

LAB: Working with Signals

Prerequisites

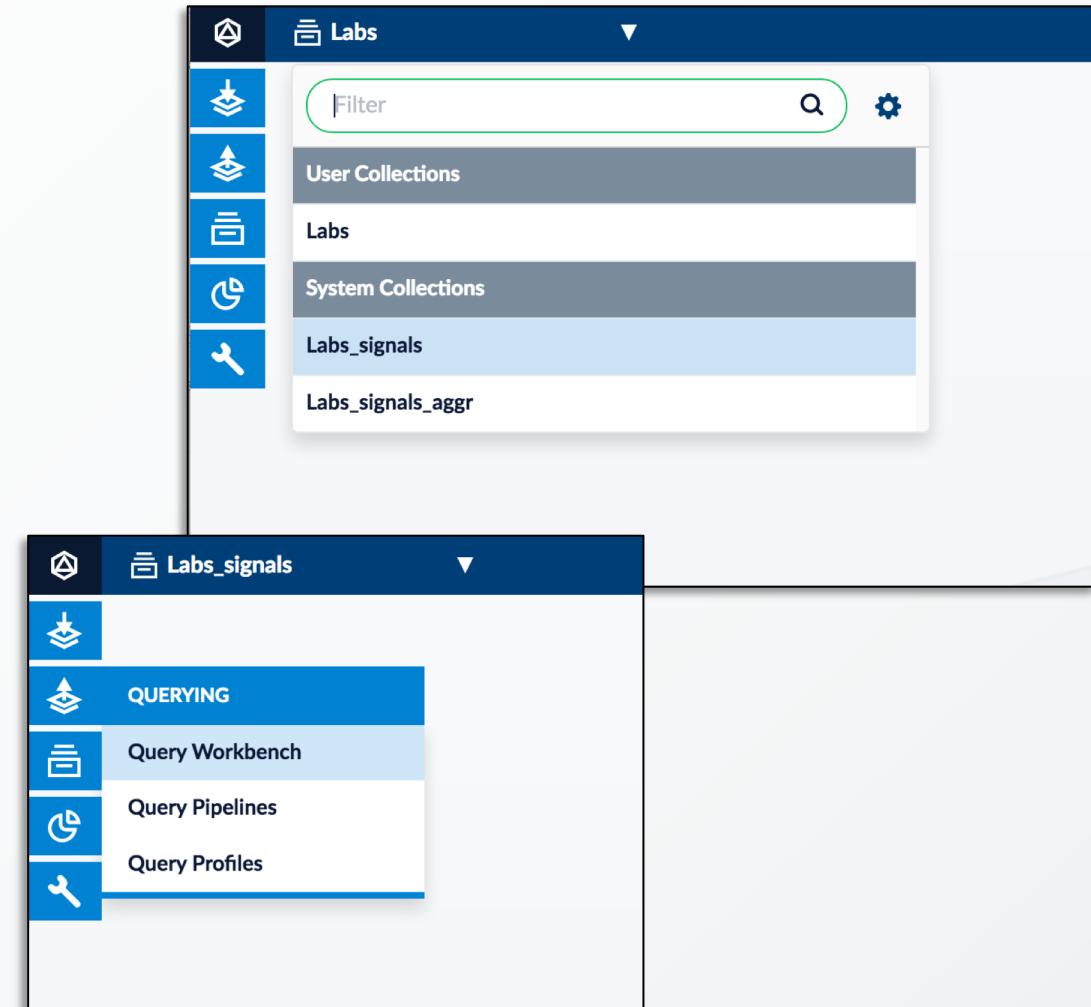
- Register for Kaggle
 - Instructions here: [Registering for Kaggle](#)
- LAB: Spark and Data Prep

Inspecting Labs_signals

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-vp-ip>:8764

- 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**



- Click Show Fields on any document

Targus - Zierra Portfolio Case for Apple® iPad® - Burgundy

This portfolio case features leather material with a scratch-resistant interior to protect your iPad 1 or 2 while you're on the go. The pockets provide ample storage space for personal accessories, such as credit cards, photos and more.

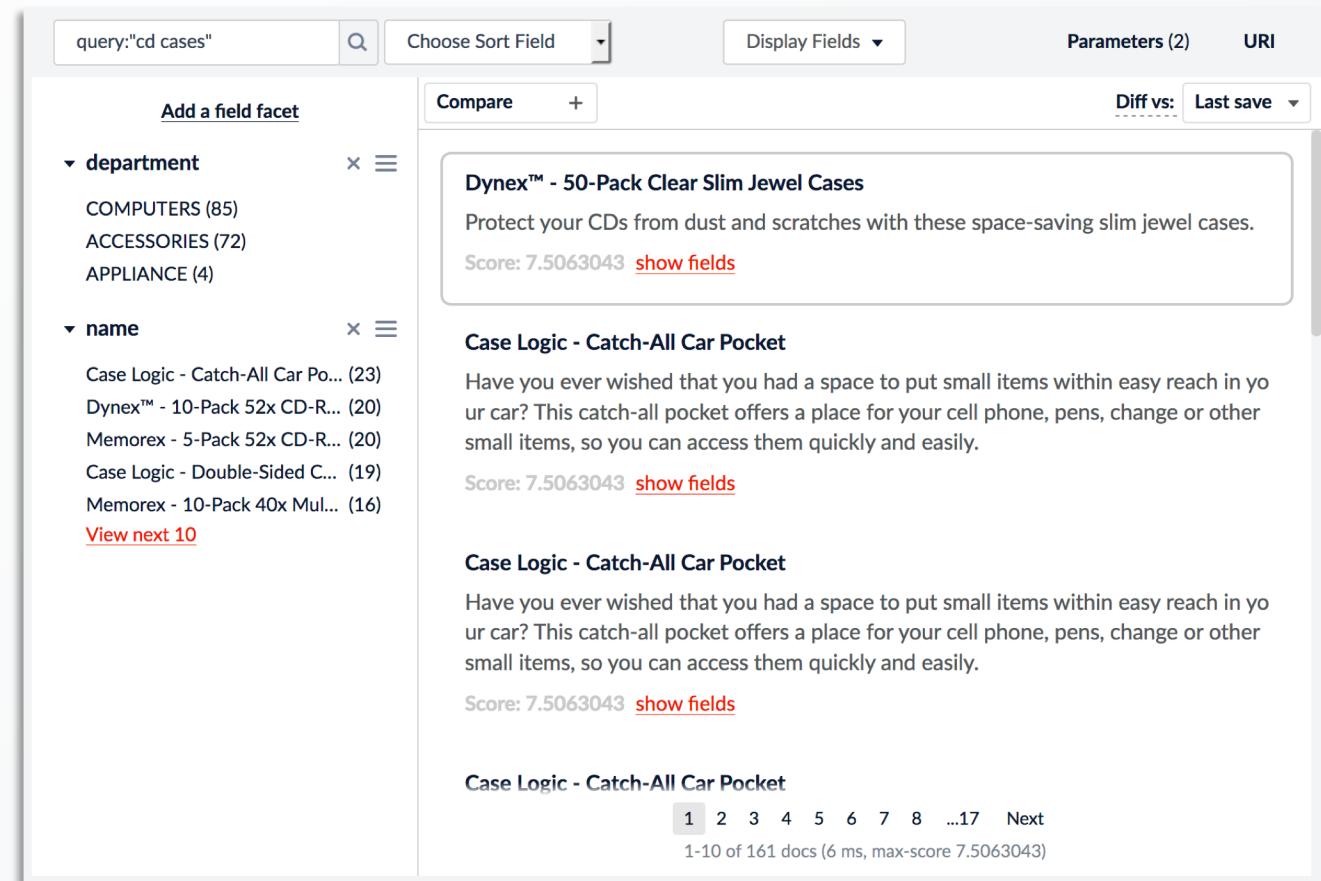
Score: 1 [hide fields](#)

