in IBM Watson® Discovery by using the restore scripts:

1. Enter the following command to set the current namespace where your Discovery instance is deployed:

   ```
   $ oc project <namespace>
   ```

2. If you haven't already, get the restore script from the GitHub repository.

   ⚠ **Important:** You need all of the files in the repository to complete a back up and restore. Follow the instructions in GitHub Help to clone or download a compressed file of the repository.

3. Make each script an executable file by running the following command:

   ```
   $ chmod +x <name-of-script>
   ```

   Replace `<name-of-script>` with the name of the script.

4. Restore the data from the backup file on your local system to the new Discovery deployment by running the following command:

   ```
   $ ./all-backup-restore.sh restore -f backup_file_name [--pvc] [--mapping]
   ```

   The `--pvc` parameter is optional. For more information about when to use it, see Configuring jobs to use PVC.

   The `--mapping` parameter is optional. For more information about when to use it, see Mapping multitenant clusters.

   By default, the backup and restore scripts create a `tmp` directory in the current directory that the script uses for extracting or compressing backup files. If you used the `--use-job` parameter when you backed up the data, specify it again when you restore the data. This parameter instructs the backup script to use a Kubernetes job to back up ElasticSearch and MinIO.

   The `gateway`, `ingestion`, `orchestrator`, `hadoop worker`, and `controller` pods automatically restart.

Using OADP to back up a cluster where Discovery is installed

If you plan to back up and restore an entire IBM Cloud Pak for Data instance by using the IBM Cloud Pak for Data Red Hat OpenShift APIs for Data Protection (OADP) backup and restore utility, you must do some additional steps in the right order for the utility to work properly when Discovery is present.

1. Run the Discovery backup script.
2. Use the OADP backup utility to back up the cluster.
3. Delete the project. This process removes the persistent volume claims and persistent volumes that are associated with Discovery.
4. Use the OADP backup utility to restore the cluster.
5. Uninstall Discovery, and then install Discovery again on the restored cluster.

   Note: A repeat of the installation is required because the utility does not always reinstall Discovery correctly.

6. Run the Discovery restore script to restore your data.

Backing up data manually

Manually back up data that is not backed up by using the scripts.

To manually back up your data from an instance of Discovery, complete the following steps:

1. Enter the following command to log on to your Discovery cluster:

   ```
   $ oc login https://<OpenShift administrative console URL> \
   -u <cluster administrator username> -p <password>
   ```

2. Enter the following command to switch to the proper namespace:

   ```
   $ oc project <discovery-install namespace>
   ```

3. Enter `oc get pods|grep crawler`.
4. Enter the following command:

```
$ oc cp <crawler pod>:/mnt <path-to-backup-directory>
```

Restoring data manually

Manually restore data that cannot be restored by using the script.

To manually restore your data from an instance of Discovery, complete the following steps:

1. Enter the following command to log on to your Discovery cluster:

```
$ oc login https://<OpenShift administrative console URL> \
-u <cluster administrator username> -p <password>
```

2. Enter the following command to switch to the proper namespace:

```
$ oc project <discovery-install namespace>
```

3. Enter `oc get pods|grep crawler`.
4. Enter the following command:

```
$ oc cp <path-to-backup-directory> <crawler pod>:/mnt
```

High availability and disaster recovery

IBM Watson® Discovery is highly available in all IBM Cloud® regions where Discovery is offered. However, recovering from potential disasters that affect an entire region requires planning and preparation.

IBM Cloud

> **Note:** This information applies only to managed deployments.

You are responsible for understanding your customization and usage of the service. You are also responsible for being ready to re-create an instance of the service in a new region and to restore your data in any region. See How do I ensure zero downtime? for more information.

High availability

IBM Watson® Discovery supports high availability with no single point of failure. The service achieves high availability automatically and transparently by using the multi-zone region (MZR) feature provided by IBM Cloud.

IBM Cloud enables multiple zones that do not share a single point of failure within a single location. It also provides automatic load balancing across the zones within a region.

Disaster recovery

Disaster recovery can become an issue if an IBM Cloud region experiences a significant failure that includes the potential loss of data. Because MZR is not available in all regions, wait for IBM to bring a region back online if it becomes unavailable. If underlying data services are compromised by the failure, also wait for IBM to restore those data services.

If a catastrophic failure occurs, IBM might not be able to recover data from database backups. In this case, you need to restore your data to return your service instance to its most recent state. You can restore the data to the same or to a different region.

Your disaster recovery plan includes knowing, preserving, and being prepared to restore all data that is maintained on IBM Cloud.

Backing up your data in Watson Discovery

There are several methods for backing up the data that is stored in IBM Watson® Discovery. Consider including these methods in your disaster recovery plan. Also, consider backing up the following data types:

- Data that you might want a copy of, such as source documents
- Data that Discovery stores and that you want to extract and back up

The following table shows the resources that you can download and re-upload to and from an instance.

Resource	Download/Re-upload from the UI	API support
Uploaded and crawled files		See note.
Relevancy training data		✓
Expansion list	✓	
Stop words list	✓	
Smart Document Understanding user-trained model	✓	
Smart Document Understanding pretrained model		
Text classifier enrichment	✓	
Dictionary enrichment	✓	
Entity extractor enrichment	✓	
Machine learning enrichment	✓	
Regular expression enrichment	✓	
Pattern enrichment	✓	
Advanced rules enrichment	✓	

Resource recovery support details

Note: You cannot subsequently download files that you add to Discovery because the original files are not stored in Discovery. However, you can retrieve information from the file that is stored in the collection index when the original file is processed. Use the Query API to submit a query that will return a passage from the file of interest, and then check the response body for data from the file. For example, for some file types, text from the original file is stored in the `text` field.

