Lucidworks

LAB: Building a Query Intent Classifier

Prerequisites

- Register for Kaggle
  - Instructions here: Registering for Kaggle

- LAB: Spark and Data Prep

Recommended

- LAB: Working with Signals

Inspecting Labs_signals

Lucidworks

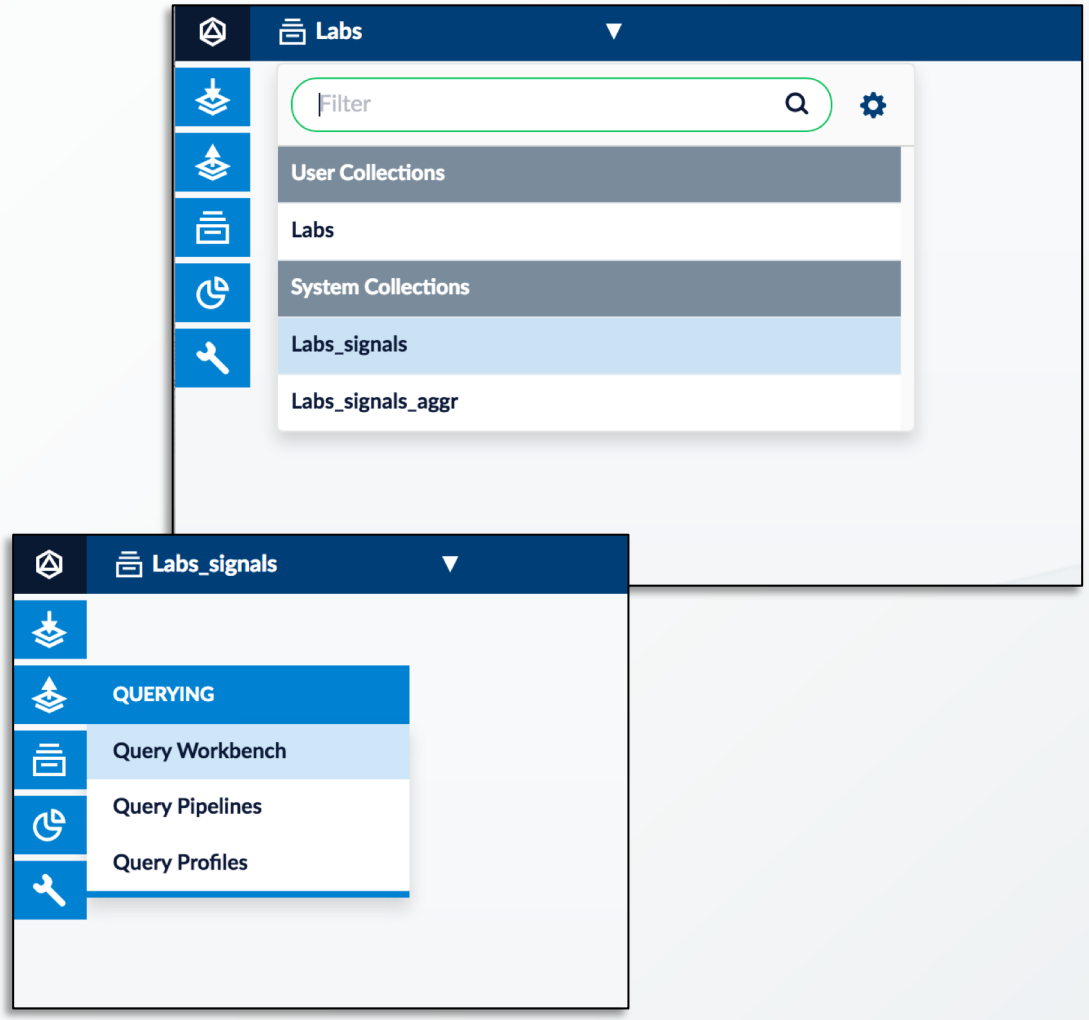
This lab assumes that you are using an AWS virtual machine provided by Lucidworks Training. If this is not the case, your filepaths and IP addresses will vary significantly from those shown.

- In a bash/shell terminal, start Fusion
  `./fusion/4.0.1/bin/fusion start`

- In a web browser, open Fusion Admin
  `<your-vm-ip>:8764`

Lucidworks

- Enter your username and password
  *The default is **admin** and **Lucidworks1***
- Click into the **Labs** Fusion App
- In the top left dropdown, change to the **Labs_signals** collection
- In the left side menu, go to **QUERYING > Query Workbench**

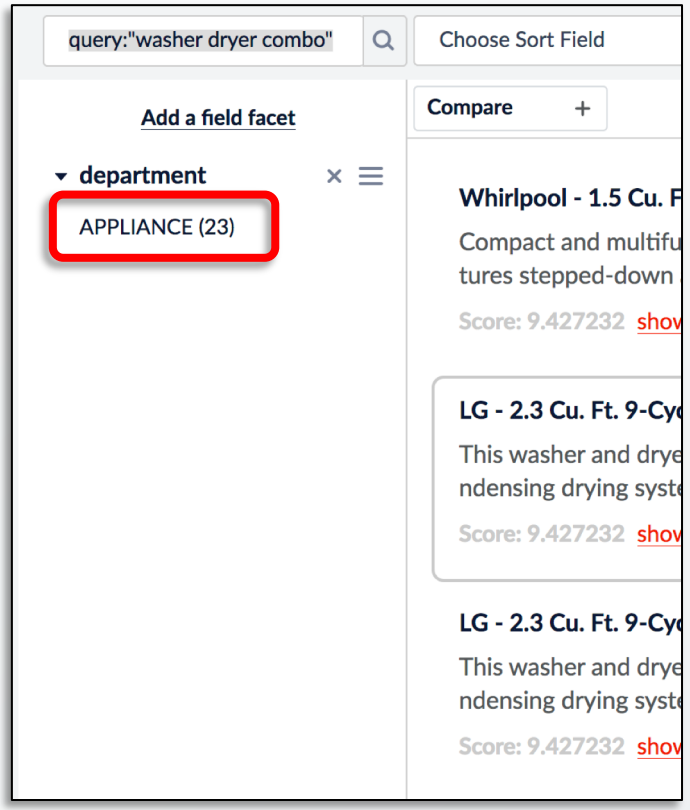


Inspecting the Signals

Lucidworks

In this lab, we will build a classifier that predicts which store department a user is interested in based on their input query. Doing so involves finding correlations between certain query terms and the store department the user ultimately clicks into.