department	COMPUTERS
doc_id_s	2339359
id	9232389f-3522-4a99-b7e7-d3368b02a854
longDescription	This portfolio case features leather material with a scratch-resistant interior to protect your iPad 1 or 2 while you're on the go. The pockets provide ample storage space for personal accessories, such as credit cards, photos and more.
name	Targus - Zierra Portfolio Case for Apple® iPad® - Burgundy
query	ipad 1 cases
query_orig_s	iPad 1 cases
score	1
timestamp_tdt	2018-03-08T01:19:15.720Z
type	click
user_id	816ea0a6c1e5124f318d090cc219b48646de4b90
version	1599387615630459000
orig_timestamp_tdt	2011-09-05T15:02:00.400Z

*As you can see, these signals include the baseline click data—**user_id**, **query**, **action type**, and **timestamp**—as well as extra information about the document clicked, with **name**, **longDescription**, and **department**.*

- Execute a query for **query:“cd cases”**
- Add a field facet on name

*This will show every instance where a user queried for “cd cases”. The **name** facet shows the items most commonly clicked on, and **department** shows which store department those items are from*



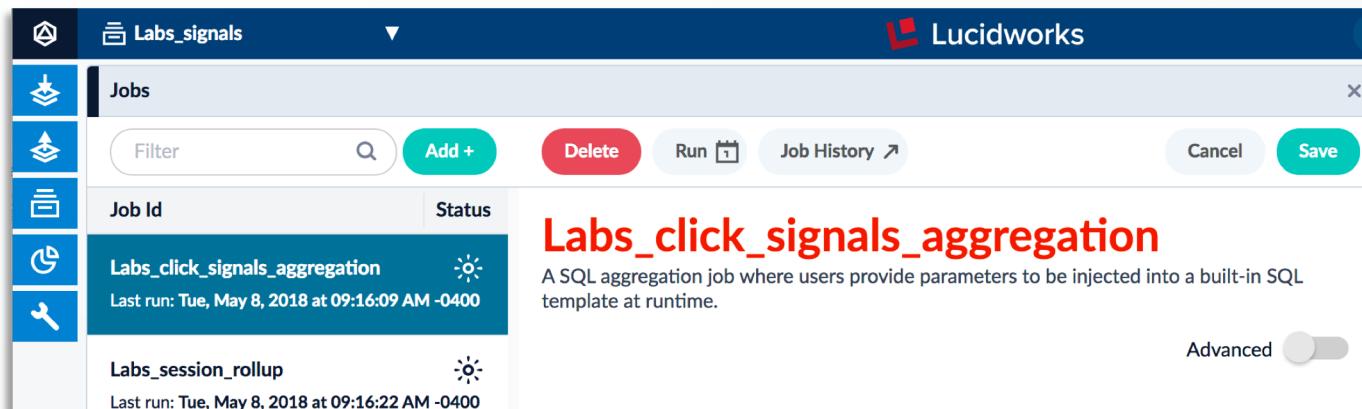
The screenshot shows a search interface with the following details:

- Query:** query:“cd cases”
- Facets:**
 - department:** COMPUTER (85), ACCESSORIES (72), APPLIANCE (4)
 - name:** Case Logic - Catch-All Car Po... (23), Dynex™ - 10-Pack 52x CD-R... (20), Memorex - 5-Pack 52x CD-R... (20), Case Logic - Double-Sided C... (19), Memorex - 10-Pack 40x Mul... (16)
- Results:** The results are grouped by item name, showing a snippet of the product description and its score.
 - Dynex™ - 50-Pack Clear Slim Jewel Cases**
Protect your CDs from dust and scratches with these space-saving slim jewel cases.
Score: 7.5063043 [show fields](#)
 - Case Logic - Catch-All Car Pocket**
Have you ever wished that you had a space to put small items within easy reach in yo ur car? This catch-all pocket offers a place for your cell phone, pens, change or other small items, so you can access them quickly and easily.
Score: 7.5063043 [show fields](#)
 - Case Logic - Catch-All Car Pocket**
Have you ever wished that you had a space to put small items within easy reach in yo ur car? This catch-all pocket offers a place for your cell phone, pens, change or other small items, so you can access them quickly and easily.
Score: 7.5063043 [show fields](#)
 - Case Logic - Catch-All Car Pocket**
Have you ever wished that you had a space to put small items within easy reach in yo ur car? This catch-all pocket offers a place for your cell phone, pens, change or other small items, so you can access them quickly and easily.
Score: 7.5063043 [show fields](#)
- Pagination:** 1 2 3 4 5 6 7 8 ...17 Next
- Page Info:** 1-10 of 161 docs (6 ms, max-score 7.5063043)