For information about resources that are created with the Content Mining application, see Content Mining resources.

Sentence classifier enrichments

To back up sentence classifier models that were created from the sentence labeling UI, download the models and store them locally. A model must be fully trained before it can be downloaded. For more information, see Downloading the sentence classifier model.

Ingested documents

Your uploaded documents are converted, enriched, and stored in the search index. If a disaster occurs, the search index is not recoverable. Store a backup of all your source documents in a safe place.

If you also import documents by doing scheduled crawls of external data sources, you might want to retain your data source credentials externally so that you can reestablish the connection to your data sources quickly. For the list of available sources and the credentials that are needed for each one, see Configuring IBM Cloud data sources.

You can get some of the text that was stored in the index when the original document was ingested by using the Query API. For more information, see Recovering documents.

Training data

Refer to this task to back up your training data queries and examples for a trained project. Training data is used for explicit training of your projects and is stored on a per project basis. To extract the training data, use the API to download the queries and the ratings from Discovery. To back up training data queries and examples, complete the following steps:

1. Download your training data by using the list training queries API.
2. Save your training queries and examples locally.

⚠ **Important:** The document IDs that you use in your training data point to the documents in your current project. Use the same IDs in your new projects to ensure that the correct documents are referenced. If the IDs do not match, your restored relevancy training will not work.

Expansion lists

If you are using synonyms (query expansions) for query modification, back up your .json expansion list, and store it locally. For more information, see Implementing synonyms.

Stopwords

In the case of stopwords, back up the text file. For more information about stopwords, see Defining stopwords.

Collection information

Tip: This is not required, but it is a best practice to retrieve the status for each collection regularly and store the information locally. By retaining these statistics, you can later verify that your restoration processes were successful if needed.

Smart Document Understanding models

If you use Smart Document Understanding (SDU), you have models that are associated with your configuration. To avoid loss of this information, export your models, back them up, and store them locally. SDU models have the file extension of `.sdumodel`.

Dictionary enrichments

1. Open your project, and click **Improve and customize**.
2. On the **Improvement tools** panel, click **Teach domain concepts** and then **Dictionaries**.
3. Click the download icon next to your dictionary. Your dictionary then downloads as a .csv file.

Regular expressions enrichments

Back up your regular expressions as a .csv file, and store them locally. Note the regular expressions that you specified to create your enrichments so that you can re-create the enrichments from them. For more information, see Regular expressions.

Machine learning enrichments

Back up your machine learning model .zip or .pear files, and store them locally. For more information, see Machine learning enrichments and Watson Explorer Content Analytics Studio models.

Pattern enrichments

1. Open your project.
2. On the *Improvement tools* panel of the *Improve and customize* page, click **Teach domain concepts** and then **Patterns (Beta)**.
3. Click the download icon next to your pattern. Your pattern model then downloads as a .zip file.

Advanced rules models enrichment

Back up your model files as .zip files, and store them locally. For more information, see Advanced rules models.

Classifier enrichments

Back up your classifier .csv files, and store them locally. For more information, see Classifier.

Entity extractor enrichments

To back up entity extractor models, download the models and store them locally. A model must be fully trained before it can be downloaded. For more information, see Exporting the entity extractor.

Content Mining application resources

You cannot back up certain data types and must manually re-create them. There are several Content Mining custom user resources that the application does not automatically back up. If data loss occurs, you must either manually re-create the following custom user resources in the Content Mining application or upload a locally saved file that contains the resource:

- Custom map
- Searched document export: You can export a searched document in the **Documents** view in the Content Mining application, but you cannot reupload it in the application.

- Facet analysis result export: You can download the results of your facet analysis by clicking the **Export** icon, then **Export results**, and **Export** in the **Analysis export options** dialog box.
- Collection: You can restore a Content Mining collection if you stored the collection locally as a .csv file and then upload it in the application. Otherwise, you must manually re-create the collection.
- Document classifier: You can restore a document classifier if you stored the document classifier locally as a .csv file and then upload it in the application. Otherwise, you must manually re-create the document classifier.
- Custom annotators
    - Dictionary: You can restore a dictionary in the application if you stored the dictionary locally as a .csv file and upload it in the application.
    - Regular expressions: You can restore a regular expression in the application if you stored the regular expression locally as a .csv file and upload it in the application.
    - Machine learning models: You can restore a machine learning model if you stored the model locally as a .zip file and then upload it in the application.
    - PEAR File: You can upload a .pear file if you stored the file locally and then upload it in the application.

You cannot back up the following resources locally and must recreate them in the Content Mining application.

- Saved analysis
- Report
- Dashboard

Restoring your data to a new Watson Discovery instance

Note: Consider using your backups to restore to a new Discovery instance in a different data center, also known as a region or location.

To begin restoration, first start by reviewing your list of collections and associated data sources, as well as your file backups.

- Create your projects and collections. Use the Discovery tooling, or the API. See Create a project and Create a collection.
- Add back stopwords into the collections. See Defining stopwords.
- If you use custom query expansion, add your query expansions. See Implementing synonyms.
- If you use any custom entity models from IBM Watson® Knowledge Studio for enrichment, reimport that model into your Discovery instance. For details, see Managing enrichments.

After you set up your projects and collections as they were before, begin ingesting your source documents. Depending upon how you ingested your documents previously, you can do so by using your own solution or one of the following methods:

- The API
- A connector

Restoring sentence classifiers

To restore a sentence classifier model that was created from the sentence labeling UI, import the exported .sc file to create a new machine learning model. You cannot open the exported model in the sentence labeling UI to continue working with it. However, you can import a finished model and apply it to collections as a sentence classifier enrichment.