- Execute the query **query:"washer dryer combo"**



*Unsurprisingly, everyone who searched for **"washer dryer combo"** ultimately clicked on an item from the **APPLIANCE** department.*

Not every query is so clear-cut, however…

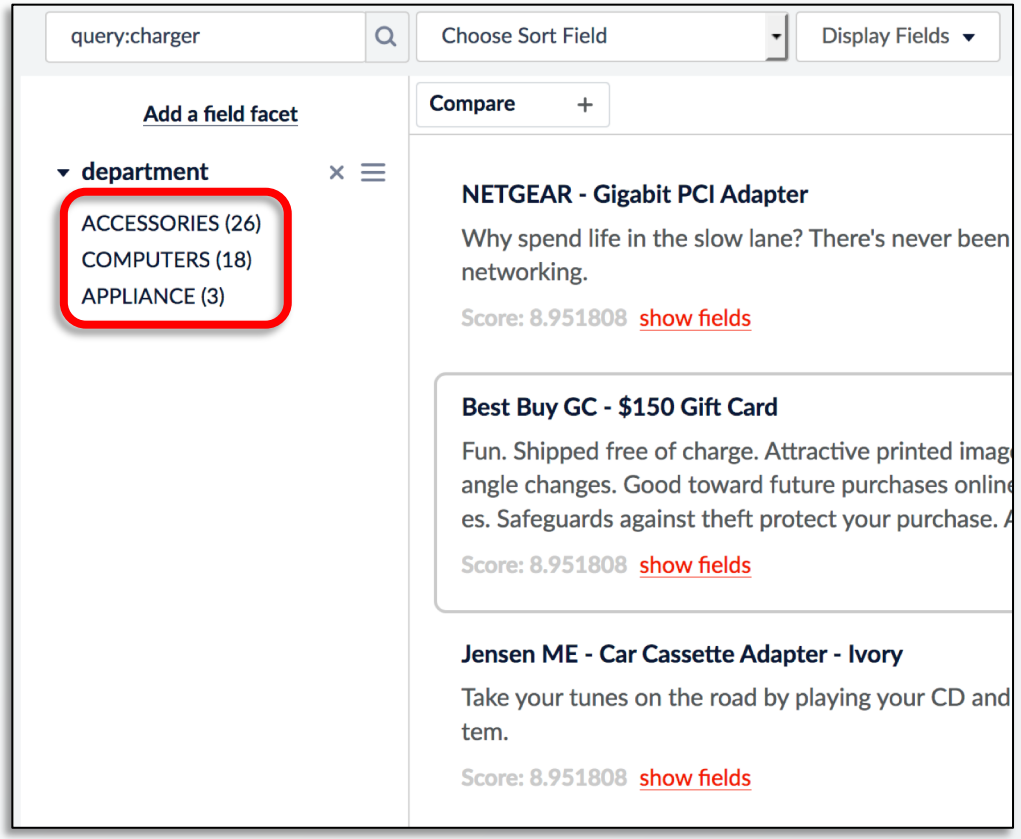
Inspecting the Signals

Lucidworks

- Execute the query **query:charger**

*These results are far more ambiguous. That said, there is still a clear trend towards **ACCESSORIES**.*

These patterns can be captured and learned by a Classifier, and used to influence relevancy at query time



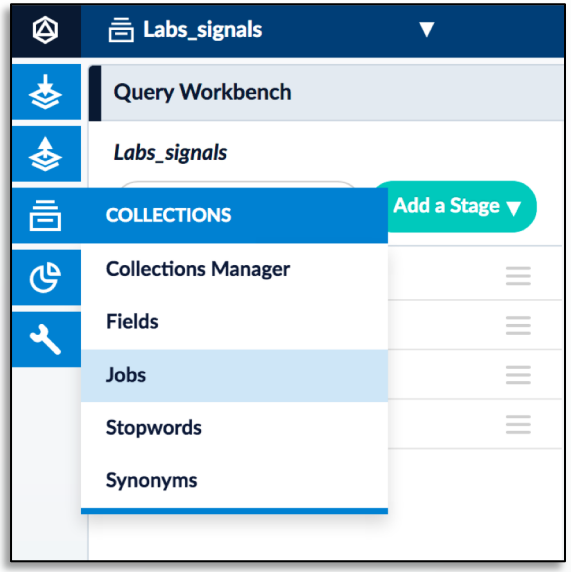
Training a Classifier Model

Creating a Classifier Job

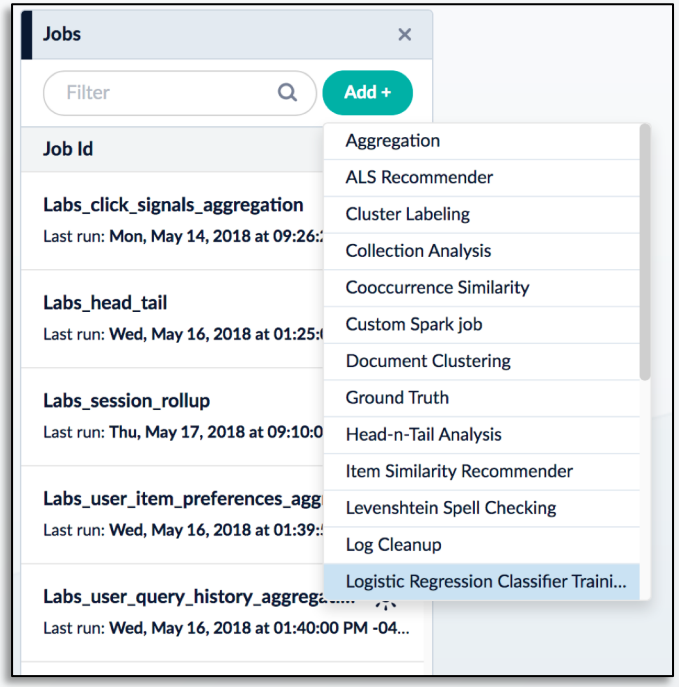
Lucidworks

Like most Machine Learning tasks, Classification models are built using Spark jobs

- In the left side menu of Fusion Admin, navigate to **COLLECTIONS > JOBS**



- In Jobs pane, click **Add**
- Select **Logistic Regression Classifier Training Job**



- Fill out the job parameters according to the following table:

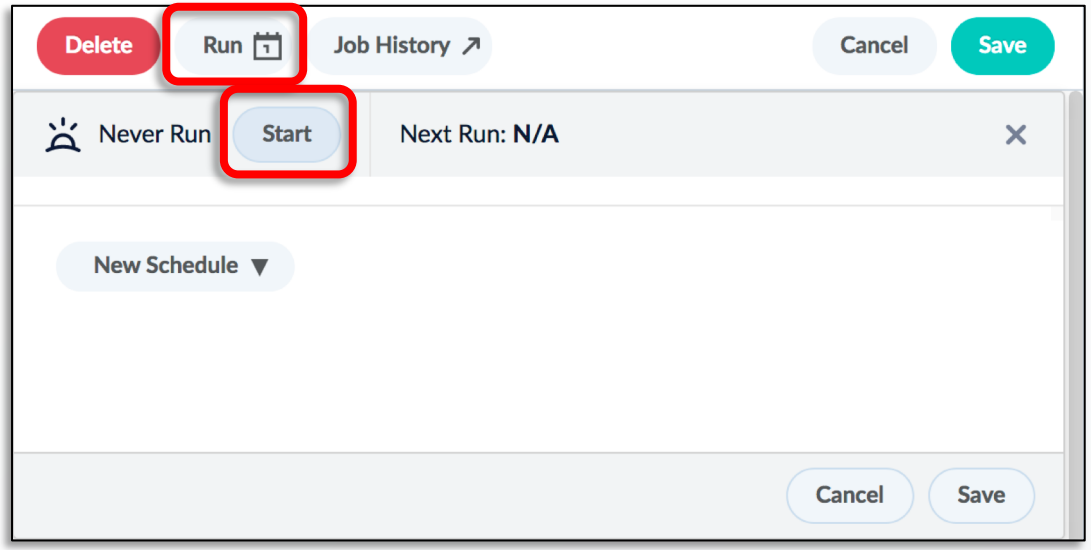
Parameter	Value	Explanation
Spark Job ID	department_qi_model	*Unique name for this job. This will also be the model name, so try to make it intuitive*
Training Collection	Labs_signals	*Collection from which to draw training data.*
Field to Vectorize	query	*The input field. The model will predict a label based on the contents of this field*
Label Field	department	*The output field. The model will write its prediction label here.*
Elastic Net Weight	0.2	*The Elastic Net (link) allows smooth interpolation of the L_1 and L_2 regularization methods. There is no single "correct" value for this, but a small number between 0 and 1 is a good base.*
Grid Search with Cross Validation	enabled	*Cross Validation is always enabled. This parameter also enables Grid Search, which will experimentally determine the "best" values for **Elastic Net Weight** and **Regularization Weight***

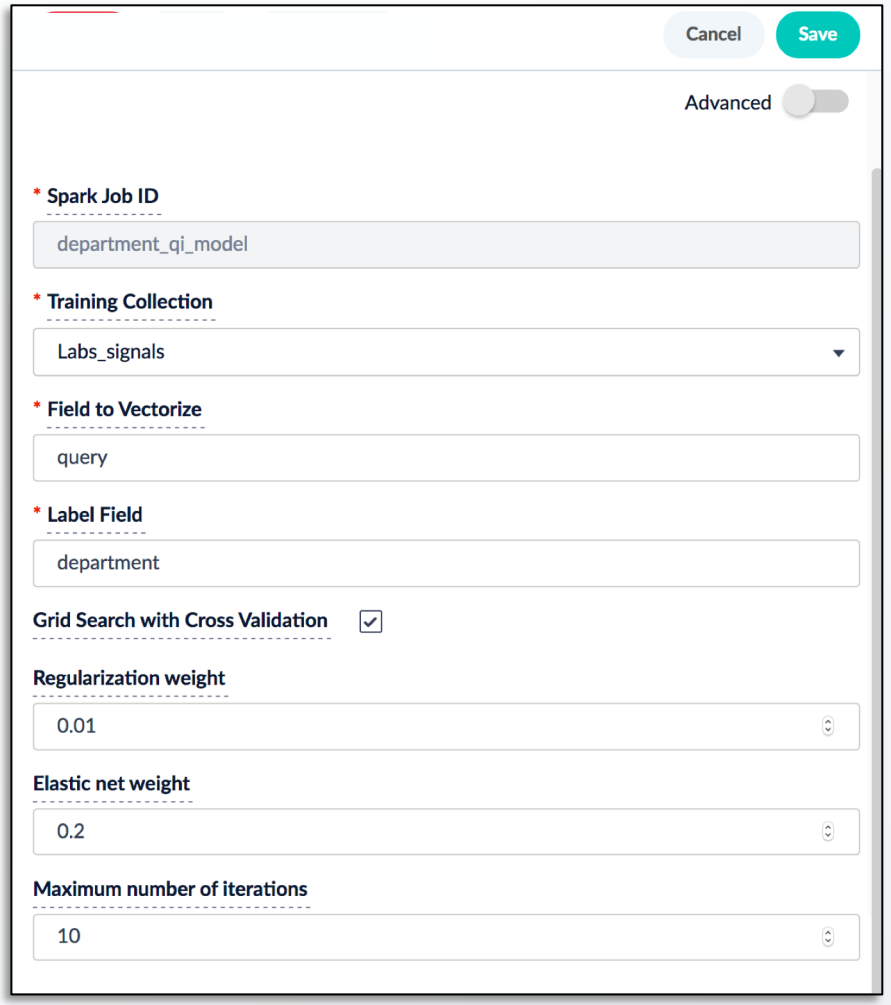
Running the Classifier Job

Lucidworks

The configured job should look like this:

- Click **Save**
- Run the job by clicking **Run > Start**

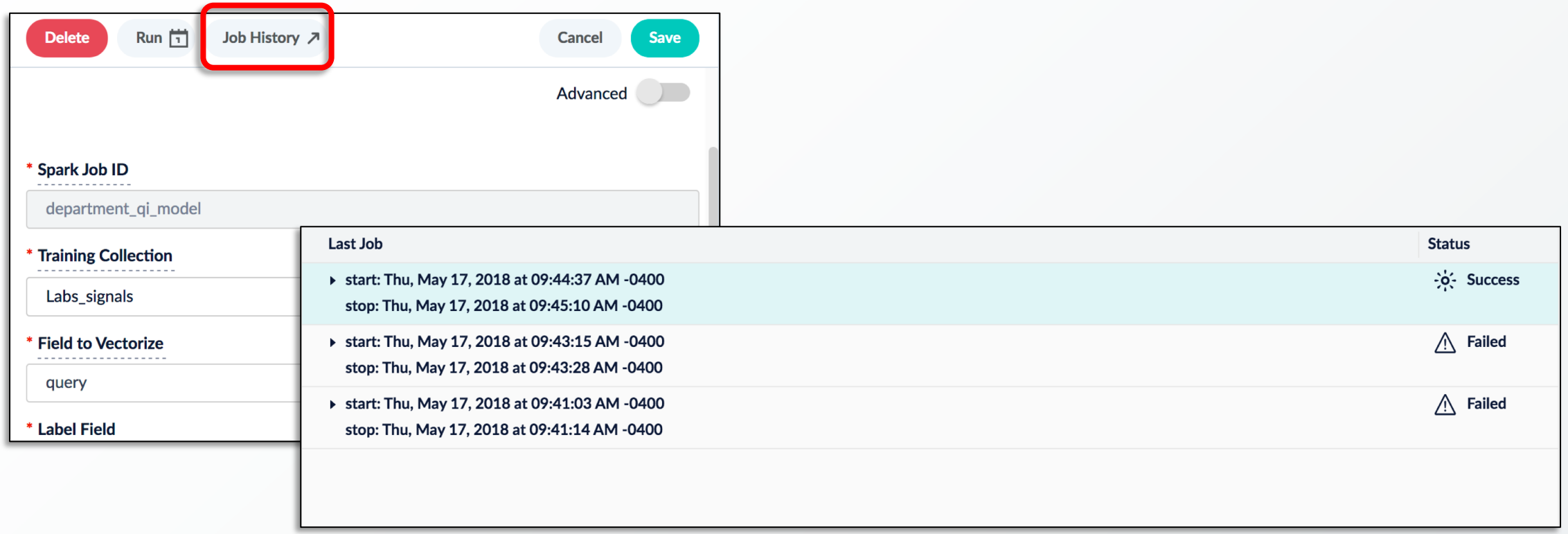




Running the Classifier Job

Lucidworks

- Once the job finishes (~2 minutes), click **Job History**
- Expand the topmost run report

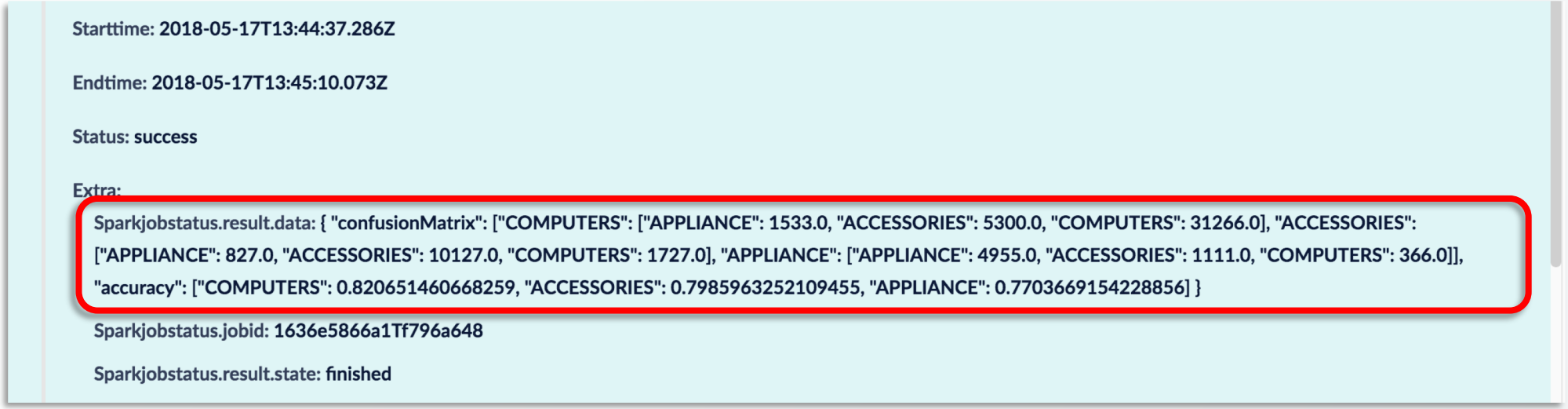


Running the Classifier Job

Lucidworks

*Observe the **confusion matrix**. This is a quick report of how accurate the model is, based on cross-validation testing. In this case, the model shows **82%** accuracy for **COMPUTERS** labels, **79%** accuracy for **ACCESSORIES**, and **77%** accuracy for **APPLIANCE**. Not bad, considering we made no effort to clean or balance the training data.*

Additionally, we can see that COMPUTERS are most often mistaken for ACESSORIES; ACCESSORIES are most often mistaken for COMPUTERS; and APPLIANCE are most often mistaken for ACCESSORIES