Aggregating Click Signals

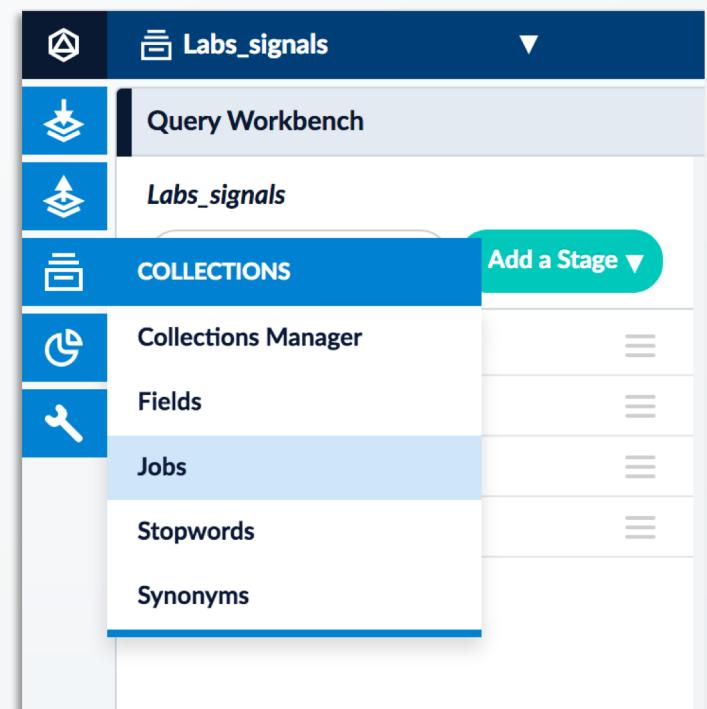
Click signals are only rarely used in their raw form. Typically, we aggregate the raw signals such that, instead of keeping 25 documents each recording 1 click each on document X, we keep a single document that records all 25 clicks for document X. This significantly streamlines query-time click usage

- In the left side menu, go to **COLLECTIONS > JOBS**
- Click the **Labs_click_signals_aggregation** job



The screenshot shows the Lucidworks Query Workbench interface. The top navigation bar has a blue header with the text "Labs_signals" and the Lucidworks logo. Below the header is a search bar and a "Jobs" button. The main content area is titled "Jobs" and contains two entries:

- Labs_click_signals_aggregation**: Status: Running (indicated by a sun icon). Last run: Tue, May 8, 2018 at 09:16:09 AM -0400. A "Save" button is visible.
- Labs_session_rollup**: Status: Pending (indicated by a sun icon). Last run: Tue, May 8, 2018 at 09:16:22 AM -0400. An "Advanced" toggle switch is visible.



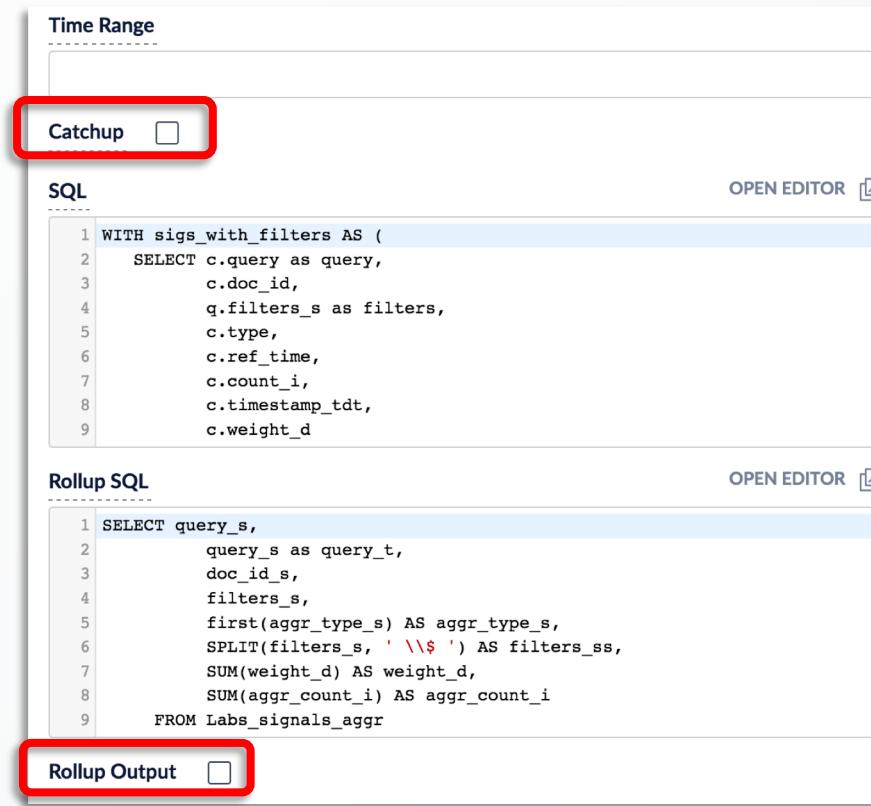
The screenshot shows the Lucidworks Query Workbench interface. The top navigation bar has a blue header with the text "Labs_signals" and the Lucidworks logo. The left sidebar is titled "COLLECTIONS" and has a "Add a Stage" button. The main content area lists several collection-related options:

- Query Workbench
- Labs_signals
- Collections** (highlighted in blue)
- Collections Manager
- Fields
- Jobs
- Stopwords
- Synonyms

- Toggle the **Advanced** switch at the top right
- Scroll down, and uncheck the **Catchup** and **Rollup Output** boxes

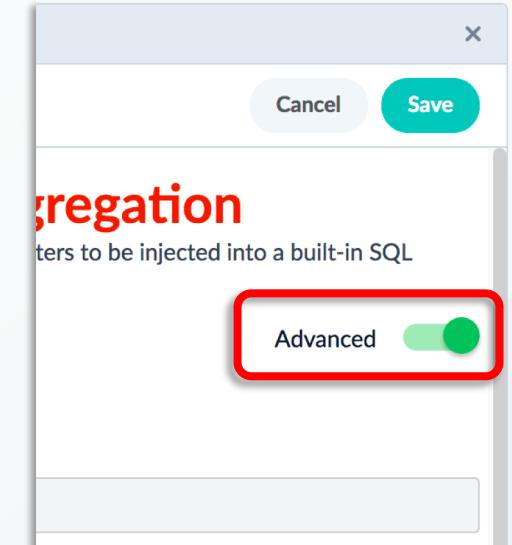
Catchup and rollup allow the aggregation job to only act upon clicks that are new since the last run. Because we have not aggregated any clicks at all yet, these parameters will interfere with expected operation.