For more information about how to import a sentence classifier model to create a machine learning enrichment, see Use imported ML models to find custom terms.

Restoring training data

After you restore your projects, you can begin the process of re-creating your relevancy training models. To restore your training data queries and examples, re-create your individual training queries and the examples by using the create training query API, or you can restore your queries and examples on Discovery. For more information about restoring your training data by using Discovery, see the instructions for accessing the **Train** page in Improving result relevance with training.

Important: For the restore to work properly, note that the document IDs that you use in your training data point to the documents in your current project. Use the same IDs in your new projects to ensure that the correct documents are referenced. If the IDs do not match, your restored relevancy training will not work.

Restoring connections to external data sources

In case of an unanticipated loss of data, you might lose your scheduled crawls of external data sources. See Configuring IBM Cloud data sources for the list of available sources.

To restore your external data, reestablish your connections to these data sources, and then recrawl them.

To find the data source credentials that you stored, follow the instructions for your chosen data source in Configuring IBM Cloud data sources. These instructions explain how you can reconnect to your data sources and get the data imported into Discovery.

Restoring Smart Document Understanding models

To import a previously exported Smart Document Understanding (SDU) model, see Importing and exporting models. SDU models have the file extension of `.sdumodel`.

When importing an SDU existing model into a new collection, it is a best practice to create the new collection and add one document, then import the model and upload the remainder of your documents.

Restoring dictionary enrichments

1. Open your project.
2. On the *Improvement tools* panel of the *Improve and customize* page, click **Teach domain concepts**, **Dictionaries**, and then **Upload**.
3. In the **Apply dictionary** dialog box, enter a name for your .csv file, select a language, specify the facet path, click **Upload**, and select your dictionary .csv file.
4. Click **Create**.

Note: After you upload dictionary .csv files for recovery, you cannot use the dictionary editor to further edit the terms. If you want to use the dictionary editor, create a dictionary, and manually add the dictionary terms.

For information about uploading a dictionary enrichment .csv file, see Dictionary.

Restoring pattern enrichments

You can restore pattern enrichment .zip files as advanced rules models .zip files by completing the following steps:

1. Open your project.
2. On the *Improvement tools* panel of the *Improve and customize* page, click **Teach domain concepts**, **Advanced rules models**, and then **Upload**.
3. In the **Apply advanced rules model** dialog box, enter a name for your .zip file, select a language, specify a result field, click **Upload**, and select your advanced rules models .zip file.
4. Click **Create**.

Note: After you upload pattern model .zip files for recovery, you cannot use the pattern editor to further edit the .zip files.

For more information about uploading an advanced rules models .zip file, see Advanced rules models.

Restoring entity extractors

To restore an entity extractor model, import the exported .ent file to create a new machine learning model. You cannot open the exported model in the entity extractor tool to continue working with it. However, you can import a finished model and apply it to collections as an entity extractor enrichment.

For more information about how to import an entity extractor model to create a machine learning enrichment, see Use imported ML models to find custom terms.

Deleting a service instance

The procedure that you follow to delete a service instance differs depending on how the instance was deployed.

Deleting a managed service instance IBM Cloud

When you delete a Discovery service instance, the data associated with your service is also removed.

Only the owner or someone with Administrator role-level access to the service instance can delete it.

To delete your Discovery service instance, complete the following steps:

1. From the IBM Cloud Resource list, expand the **AI/Machine Learning** section.
2. Find the service instance that you want to delete.
3. Click the *Actions* menu icon, and then select **Delete**.

The data that is associated with the service instance is retained for 7 days. After the 7-day retention period, it is deleted. If a service instance is deleted by mistake, you can restore it as long as you do so before the 7-day window closes. For more information, see Using resource reclamations in the IBM Cloud documentation.

For more information about how data is handled by IBM Cloud, see the following information:

- Data Processing and Protection Datasheet.
- IBM Cloud Data security and privacy terms

For more information about how to delete a deployment of the service from Cloud Pak for Data as a Service, see Deleting a deployment.

Deleting an installed service instance IBM Cloud Pak for Data

Only the owner or an Administrator of the provisioned service instance can delete it.

To deprovision a service instance of Discovery for IBM Cloud Pak for Data, complete the following steps:

1. From the IBM Cloud Pak for Data web client menu, click **Services > Instances**.
2. Find your service instance, and then click the menu icon and choose **Delete**.

For more information about how to uninstall the service, see Uninstalling Discovery.

FAQ

Find answers to frequently asked questions.

For more information about Discovery-specific concepts, such as *projects* or *enrichments*, see the glossary.

How do I search the product documentation?

To search the entire IBM Cloud Docs site, enter your search term into the search field in the IBM Cloud website banner. To search for information about the Discovery service only, scroll to the start of the page and enter your search term into the search field in the page header.

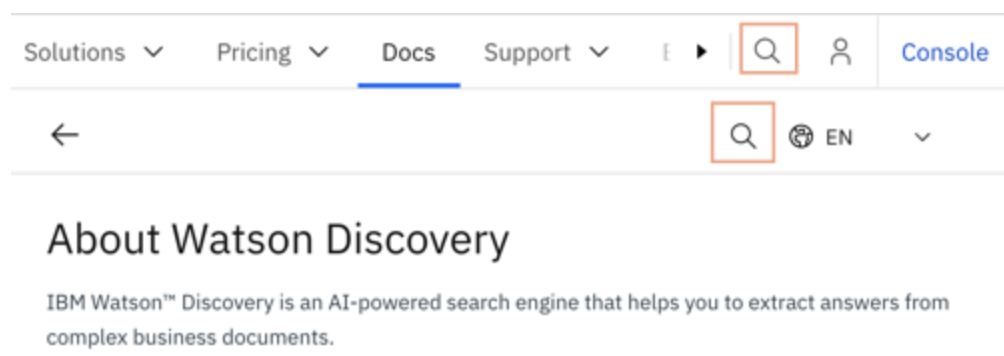


Figure 1. Search bar for this product documentation versus all IBM Cloud docs

How does Watson Discovery access my data?