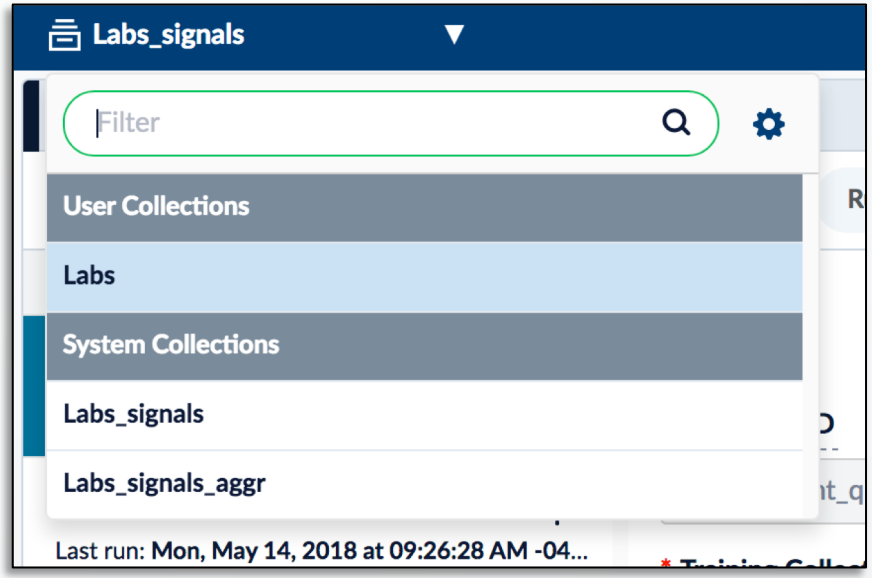


Implementing Classification at Query Time

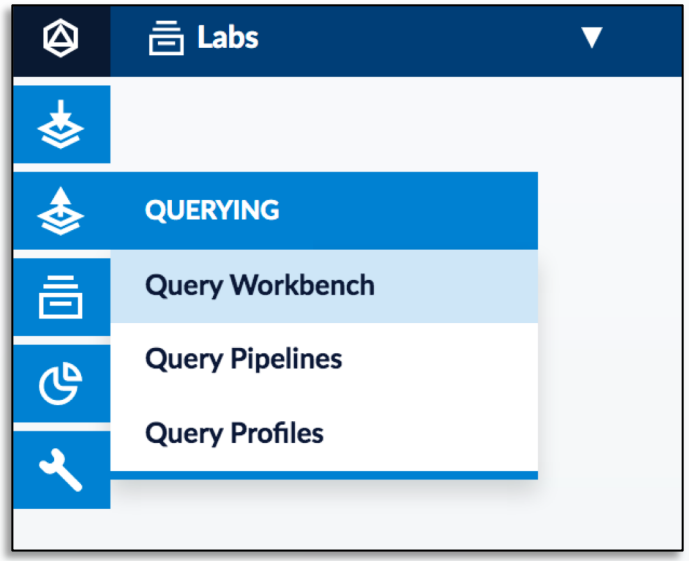
Creating a New Pipeline

Lucidworks

- In the topmost dropdown in Fusion Admin, switch to the **Labs** collection



- In the left side menu, go to **QUERYING > QUERY WORKBENCH**



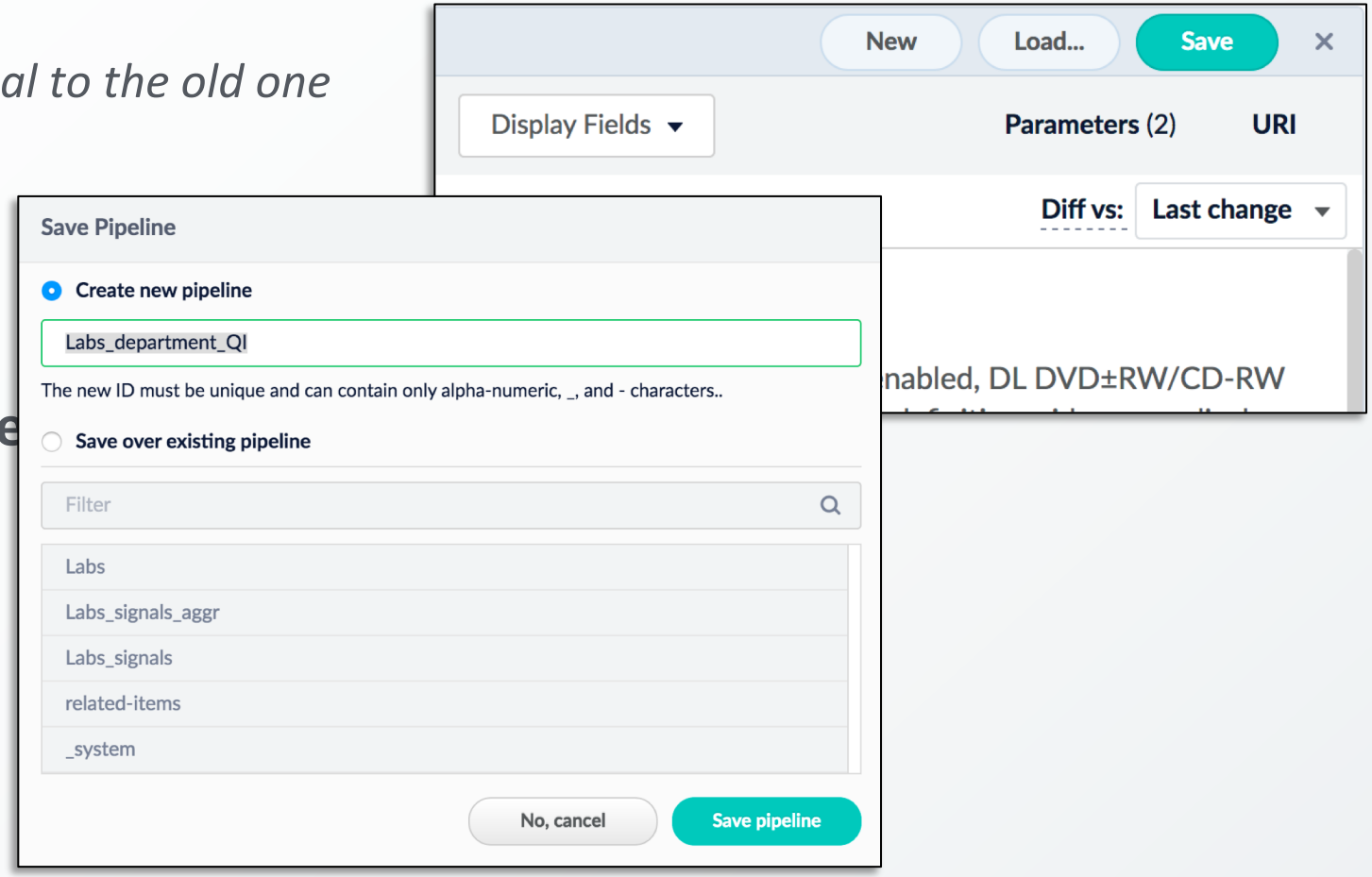
Creating a New Pipeline

Lucidworks

- Create a new query pipeline by clicking **New** in the top right corner

The initial pipeline will look identical to the old one

- Click **Save**
- Name the pipeline **Labs_de**
- Click **Save Pipeline**

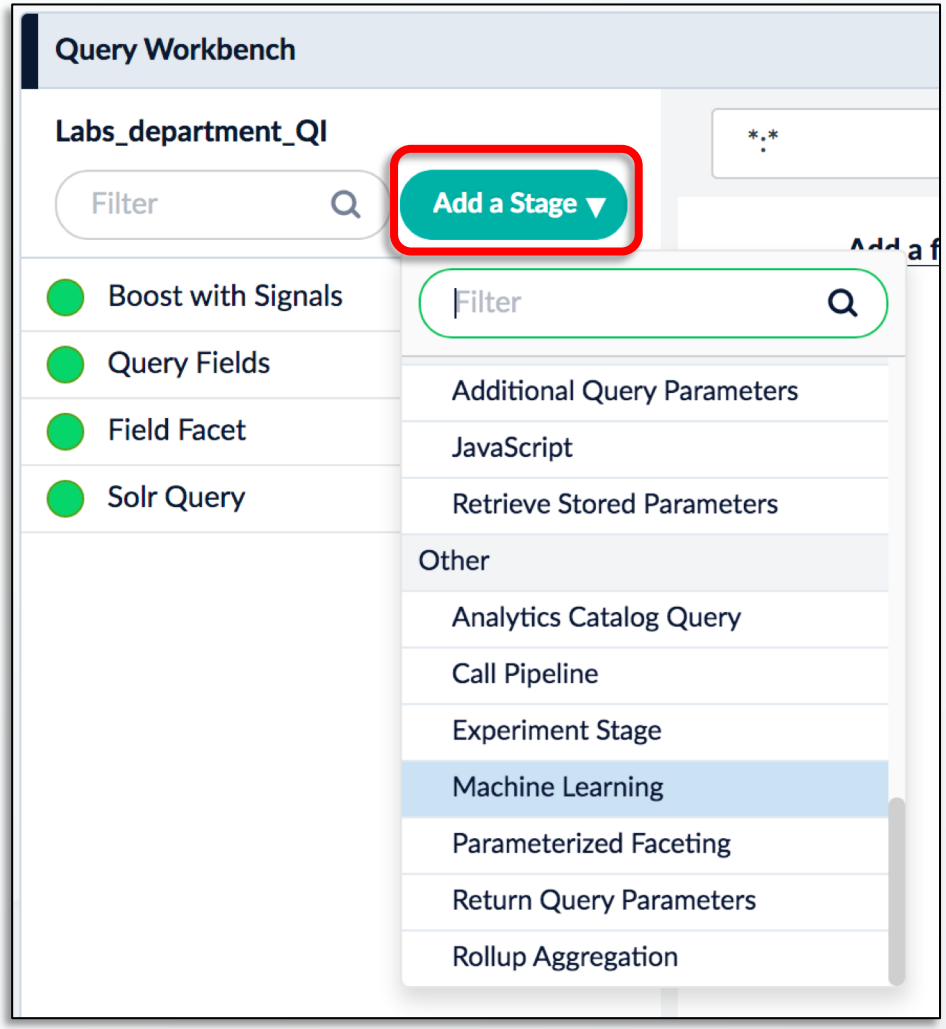


Adding a Machine Learning Stage

Lucidworks

- In the Pipeline pane, click **Add Stage**
- Select **Machine Learning**

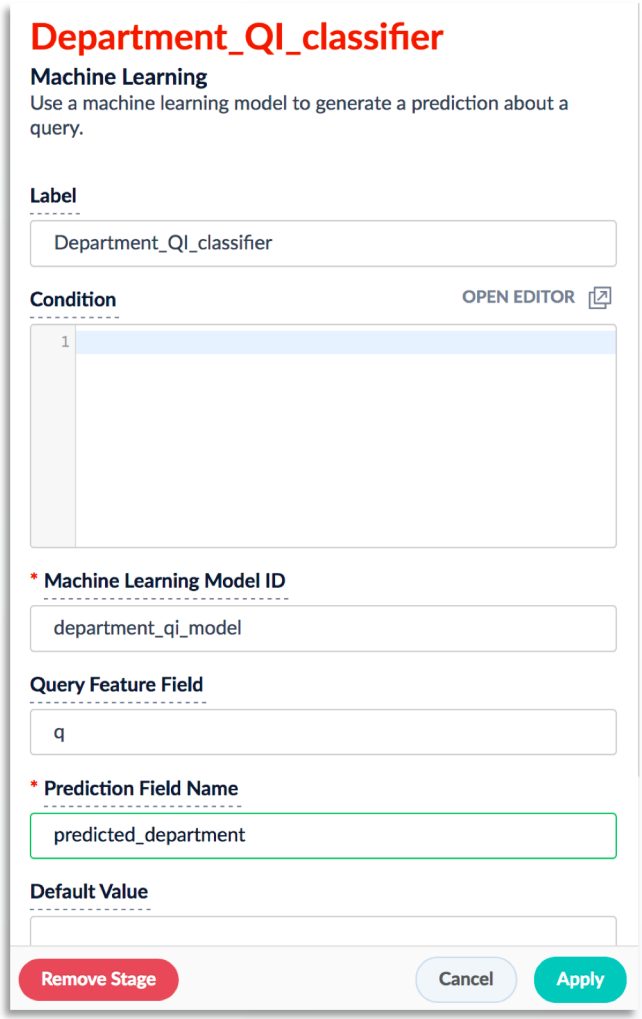
A new pipeline stage editor pane will appear



- Fill out the stage parameters according to the following table:

Parameter	Value	Explanation
Label	Department_QI_classifier	*Unique name for this pipeline stage.*
Machine Learning Model ID	department_qi_model	*Classification model to use. Recall that the model has the same name as the job used to create it. You can check the name and existence of the model by navigating to **SYSTEM > Blobs > ML Model***
Prediction Field Name	predicted_department	*Field used to store the prediction label output by the classifier. By itself, this labeling does nothing—we will implement another stage later that does something with this field*