- Click **Save**

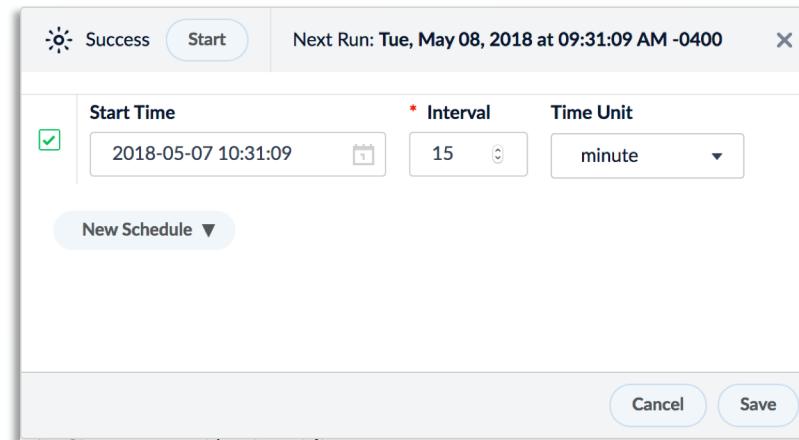


The screenshot shows the 'Time Range' section with a 'Catchup' checkbox (unchecked) highlighted by a red box. Below it is the 'SQL' section containing a complex query for signal aggregation. At the bottom is the 'Rollup SQL' section with another complex query. Both sections have 'OPEN EDITOR' buttons. At the very bottom is a 'Rollup Output' checkbox (unchecked) also highlighted by a red box.

```
WITH sigs_with_filters AS (
  SELECT c.query as query,
         c.doc_id,
         q.filters_s as filters,
         c.type,
         c.ref_time,
         c.count_i,
         c.timestamp_tdt,
         c.weight_d
)
SELECT query_s,
       query_s as query_t,
       doc_id_s,
       filters_s,
       first(aggr_type_s) AS aggr_type_s,
       SPLIT(filters_s, ' \\\$ ') AS filters_ss,
       SUM(weight_d) AS weight_d,
       SUM(aggr_count_i) AS aggr_count_i
FROM Labs_signals_aggr
```



- In the top bar, click Run



By default click signals are aggregated every fifteen minutes. Rather than wait, we will kick off the aggregation job now.

- Click Start

Performing Signal Aggregation

- In the top bar, click Job History
- Expand the topmost job

*When the job works successfully, you will see a large value for **Aggregated** items.*

- Once you have verified success, re-check the **Catchup** and **Rollup** parameters and **Save**

This prevents accidentally reduplicating the aggregated signals every time the job runs.

Last Job	Status
▼ start: Mon, May 14, 2018 at 09:26:28 AM -0400 stop: Mon, May 14, 2018 at 09:26:41 AM -0400	 Success
Resource: spark:Labs_click_signals_aggregation	
Starttime: 2018-05-14T13:26:28.299Z	
Endtime: 2018-05-14T13:26:41.465Z	
Status: success	
Extra:	
Sparkjobstatus.result.state: finished	
Sparkjobstatus.properties.collection: Labs_signals	
Sparkjobstatus.result.raw: 27	
Query: WITH sigs_with_filters AS (SELECT c.query as query, c.doc_id, q.filters_s as filters, c.type, c.ref_time, c.count_i, c.timestamp_tdt, c.weight_d FROM labs_signals c LEFT JOIN (SELECT id, filters_s FROM labs_signals WHERE type='response') q ON q.id = c.fusion_query_id WHERE c.type IN ('click','cart','purchase')), signal_type_groups AS (SELECT SUM(count_i) AS typed_aggr_count_i, query, doc_id, type, filters, time_decay(count_i, timestamp_tdt) AS typed_weight_d FROM sigs_with_filters GROUP BY doc_id, query, filters, type) SELECT SUM(typed_aggr_count_i) AS aggr_count_i, query AS query_s, query AS query_t, doc_id AS doc_id_s, filters AS filters_s, SPLIT(filters, '\\$') AS filters_ss, weighted_sum(typed_weight_d, type, 'click:1.0,cart:10.0,purchase:25.0') AS weight_d FROM signal_type_groups GROUP BY query, doc_id, filters	
Sparkjobstatus.hostname: 172.31.48.117	
Raw: 27	
Sparkjobstatus.properties.aggregation: 2018-05-14T13:26:28.28Z	
Aggregated: 119953	
Sparkjobstatus.result.aggr: click@doc_id,filters,query	
Sparkjobstatus.properties.now: 1526304388288	

- In the top left dropdown, change to the **Labs_signals_aggr** collection
- In the left side menu, navigate to **QUERYING > Query Workbench**
- Click **Show Fields** on any document

b0421585-84f4-4e29-881f-8b5f8fd2196e	↑ Boost	Block
9735301		
Score: 1	hide fields	
aggr_count_i	84	
aggr_id_s	Labs_click_signals_aggregation	
aggr_job_id_s	1635ed4b4c0Tca1b2cdc	
aggr_type_s	click@doc_id,filters,query	
doc_id_s	9735301	
flag_s	aggr	
id	b0421585-84f4-4e29-881f-8b5f8fd2196e	
query_s	dvd rack	
query_t	["dvd rack"]	
score	1	
timestamp_tdt	2018-05-14T13:26:28.280Z	
weight_d	5.032842296554202	
version	1600446158935687200	