Discovery has built-in connectors that can crawl various data sources, including websites, IBM Cloud Object Storage, Box, Microsoft SharePoint, and Salesforce sites. It even has support for you to build custom connectors. You can schedule crawls so that as the source data changes, the latest version is picked up by your collection automatically. Discovery only ever reads from external data sources; it never writes, updates, or deletes any content in the original data source. For more information, see Creating collections

Can I upload documents?

Yes, you can upload documents directly to a collection in your project. An upload is a one-time operation that you can use to get started. An alternative approach is to connect to a data source and crawl the source for information. When you crawl data sources, the data can stay where it is and you can set up a schedule by which to crawl the external source to find new and changed information. When you crawl the data, you know that the information in your collection is always up to date. For more information, see Creating collections.

Must all my documents be English?

No. Discovery support multiple languages. For more information about language support per feature, see Language support.

What types of files can Discovery ingest?

Discovery can ingest most standard business file types, including PDF, Microsoft Word documents, spreadsheets, and presentations. For a complete list, see Supported file types.

How do I know whether I have Discovery v1 or v2?

If you're using Discovery on IBM Cloud Pak® for Data, then you're using Discovery v2.

If you have a service instance that is managed by IBM Cloud, then check what you see when you launch the product. When you open the product user interface in v2, the following page is displayed:

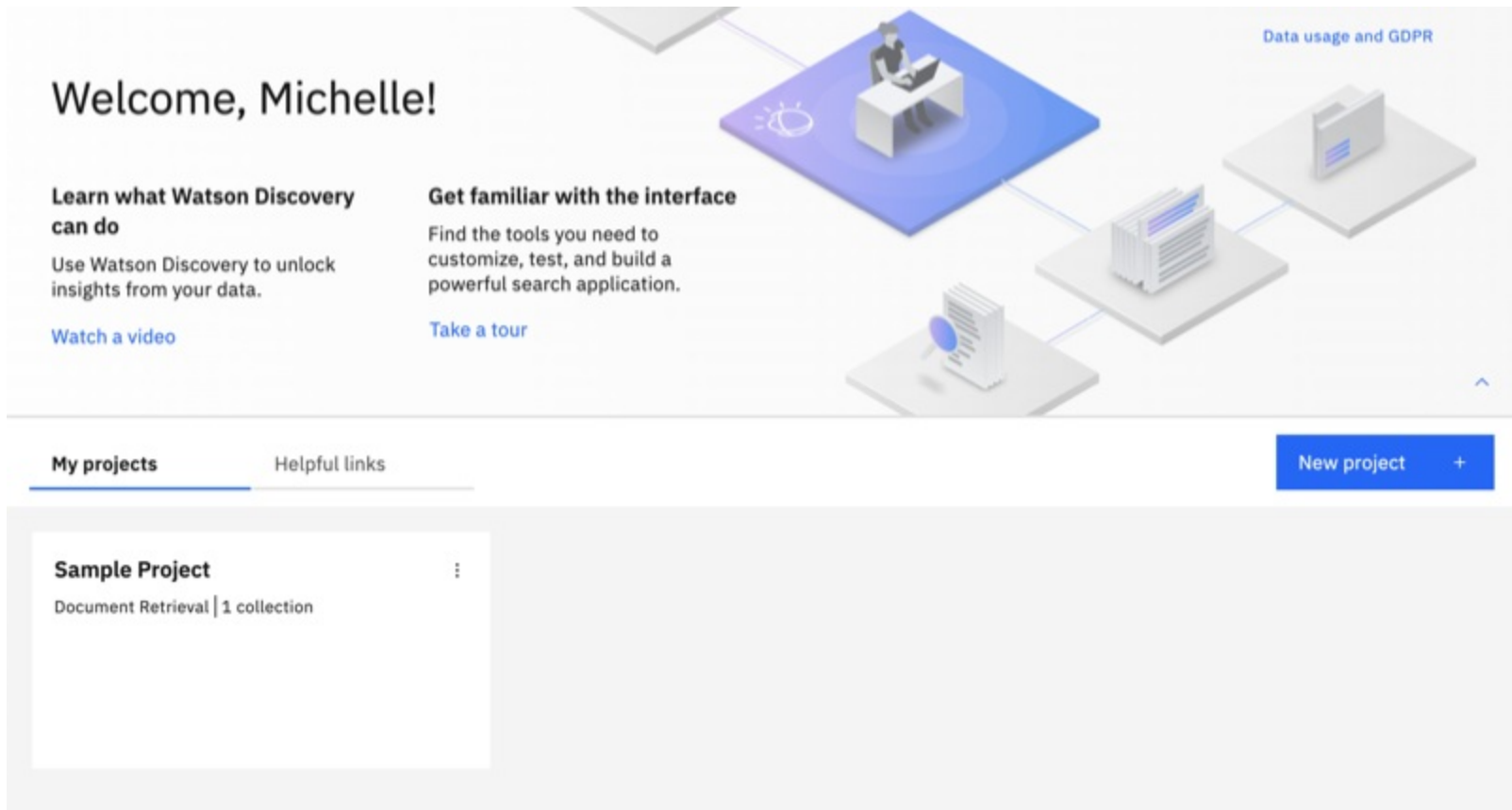


Figure 2. Discovery v2 home page

Can I integrate Watson Discovery with Watson Assistant?