Adding a Machine Learning Stage



- At the bottom of the stage editor, click **Apply**
- At the top right of the workbench, click **Save**

*If prompted, specify **Save over existing pipeline***

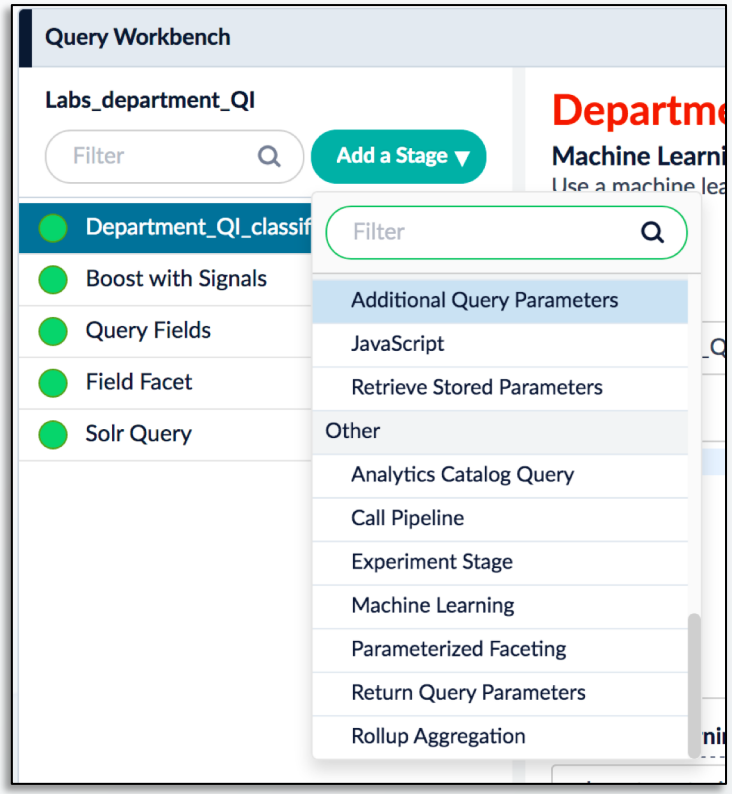
Utilizing the Classifier Labels

Lucidworks

*The classifier stage applies a label in the **predicted_department** field of each query request, but doesn't actually do anything with that label. We will create another stage that uses the label to filter or boost documents based on the label.*

- At the top right of the workbench, click **Save**
- In the Pipeline pane, click **Add Stage**
- Select **Additional Query Parameters**

A new pipeline stage editor pane will appear



Utilizing the Classifier Labels

Lucidworks

- Name the new stage **QI_department_filter**
- Click the green **+** to add a new parameter
- Fill out that parameter according to the following table:

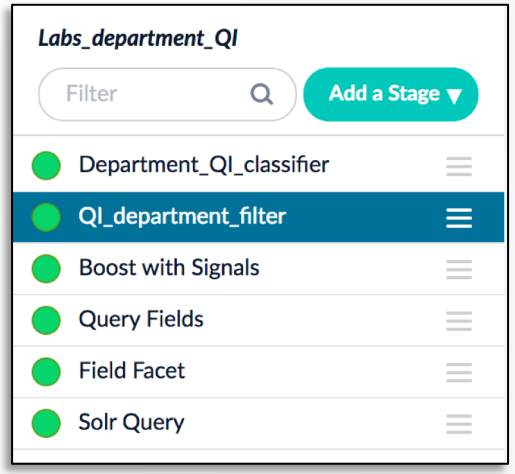
Parameter	Value	Explanation
Parameter Name	fq	*fq (Filter Query) filters search results, rather than affecting relevance*
Parameter Value	department:"<predicted_department>"	*This query matches items where the **department** field in the document matches the **predicted_department** field in the query*