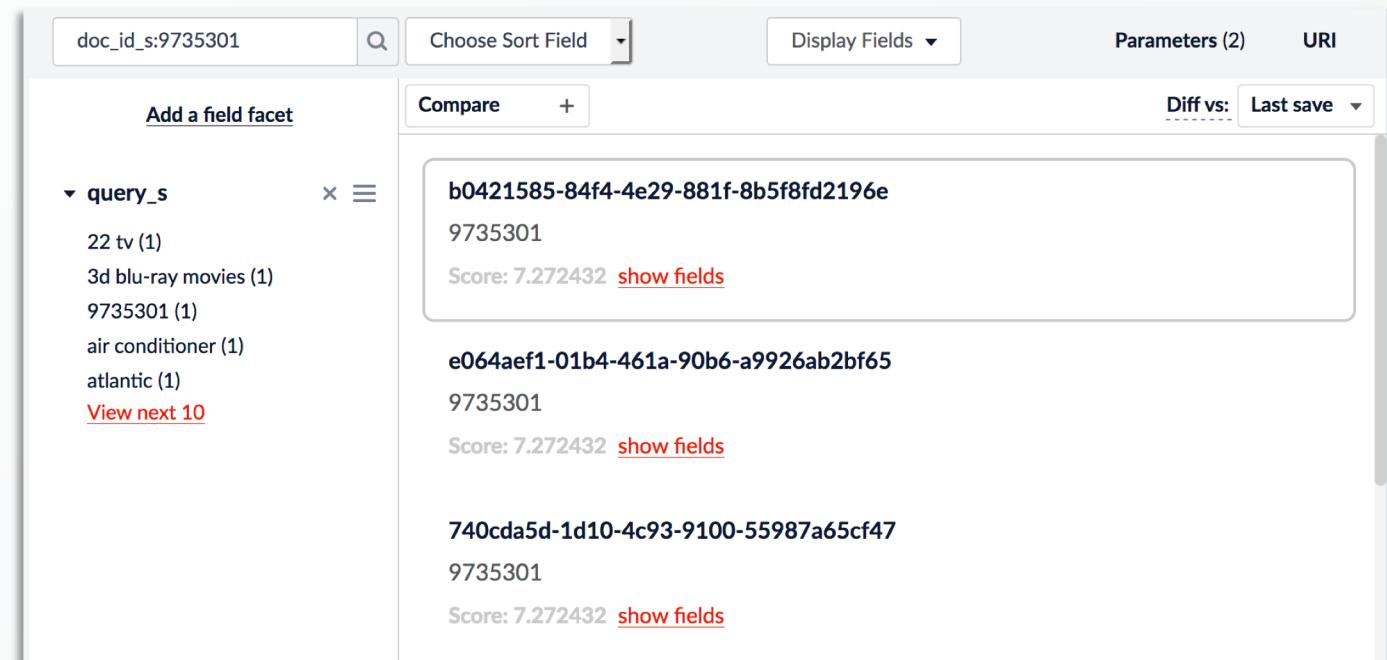
The fields **doc_id_s** and **query_s** record the original query entered by the user(s), along with which document they ultimately clicked.

aggr_count_i records how many times this specific query ("dvd rack") led to a click on this document (9735301)

weight_d indicates how much to promote this document in response to this query

- Execute a query for **doc_id_s:9735301**
- Add a field facet on **query_s**

This query and facet show all of the user queries that led to a click on this document. We would need to examine individual documents to see the click-count and weight for each case



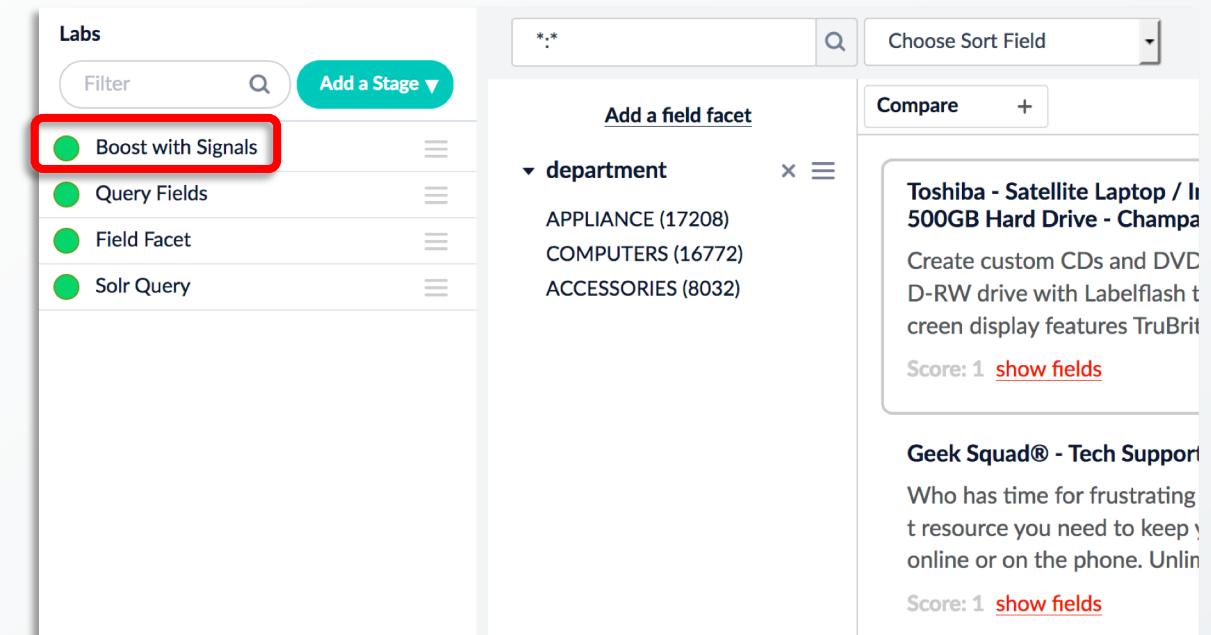
The screenshot shows the Lucidworks interface with the following details:

- Search Bar:** doc_id_s:9735301
- Facet Panel:** Add a field facet (selected), query_s:
 - 22 tv (1)
 - 3d blu-ray movies (1)
 - 9735301 (1)
 - air conditioner (1)
 - atlantic (1)
 - [View next 10](#)
- Result List:**
 - b0421585-84f4-4e29-881f-8b5f8fd2196e
9735301
Score: 7.272432 [show fields](#)
 - e064aef1-01b4-461a-90b6-a9926ab2bf65
9735301
Score: 7.272432 [show fields](#)
 - 740cda5d-1d10-4c93-9100-55987a65cf47
9735301
Score: 7.272432 [show fields](#)
- Header Buttons:** Choose Sort Field, Display Fields, Parameters (2), URI, Diff vs: Last save

Applying Signal Boosting

- In the top left dropdown, change to the **Labs** collection
- In the left side menu, navigate to **QUERYING > Query Workbench**

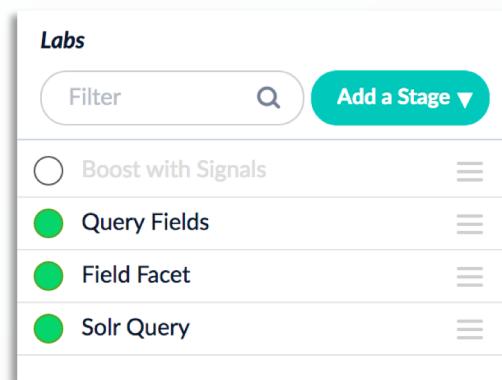
*As you can see in the pipeline stages,
signal boosting is enabled by default*