You can integrate Discovery and watsonx Assistant to make information that is stored in external data sources available to a virtual assistant. Create a *Conversational Search* project in Discovery, and then add the data sources that you want to make available to it. Next, create a *search integration* in watsonx Assistant, and connect it to your Discovery project and collection.

Can I increase the collection limit for a project?

If you want to add more than 5 collections to your project and you have a Premium plan, you can request an increase to the collection limit by opening a support request. For more information, see Getting help.

Can I find related documents after I add them to a collection

If you want to retain information about the relationship of two or more documents to one another, you can do so. For example, if 3 documents are uploaded from the same folder and their placement in the folder is significant to their meaning, you might want to retain the parent folder information.

When you upload a document, no such information about its relationships to other documents is stored by default. To add the information, you can use the API to add the documents. When you add documents by using the API, you can specify metadata values. You might want to specify a metadata value, such as `"foldername": "company_a"`, for each document.

Alternatively, you can copy the document body of each document into a JSON file, where each document is an object in a single array. When the JSON file is ingested, each item in the array is added as a separate document with a seprarate document ID. Each document shares the same parent ID, which identifies the relationship between them.

You can quickly find documents that share the same parent ID or other common metadata value from the *Manage data* page. Customize the view to show the field, such as `extracted_metadata.parent_document_id` or `extracted_metadata.foldername`, that the documents share in common.

Can I customize Discovery to understand my data

Yes. Use the intuitive tools provided with the product to teach Discovery about the unique terminology of your domain. For example, you can teach it to recognize patterns, such as BOM or part numbers that you use, or add dictionaries that recognize your product names and other industry-specific word meanings. For more information, see Adding domain-specific resources.

How does the Smart Document Understanding tool work?

You can use the Smart Document Understanding tool to teach Discovery about sections in your documents with distinct format and structure that you want Discovery to index. You can define a new field, and then annotate documents to train Discovery to understand what type of information is typically stored in the field. For more information, see Using Smart Document Understanding.

What's the best way to add synonyms?

You can use two different methods to define synonyms.

- To define synonyms that are recognized and tagged when a document is ingested and that can be retrieved by search, create a dictionary and add

synonyms for the dictionary term entry. A dictionary defines special terms that you want to tag in your documents, such as product names or industry-specific terminology. You can use the dictionary terms later to create facets and to filter documents. For more information, see Dictionary.

- To define synonyms that are applied to the query text that is submitted by users to expand the meaning of the query, add synonyms by using the Synonyms tool on the *Improve relevance* section of the *Improve and customize* page. For more information, see Expanding the meaning of queries.

Can I use Discovery to detect sentiment?

You can use Discovery to detect both phrase and document sentiment. Document sentiment is a built-in Natural Language Processing enrichment that is available for all project types. Document sentiment evaluates the overall sentiment that is expressed in a document to determine whether it is positive, neutral, or negative. Phrase sentiment does the same. However, phrase sentiment can detect and assess multiple opinions in a single document and, in English and Japanese documents, can find specific phrases. For more information about document sentiment, see Sentiment. For more information about phrase sentiment, see Detecting phrases that express sentiment. You cannot detect the sentiment of entities or keywords in v2.

What is a nested field?

When you ingest a file or crawl an external data source, the data that you add to Discovery is processed and added to the collection as a document. Fields from the original file are converted to document fields and are added to the collection's index. Some content is added to root-level index fields and some information is stored in nested fields. Where data gets stored differs by file type. Most of the fields from structured data sources are stored as root-level fields. For files with unstructured data, much of the body of the file is stored in the `text` field in the index. Other information, such as the file name, is stored in nested fields with names like `extracted_metadata.filename`. You can determine whether a field is a nested field by its name. If the field name includes a period, it is a nested field. For more information about how different file types are handled, see How your data source is processed.

Which type of query should I use in my custom app?

When you submit a query, you can choose to submit a natural language query or use the Discovery Query Language to customize the search to target specific fields in the index, for example. For more information about the different types of queries and how to decide which one to use, see Choosing the right query type.

Getting help

Get help to solve issues that you encounter when you use the product.

Use these resources to get answers to your questions:

- Walk through a guided tour to learn about a project type or a feature. Click **Guided tours** from the page header to see a list of available tours.
- For answers to frequently asked questions, see the FAQ.
- Find answers to common questions or ask questions where experts and other community members can answer. Go to the Watson Discovery Community forum.

IBM Cloud **Contacting IBM Cloud Support for managed deployments**

Managed deployments are deployments that are hosted on IBM Cloud, including IBM Cloud Pak for Data as a Service deployments.

If your service plan covers it, you can get help by opening a case from IBM Cloud Support.

Be ready to share the following information with IBM Support:

Account information

- Account name or customer name.
- Business impact so IBM Support understands the urgency of the issue and can prioritize it.
- Case information for any related cases or a parent case.
- Cloud location where the service instance is hosted (Dallas, Frankfurt, and so on).
- Your service plan (Plus, Premium and so on).

Problem description

- What outcome were you expecting and what happened?
- Message text that is displayed when the error occurs, especially the document ID, if specified.
- Steps to take to reproduce the issue.
- Any screen captures that illustrate the problem.
- When did the problem occur?
- Instance ID. (The instance ID is part of the URL that is specified in the *Credentials* section of the service page on IBM Cloud. You can copy the full URL and provide that.)
- Collect and share the HTTP archive (HAR) file from your browser. The HAR file contains a log of trace information from within a browser session. It records web requests that are made by the browser to the website page including request and response headers, the body, and the time it takes to load the assets.
- If you are using the API, share example API calls, including the version parameter value that was specified, and the API response body.

