Lucidworks

LAB: Working with Signals

Prerequisites

- Register for Kaggle
  - Instructions here: Registering for Kaggle

- LAB: Spark and Data Prep

Inspecting Labs_signals

Lucidworks

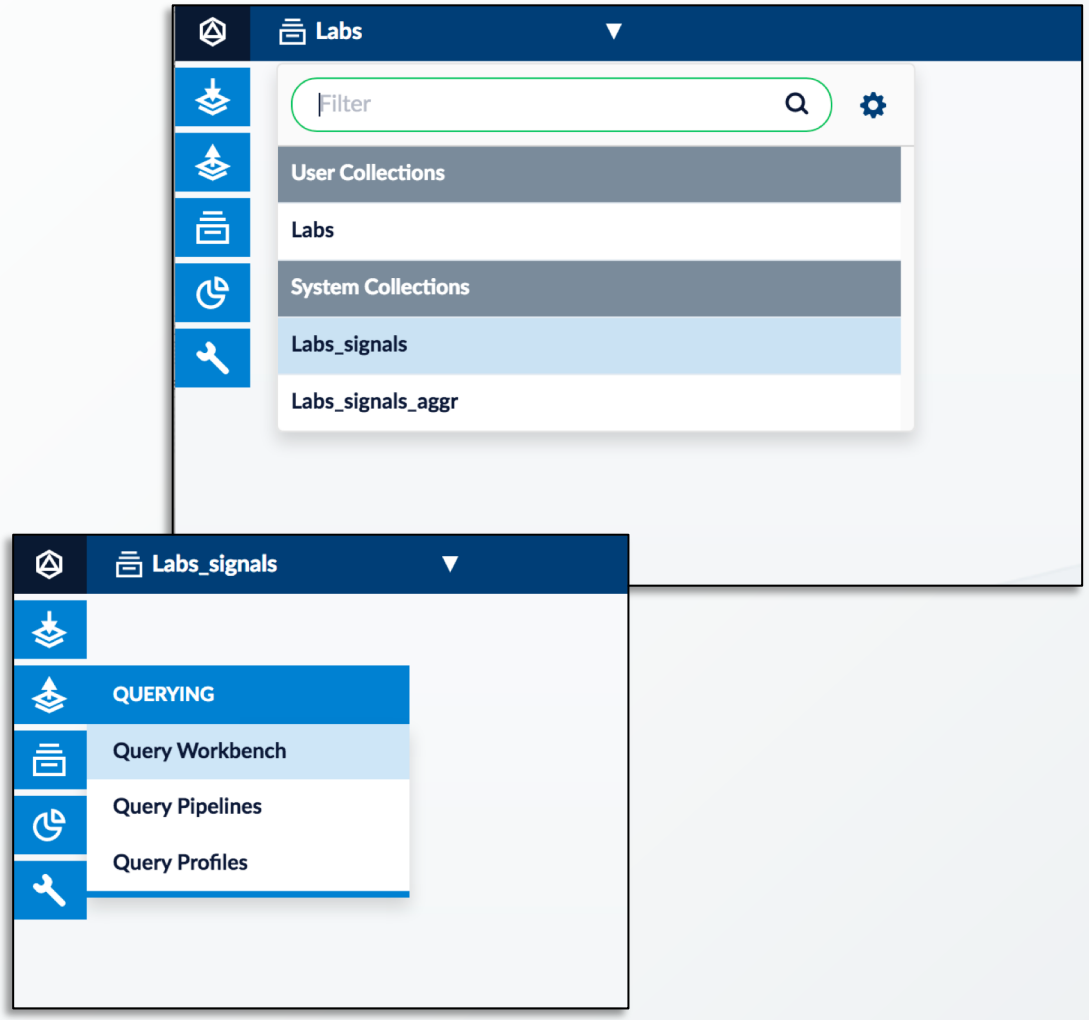
This lab assumes that you are using an AWS virtual machine provided by Lucidworks Training. If this is not the case, your filepaths and IP addresses will vary significantly from those shown.