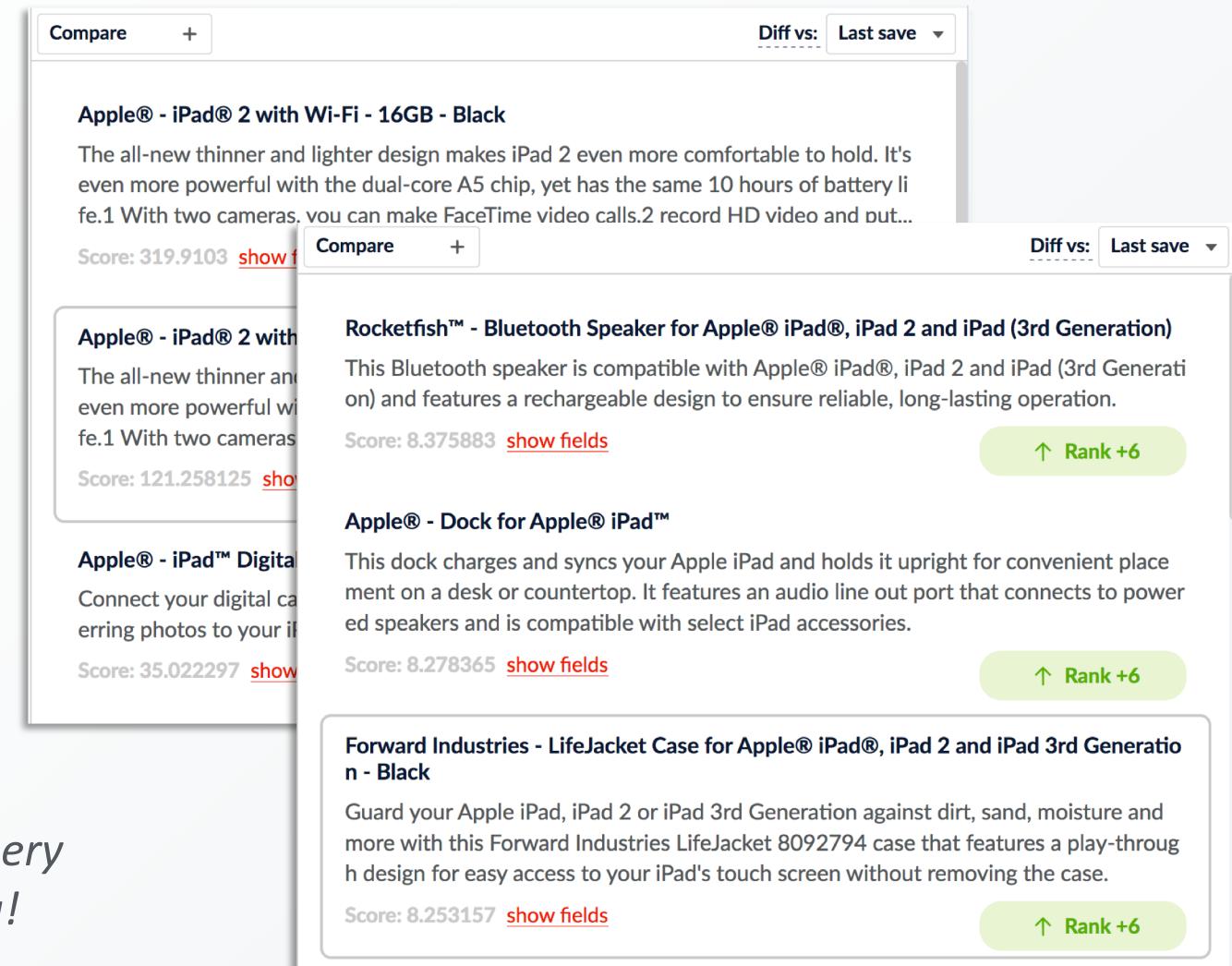
The screenshot shows the Lucidworks Query Workbench interface. On the left, there's a sidebar titled "Labs" with a "Boost with Signals" stage highlighted by a red box. The main area displays search results for "department". A facet for "department" is shown with categories: APPLIANCE (17208), COMPUTERS (16772), and ACCESSORIES (8032). Below the facets, search results are listed, starting with a result for "Toshiba - Satellite Laptop / I 500GB Hard Drive - Champa". Another result for "Geek Squad® - Tech Support" is partially visible at the bottom.

Evaluating Signal Boosting

- Execute a query for **ipad**
- In the query pipeline, disable the **Boost with Signals** stage by clicking the green circle next to it



Observe how much less intuitive the query results become without signal boosting!



The screenshot shows a search results page with the following items:

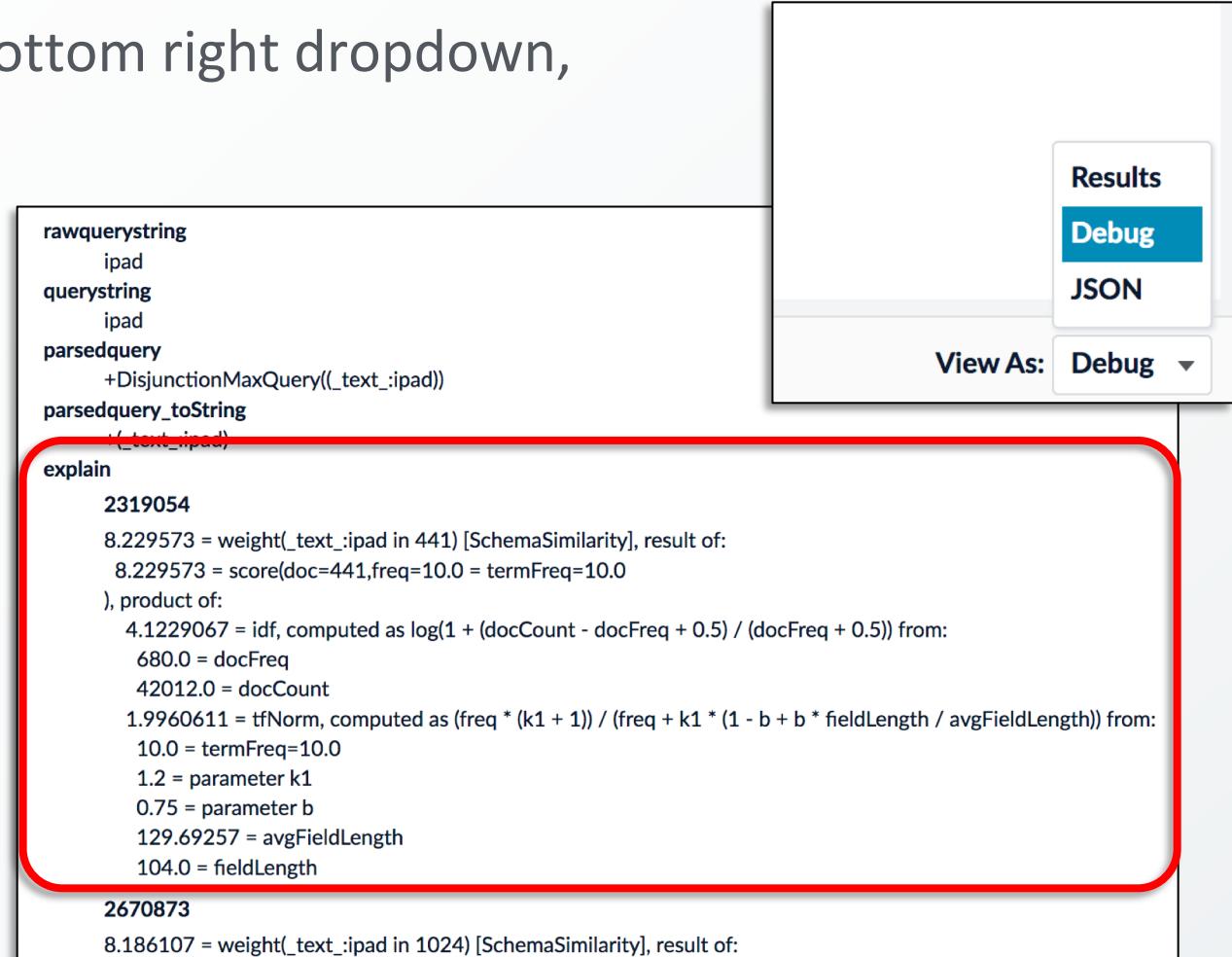
- Apple® - iPad® 2 with Wi-Fi - 16GB - Black**
The all-new thinner and lighter design makes iPad 2 even more comfortable to hold. It's even more powerful with the dual-core A5 chip, yet has the same 10 hours of battery life.1 With two cameras, you can make FaceTime video calls.2 record HD video and put...
Score: 319.9103 [show fields](#)
- Apple® - iPad® 2 with**
The all-new thinner and lighter design makes iPad 2 even more comfortable to hold. It's even more powerful with the dual-core A5 chip, yet has the same 10 hours of battery life.1 With two cameras, you can make FaceTime video calls.2 record HD video and put...
Score: 121.258125 [show fields](#)
- Rocketfish™ - Bluetooth Speaker for Apple® iPad®, iPad 2 and iPad (3rd Generation)**
This Bluetooth speaker is compatible with Apple® iPad®, iPad 2 and iPad (3rd Generation) and features a rechargeable design to ensure reliable, long-lasting operation.
Score: 8.375883 [show fields](#) ↑ Rank +6
- Apple® - Dock for Apple® iPad™**
This dock charges and syncs your Apple iPad and holds it upright for convenient placement on a desk or countertop. It features an audio line out port that connects to powered speakers and is compatible with select iPad accessories.
Score: 8.278365 [show fields](#) ↑ Rank +6
- Forward Industries - LifeJacket Case for Apple® iPad®, iPad 2 and iPad 3rd Generation - Black**
Guard your Apple iPad, iPad 2 or iPad 3rd Generation against dirt, sand, moisture and more with this Forward Industries LifeJacket 8092794 case that features a play-through design for easy access to your iPad's touch screen without removing the case.
Score: 8.253157 [show fields](#) ↑ Rank +6

Optional:

Analyzing Signal Boosting

- In the Query Workbench, in the bottom right dropdown, change to **Debug** output
- Observe the **explain** section

*This section provides a full description of how the relevancy score for each results document was calculated. At present, with signal boosting disabled, this is based solely on TF*IDF*



```
rawquerystring
  ipad
queryString
  ipad
parsedquery
  +DisjunctionMaxQuery(_text_ipad)
parsedquery_toString
  '_text_ipad'
explain
  2319054
    8.229573 = weight(_text_ipad in 441) [SchemaSimilarity], result of:
      8.229573 = score(doc=441,freq=10.0 = termFreq=10.0
    ), product of:
      4.1229067 = idf, computed as log(1 + (docCount - docFreq + 0.5) / (docFreq + 0.5)) from:
        680.0 = docFreq
        42012.0 = docCount
      1.9960611 = tfNorm, computed as (freq * (k1 + 1)) / (freq + k1 * (1 - b + b * fieldLength / avgFieldLength)) from:
        10.0 = termFreq=10.0
        1.2 = parameter k1
        0.75 = parameter b
        129.69257 = avgFieldLength
        104.0 = fieldLength
  2670873
    8.186107 = weight(_text_ipad in 1024) [SchemaSimilarity], result of:
```

- Re-enable the **Boost with Signals** stage, and observe the debug output

*First, you notice that the **parsedquery** became MUCH more complex, due to the added boosting.*

*The doc ids you see in the boosted query were generated by first executing the same user query (“ipad”) on the **Labs_signals_aggr** collection, and collecting the top-weighted results*

```
rawquerystring
  ipad
querystring
  ipad
parsedquery
  BoostedQuery(boost(+_text_ipad),product(map(query(id:1945531,def=0.0),0.0,0.0,0,const(1),const(47.2156)),map(query(id:2339322,def=0.0),0.0,0.0,0,const(1),const(17.8965)),map(query(id:2842056,def=0.0),0.0,0.0,0,const(1),const(7.8027)),map(query(id:2475916,def=0.0),0.0,0.0,0,const(1),const(7.5812)),map(query(id:9924603,def=0.0),0.0,0.0,0,const(1),const(4.901)),map(query(id:1918265,def=0.0),0.0,0.0,0,const(1),const(3.5606)),map(query(id:9947181,def=0.0),0.0,0.0,0,const(1),const(3.2826)),map(query(id:1918159,def=0.0),0.0,0.0,0,const(1),const(3.0542)),map(query(id:2343139,def=0.0),0.0,0.0,0,const(1),const(2.6459)),map(query(id:9635348,def=0.0),0.0,0.0,0,const(1),const(2.5955)))))

parsedquery_toString
  boost(+_text_ipad),product(map(query(id:1945531,def=0.0),0.0,0.0,0,const(1),const(47.2156)),map(query(id:2339322,def=0.0),0.0,0.0,0,const(1),const(17.8965)),map(query(id:2842056,def=0.0),0.0,0.0,0,const(1),const(7.8027)),map(query(id:2475916,def=0.0),0.0,0.0,0,const(1),const(7.5812)),map(query(id:9924603,def=0.0),0.0,0.0,0,const(1),const(4.901)),map(query(id:1918265,def=0.0),0.0,0.0,0,const(1),const(3.5606)),map(query(id:9947181,def=0.0),0.0,0.0,0,const(1),const(3.2826)),map(query(id:1918159,def=0.0),0.0,0.0,0,const(1),const(3.0542)),map(query(id:2343139,def=0.0),0.0,0.0,0,const(1),const(2.6459)),map(query(id:9635348,def=0.0),0.0,0.0,0,const(1),const(2.5955))))
```

Second, notice that the relevance **explain** became more complex as well. Now there is a Signal Boosting factor in addition to the TF*IDF factors.