  Note: Do not share code examples. IBM Support cannot debug custom code.

- If the problem is related to a particular project or collection, provide the project ID and collection ID.
  - Project ID. (You can copy the Project ID from the *API Information* tab of the *Integrate and deploy* page in the product user interface.)
  - Collection ID, if you were able to create a collection. (To get the ID, open the *Manage collections* page, and then click the collection to open it. From the web browser location field, scroll to the end of the URL. Look for the `collections/` section, and then copy the ID that is displayed after it. For example, in the URL `/collections/5a525eb7-b175-3820-0000-017d00f0fcd1/activity`, the collection ID is `5a525eb7-b175-3820-0000-017d00f0fcd1`.)
- If the problem has to do with documents failing to load, provide the following information if known:
  - What kind of documents are being uploaded (such as PDF, Json, CSV). Was optical character recognition (OCR) enabled for the collection?
  - How were the documents loaded into the collection? (using the API, from the product UI, data source connector)
  - Did you identify fields in the collection by using Smart Document Understanding? If so, what type of SDU model was applied to the collection (user-trained or pretrained)?
  - What enrichments were applied to the collection?
- If the problem is related to a particular document (of a small set of documents), provide the `document_id` of the document, if known. You can share example documents if they might be helpful.
- If the problem is related to querying documents, describe the kind of query being used.

IBM Cloud Pak for Data **Contacting IBM Support for installed deployments**

Installed deployments are deployment that you provision on IBM Cloud Pak for Data.

You can get help by opening a case from IBM Support from IBM Support.

Be ready to share the following information with IBM Support:

Account information

- Account name or customer name.
- Business impact so IBM Support understands the urgency of the issue and can prioritize it.
- Case information for any related cases or a parent case.
- Software versions of both the Discovery service version and IBM Cloud Pak for Data version.
- Relevant details about configuration choices that were made during installation and deployment.

Problem description

- What outcome were you expecting and what happened?
- Message text that is displayed when the error occurs, especially the document ID, if specified.
- Steps to take to reproduce the issue.
- Any screen captures that illustrate the problem.
- When did the problem occur?
- Instance ID. (From the IBM Cloud Pak for Data web client main menu, expand *Services*, and then click *Instances*. Find your instance, and open its summary page. Scroll to the *Access information* section of the page, and then copy the URL. The instance ID is part of the URL. You can provide the full URL to IBM Support.)
- If you are using the API, share example API calls, including the version parameter value that was specified, and the API response body.

  > **Note:** Do not share code examples. IBM Support cannot debug custom code.

- Relevant logs, including the Red Hat OpenShift collector logs.

  The IBM Support representative can share a script with you that collects relevant logs from your cluster.

- If the problem is related to a particular project or collection, provide the project ID and collection ID.
  - Project ID. (You can copy the Project ID from the *API Information* tab of the *Integrate and deploy* page in the product user interface.)
  - Collection ID, if you were able to create a collection. (To get the ID, open the *Manage collections* page, and then click the collection to open it. From the web browser location field, scroll to the end of the URL. Look for the `collections/` section, and then copy the ID that is displayed after it. For example, in the URL `/collections/5a525eb7-b175-3820-0000-017d00f0fcd1/activity`, the collection ID is `5a525eb7-b175-3820-0000-017d00f0fcd1`.)
- If the problem has to do with documents failing to load, provide the following information if known:
  - What kind of documents are being uploaded (such as PDF, Json, CSV). Was optical character recognition (OCR) enabled for the collection?
  - How were the documents loaded into the collection? (using the API, from the product UI, data source connector)
  - Did you identify fields in the collection by using Smart Document Understanding? If so, what type of SDU model was applied to the collection (user-trained or pretrained)?
  - What enrichments were applied to the collection?
- If the problem is related to a particular document (of a small set of documents), provide the `document_id` of the document, if known. You can share example documents if they might be helpful.
- If the problem is related to querying documents, describe the kind of query being used.

Feedback

We value your opinion and want to hear it.

Product feedback

How you share product feedback differs depending on the type of deployment you are using.

IBM Cloud

From the page header, click **Share feedback** to open a simple feedback form. You can share your opinion and submit it to the product team for consideration. We appreciate your insights and suggestions.

IBM Cloud Pak for Data

To share ideas or suggest new features for the Discovery service on IBM Cloud Pak for Data, go to the IBM Data and AI Ideas Portal for Customers

Product documentation feedback

Give us feedback about the product documentation. A **Feedback** button is displayed along the edge of each page. Click it to open a form where you can rate the current topic and share a comment.

Plan information

Discovery pricing plans

Learn more about the IBM Watson® Discovery service plans, so you can pick the plan type that best meets your needs.

Plus

Find targeted answers and insights across many document types.

The first Plus plan that is created includes a 30-day trial at no cost. You must pay for each additional Plus plan (and for use of the first plan after the 30-day trial ends).

⚠ **Important:** If you continue to use a Plus plan service instance after the 30-day trial ends, you are charged for it. To avoid being charged after 30 days, you must delete the Plus plan service instance.

What's included

For subscription accounts, the monthly fee per instance is $500.