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

- In a web browser, open Fusion Admin
  <your-vp-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 a Click Signal

Lucidworks

- 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

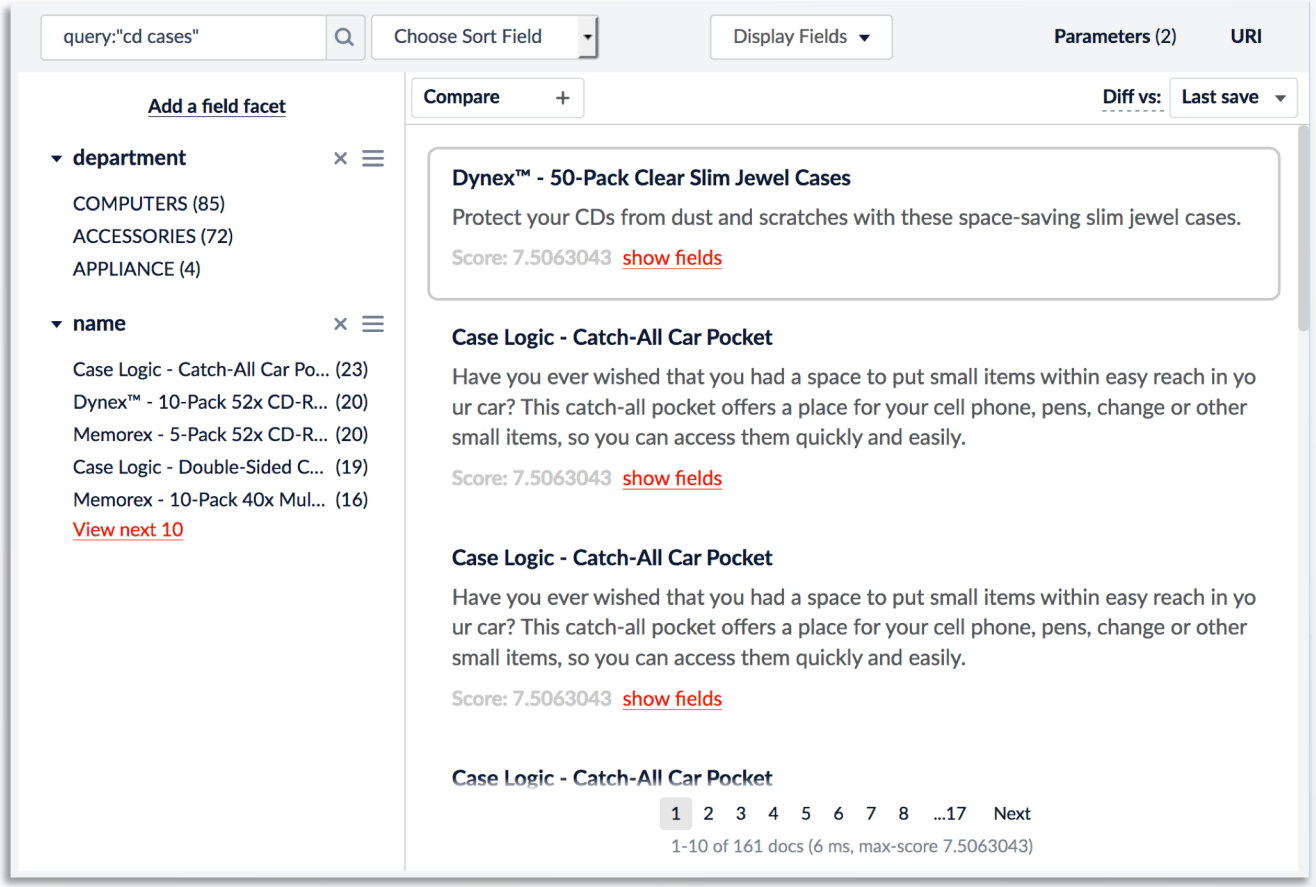
*As you can see, these signals include the baseline click data—**user_id**, **query**, action **type**, and **timestamp**—as well as extra information about the document clicked, with name, **longDescription**, and **department**.*