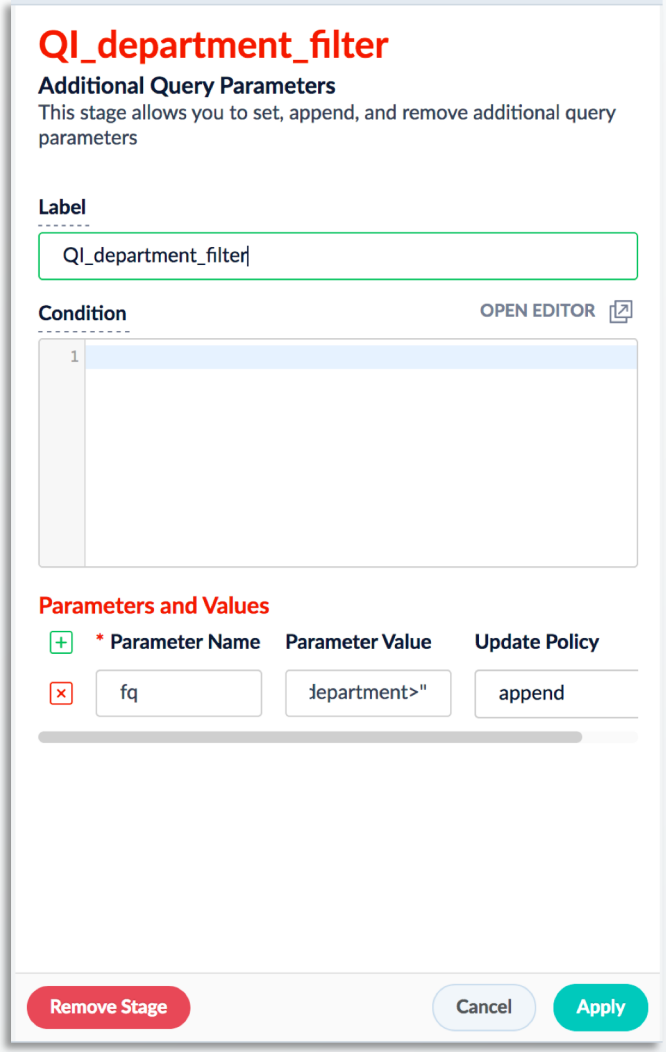
Utilizing the Classifier Labels

Lucidworks

The finished stage should look like this:

- At the bottom of the stage editor, click **Apply**
- In the pipeline pane, click and drag **QI_department_filter** so that it occurs after **Department_QI_classifier**
- At the top right of the workbench, click **Save**





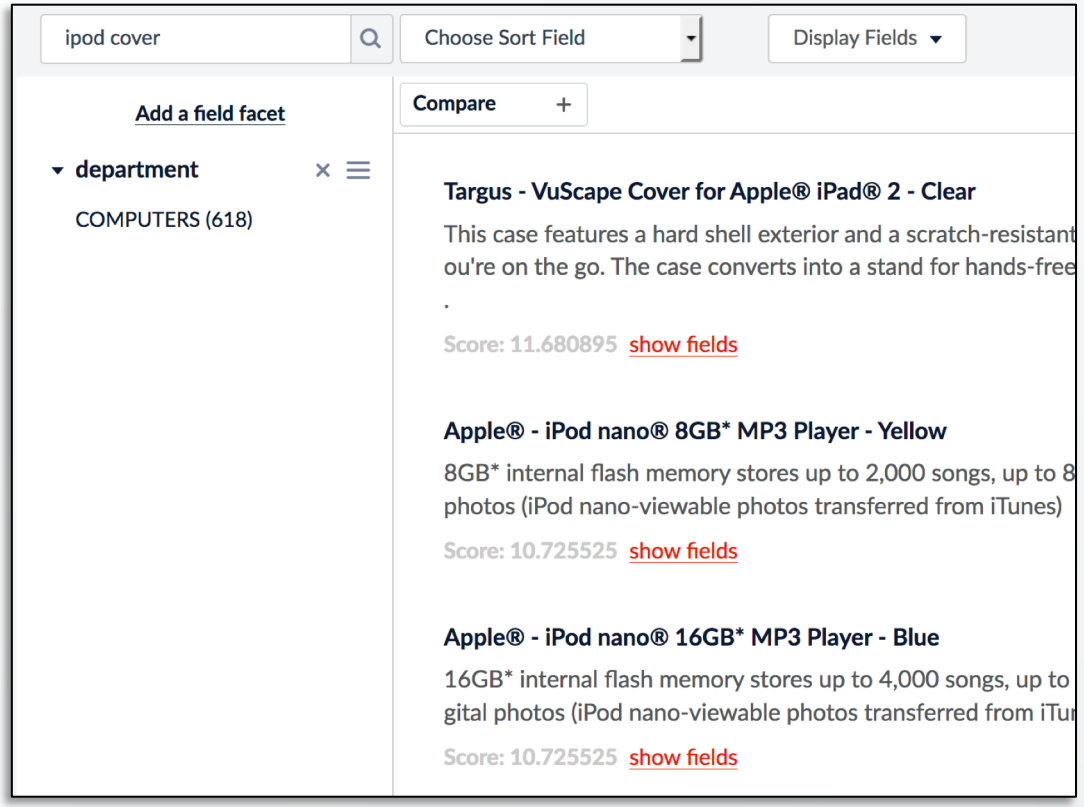
Testing the Classifier Pipeline

Testing the Classifier Pipeline

Lucidworks

- In the **Labs** Query Workbench, execute the query **ipod cover**
- **Add a facet field** for **department**

*The classifier associates the query **ipod cover** with the **COMPUTERS** department, so the department filter stage eliminates all documents from every other department*