In this case, the final score for this document (20.69) was produced by similarly-scaled factors for TF*IDF (4.12, 1.64) and Signals (3.05). One did not overpower the other—a very good thing!

```
explain
1918159
20.6938 = boost(_text_ipad,product(map(query(id:1945531,def=0.0),0.0,0.0,0.0,const(1),const(47.2156)),map(query(id:2339322,def=0.0),0.0,0.0,0.0,const(1),const(17.8965)),map(query(id:2842056,def=0.0),0.0,0.0,0.0,const(1),const(7.8027)),map(query(id:2475916,def=0.0),0.0,0.0,0.0,const(1),const(7.5812)),map(query(id:9924603,def=0.0),0.0,0.0,0.0,const(1),const(4.901)),map(query(id:1918265,def=0.0),0.0,0.0,0.0,const(1),const(3.5606)),map(query(id:9947181,def=0.0),0.0,0.0,0.0,const(1),const(3.2826)),map(query(id:1918159,def=0.0),0.0,0.0,0.0,const(1),const(3.0542)),map(query(id:2343139,def=0.0),0.0,0.0,0.0,const(1),const(2.6459)),map(query(id:9635348,def=0.0),0.0,0.0,0.0,const(1),const(2.5955))), product of:
6.775522 = weight(_text_ipad in 21135) [SchemaSimilarity], result of:
6.775522 = score(doc=21135,freq=4.0 = termFreq=4.0
), product of:
4.1229067 = idf, computed as log(1 + (docCount - docFreq + 0.5) / (docFreq + 0.5)) from:
680.0 = docFreq
42012.0 = docCount
1.6433848 = tfNorm, computed as (freq * (k1 + 1)) / (freq + k1 * (1 - b + b * fieldLength / avgFieldLength)) from:
4.0 = termFreq=4.0
1.2 = parameter k1
0.75 = parameter b
129.69257 = avgFieldLength
152.0 = fieldLength
3.0542 = product(map(query(id:1945531,def=0.0)=0.0,min=0.0,max=0.0,target=const(1),defaultVal=const(47.2156)),map(query(id:2339322,def=0.0)=0.0,min=0.0,max=0.0,target=const(1),defaultVal=const(17.8965)),map(query(id:2842056,def=0.0)=0.0,min=0.0,max=0.0,target=const(1),defaultVal=const(7.8027)),map(query(id:2475916,def=0.0)=0.0,min=0.0,max=0.0,target=const(1),defaultVal=const(7.5812)),map(query(id:9924603,def=0.0)=0.0,min=0.0,max=0.0,target=const(1),defaultVal=const(4.901)),map(query(id:1918265,def=0.0)=0.0,min=0.0,max=0.0,target=const(1),defaultVal=const(3.5606)),map(query(id:9947181,def=0.0)=0.0,min=0.0,max=0.0,target=const(1),defaultVal=const(3.2826)),map(query(id:1918159,def=0.0)=10.24027,min=0.0,max=0.0,target=const(1),defaultVal=const(3.0542)),map(query(id:2343139,def=0.0)=0.0,min=0.0,max=0.0,target=const(1),defaultVal=const(2.6459)),map(query(id:9635348,def=0.0)=0.0,min=0.0,max=0.0,target=const(1),defaultVal=const(2.5955)))
```

TF*IDF

Signal Boosting

- Click to open the **Boost with Signals** pipeline stage, and scroll down

*These parameters govern how Fusion should locate and prioritize the signals related to a given query, as well as how the Signals-score should be combined with the standard TF*IDF score.*

When implementing Signal Boosting in a Fusion App, all of these parameters can and should be tuned to suit your specific use cases.

* Boost Param

Scale Boosts

Scale the boost values to a [min,max] range

include

Solr Query parameters

Parameters for querying Signal aggregation collection

* Parameter Name	Parameter Value
qf	query_t
pf	query_t^10
pf	query_t~2^7
mm	50%
defType	edismax
sort	score desc, weight_d de

Understanding the Boost with Signals Stage, cont.

- **Boost Param** has two options: **boost**, for a multiplicative combination of TF*IDF and signal boosting; and **bq** (Boost Query), for an additive combination
- The **qf** (query fields) parameter specifies that the user query must appear in the **query_t** field in the **signals_aggr** documents
- The **pf** (Phrase Field) parameters specify two ways to rank the signals aggregation documents. The first one says that an **exact match** should be rewarded with a **10x** multiplier. The second says that **any match within a slop factor of 2** should get a **7x** multiplier

* **Boost Param**

Scale Boosts
Scale the boost values to a [min,max] range

include

Solr Query parameters
Parameters for querying Signal aggregation collection

* Parameter Name	Parameter Value
qf	query_t
pf	query_t^10
pf	query_t~2^7
mm	50%
defType	edismax
sort	score desc, weight_d de

Understanding the Boost with Signals Stage, cont.

- The **mm** (Minimum Match) parameter specifies that at least **50%** of the user query terms must appear in signal document. Any fewer than that will not be considered a match.
- **defType** specifies which query parser to use. **edismax** is the standard for Fusion.
- The **sort** parameter specifies that the signals docs should ranked first by relevance **score** (i.e., how well they match the current user query). If two signals have the same score (e.g., two identical queries that led to clicks on different documents), the tie will be broken based on **weight**, which is a measure of how many times the click pattern occurred

* Boost Param

Scale Boosts

Scale the boost values to a [min,max] range

include

Solr Query parameters

Parameters for querying Signal aggregation collection

* Parameter Name	Parameter Value
qf	query_t
pf	query_t^10
pf	query_t~2^7
mm	50%
defType	edismax
sort	score desc, weight_d de