The plan includes the following features:

- 10,000 documents per month ($50 for every additional 1,000 documents per month)
- 10,000 queries per month ($20 for every additional 1,000 queries per month)
- Up to 5 queries per second
- Optical Character Recognition(OCR)
- Out of the box data source connectors
- Table Retrieval
- Custom NLP models
- Custom Relevance Model
- Entity extractor

Artifact limits

- Up to 500,000 queries per month
- Up to 500,000 documents
- Up to 20 projects
- Up to 40 collections (Up to 5 collections per project)
- Up to 10 MB document size
- Up to 3 custom models Learn more
- Up to 40 custom fields for Smart Document Understanding model and model import and export
- Up to 20 custom dictionaries
- Up to 20 custom pattern extraction models
- Up to 20 custom regular expression models
- Up to 20 custom text classification models

Enterprise

Scale and secure your Discovery application with enterprise-grade support and performance, and address more use cases including contract analysis and content mining to explore insights across documents.

What's included

For subscription accounts, the monthly fee per instance is $5,000.

The plan includes the following features:

- 100,000 documents per month ($5 for every additional 1,000 documents per month)
- 100,000 query or analyze API calls per month ($5 for every additional 1,000 calls per month)
- Up to 3 custom models Learn more
- Up to 10 entity extractor models
- Up to 5 queries per second
- Everything that's available in Plus
- Analyze API
- Content Intelligence
- Content Mining
- Document classification (Text classification is available in all plans. Document classification is available with Enterprise and higher-level plans only.)

Note: Your bill labels requests that are generated from both query searches and analyze API calls as "Queries".

Artifact limits

- Unlimited queries per month
- Unlimited documents
- Up to 100 projects
- Up to 300 collections (Up to 5 collections for all project types except Content Mining, which supports 1)
- Up to 10 MB document size
- Up to 10 custom Knowledge Studio models
- Up to 10 entity extractor models
- Up to 100 custom fields for Smart Document Understanding model and model import and export
- Up to 100 custom dictionaries
- Up to 100 custom pattern extraction models
- Up to 100 custom regular expression models
- Up to 20 custom text classification models
- Up to 20 custom document classification models

Note: Autocompletion, curation, and notice requests are not billed in any plan type.

How is document pricing calculated?

IBM uses the maximum number of documents per day and then prorates the document price to calculate the cost for a month.

For example, imagine that 300,000 documents are added to a collection in the first 6 days (6/30) of the month, and then another 1 million documents are added over a two-day span. In the first day (1/30) of the two-day span, 700,000 documents are added. All of the documents (1 million + 300,000 = 1,300,000) remain in the collection index for the rest of the month (23/30). The number of documents for the month might be calculated by using an equation like this:

```
300K * (6/30) + 700K * (1/30) + 1300K * (23/30)
```

Document pricing counts the number of indexed documents in each collection. If you reuse data in a second collection, it generates a second set of documents in the index, which are counted separately.

For more information about what counts as a document, see Document limits.

What is a custom model?

Custom models include any of the following model types:

- Discovery entity extractor
- Knowledge Studio machine learning
- Knowledge Studio rule-based

Note: Advanced rules models are not counted as custom models.

Discovery entity extractor models that are trained and published as an enrichment count toward the model limit for the plan. The entity extractor enrichment is what incurs charges, not the workspace or model. The entity extractor enrichment incurs charges whether or not it is applied to a collection.

Premium

Premium plans offer developers and organizations a single tenant instance of one or more Watson services for better isolation and security. These plans offer compute-level isolation on the existing shared platform, as well as end-to-end encrypted data while in transit and at rest.

For more information, or to purchase a Premium plan, contact Sales.

Other plans

Features and limitations are similar between instances that you deploy on installed deployments in IBM Cloud Pak for Data and Premium plan instances that are managed by IBM Cloud.

Additional information

- For more information about how queries are counted, see Query limits.
- For more information about pricing or to create a service instance, see the IBM Cloud catalog.

IBM Cloud resources:

- How you're charged.
- IBM Cloud Cost estimator
- IBM Cloud Services terms
- For more information about IBM Cloud security, see IBM terms.

Upgrading

Learn how to upgrade your service plan.

IBM Cloud **IBM Cloud only**

> **Note:** This information applies only to managed deployments. For more information about upgrading an installed deployment that is hosted by IBM Cloud Pak for Data, see Upgrading the service.

Upgrading your plan

You can explore the Discovery service plan options to decide which plan is best for you.

The page header shows the plan you are using today.

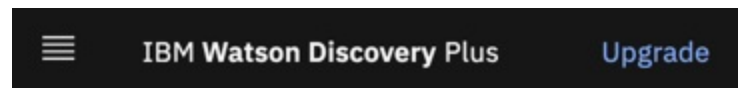


Figure 1. Plus plan is displayed in the page header

You cannot upgrade from any v1 plan to any v2 plan. For example, you cannot upgrade a Lite plan to a Plus, Enterprise, or Premium (v2) plan. And you cannot upgrade an Advanced, Partner, Standard, or Premium (v1) plan to an Enterprise or Premium (v2) plan. To start using v2, create a new Plus, Enterprise, or Premium plan.

For information about upgrading from a Lite to an Advanced v1 plan, see Upgrading your service in the v1 documentation.

Even though you can use the Plus plan for the first 30 days at no charge, you must have a paid account to create a Plus plan. For more information about creating a paid account, see Upgrading your account.

1. How you upgrade depends on your plan.

    - If you decide you want to keep the Plus plan after using the 30-day free trial, no action is required.

      After 30 days of using the Plus plan at no cost, you are charged for it.

    - If you decide you do *not* want to continue using the Plus plan, delete the Plus plan service instance before the 30-day trial period ends. You can delete the service instance from the IBM Cloud Resource list.

      The number of days that are left in your trial is displayed in the page header.