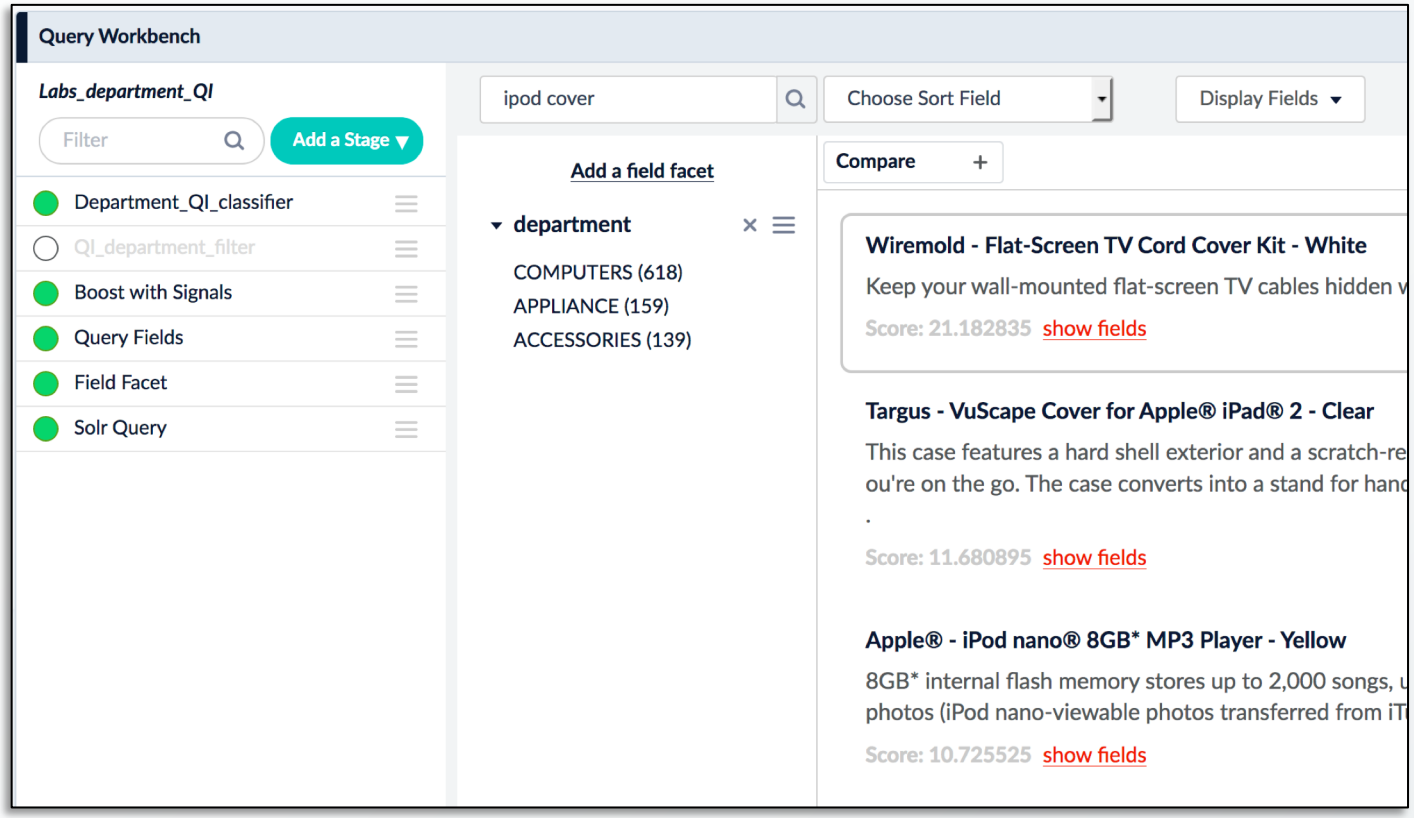


Testing the Classifier Pipeline

Lucidworks

- In the Pipeline pane, disable the classifier by clicking the green circle next to **QI_department_filter**

With the filter disabled, the query is still classified as before, but nothing is done with the label. Note that the department facet now includes documents from APPLIANCE and ACCESSORIES as well as COMPUTERS



- Re-enable the classifier by clicking the circle next to **QI_department_filter**

Filtering is not the only way we can utilize the classifier label. In fact, it is probably one of the least effective things we could do, because it means that in cases where the classifier is "wrong" the department that the user is actually interested in will not appear at all.

Let us instead use the classifier label to implement a relevance boost. In this case, the predicted department will still rise to the top of the results, but the other documents are still there to be found in case the prediction was wrong.

- Open the **QI_department_filter** editor by clicking on the stage in the Pipeline pane

Testing the Classifier Pipeline

Lucidworks

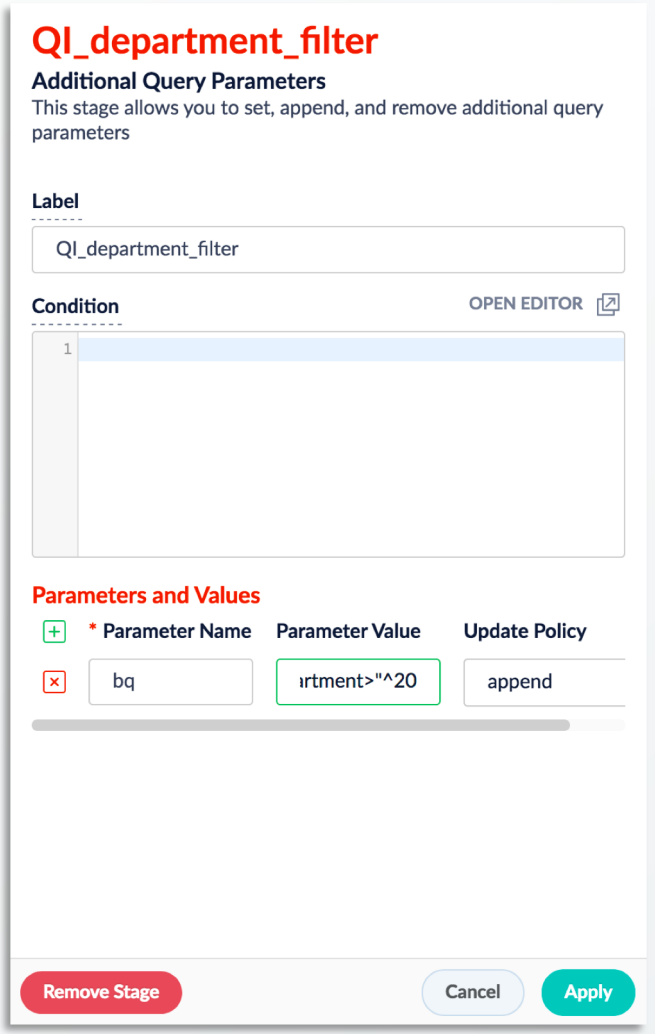
- Change the **fq** parameter to **bq**

This changes the Filter Query into a Boost query, swapping a filtering action for a relevance-boosting action.

- Change the **Parameter Value** to **department:"<predicted_department>"^20**

*The default behavior for **bq** is to add 1 to the relevance score. We'd like a larger boost, adding 20.*

- Click **Apply**



Testing the Classifier Pipeline

Lucidworks

Now we get the same COMPUTERS documents in the top results as we did with a Filter Query, but without sacrificing the other documents in case of a classifier mistake.