    - To upgrade a Plus plan to an Enterprise plan, complete the following steps:

        - Open the service page for your Plus plan service instance from the IBM Cloud Resource list.
        - Click *Upgrade*.
        - Choose the Enterprise plan, and then click *Save*.
        - Give the upgrade process time to finish.

          The time it takes to convert the plan varies depending on the amount of data in your existing Plus plan service instance. It takes at least 20 minutes and, for instances with large amounts of data, can take more than a day to complete. The IBM Cloud page does not show progress information and doesn't indicate when the plan upgrade process is finished. To check whether the new plan is in effect, you must refresh the service instance overview page, and then check for the new plan name to be displayed in the *Plan* tile.

          During the plan upgrade process, you can continue to submit search queries in your existing projects. However, avoid the following actions:

            - Adding new projects or collections
            - Deleting or changing existing collections, including adding documents, editing fields, and changing enrichment settings.

    - If you are creating an Enterprise plan in the same data center location where you have an existing Premium plan, you must create a new resource group for the new plan. You cannot use the same resource group for Enterprise and Premium plans that are hosted in the same location. For more information, see Managing resource groups.

    - You cannot do an in-place upgrade from a Plus or Enterprise plan to a Premium plan.

      A Premium plan instance must be provisioned for you. To start the process, contact Sales. You will be asked to provide the following details:

- Customer name
- Customer email
- Planned deployment date
- Data center location, such as Dallas or Frankfurt
- Account ID
- Resource group name
- Resource group ID

  The resource group is created by the account holder. For more information, see Managing resource groups.

> ⚠ **Important:** You cannot directly downgrade from one plan to another. If you want to move from an Enterprise plan to a Plus plan, for example, you must provision a new Plus plan and then move data to it from your existing Enterprise plan. After the data is moved, you can delete the Enterprise plan. For more information about how to back up data that you want to move between service instances, see High availability and disaster recovery.

For more information about plans, see Discovery pricing plans.

Accessibility

IBM strives to provide products with usable access for everyone, regardless of age or ability.

You can interact with all functions of the IBM Watson® Discovery content by using only the keyboard.

For more information about the accessibility compliance of the product, go to the Product Accessibility Conformance Reports website, and then search for IBM Watson® Discovery.

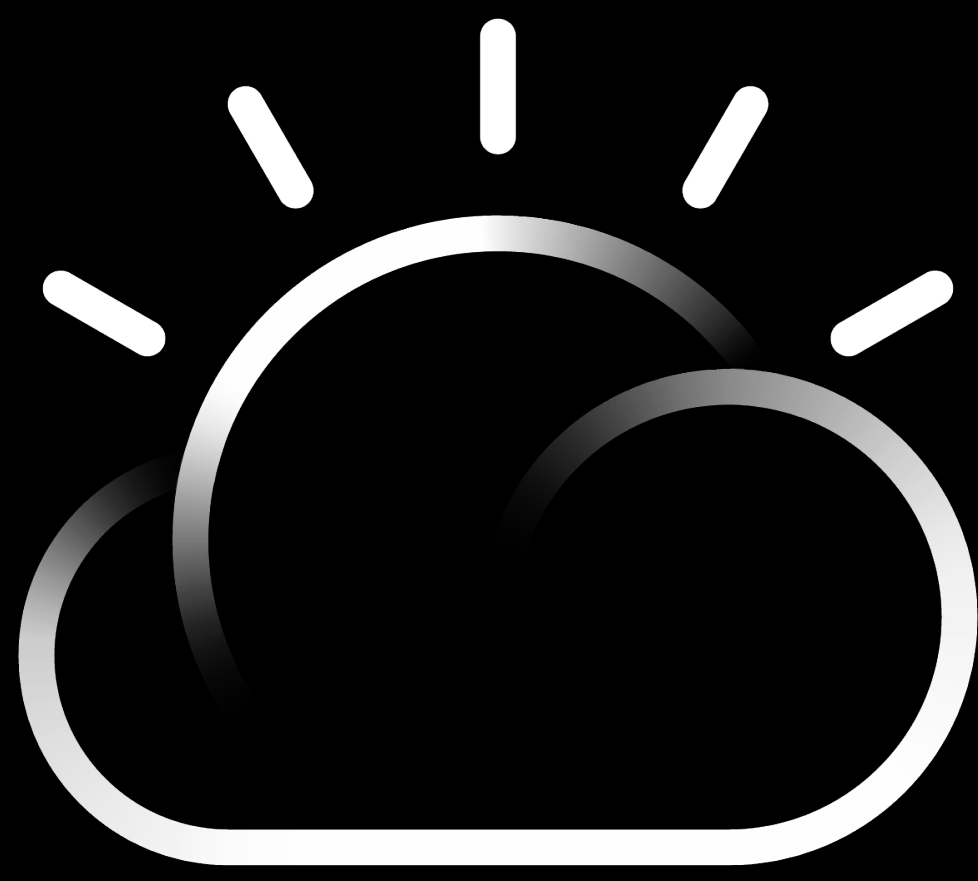
Accessibility features in the product documentation

Accessibility features help people with a physical disability, such as restricted mobility or limited vision, or with other special needs, to use information technology products successfully.

The accessibility features in this product documentation allow users to do the following things:

- Use screen-reader software and digital speech synthesizers to hear what is displayed on the screen. Consult the product documentation of the assistive technology for details on using assistive technologies with HTML-based information.
- Use screen magnifiers to magnify what is displayed on the screen.
- Operate specific or equivalent features by using only the keyboard.

The documentation content is published in the IBM Cloud Docs site. For more information about the accessibility of the site, see Accessibility features for IBM Cloud.



IBM **Cloud**