Exploring Labs_signals

Lucidworks

- 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*



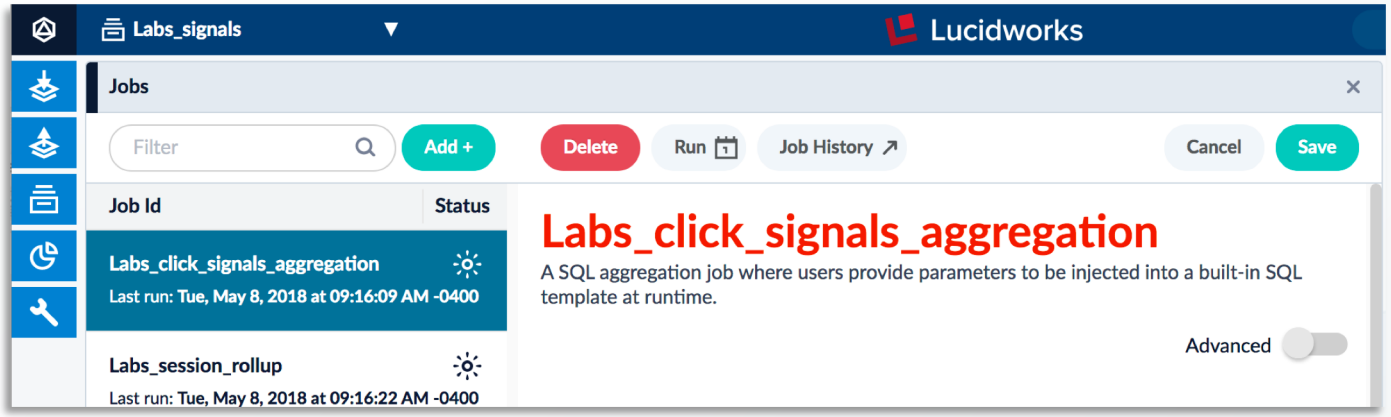
Aggregating Click Signals

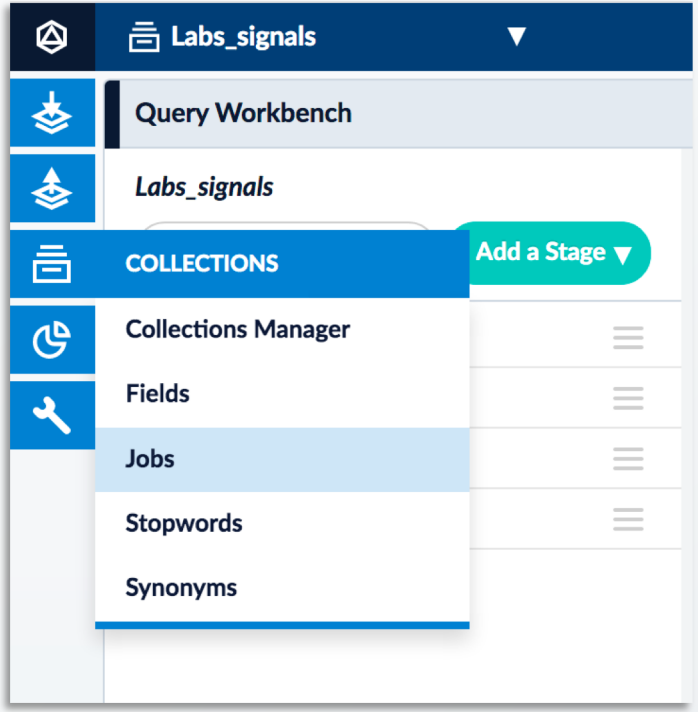
About Signal Aggregation

Lucidworks

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



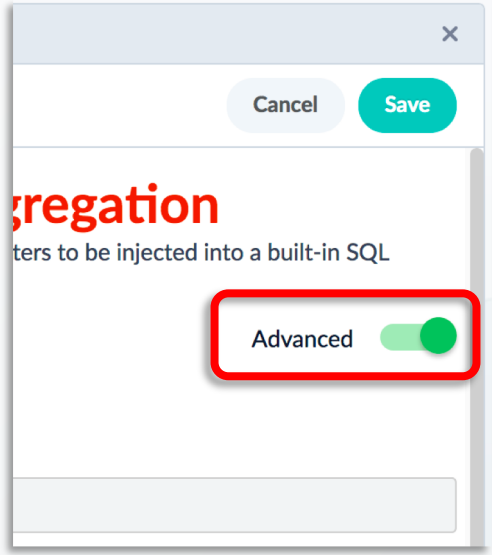


Performing Signal Aggregation

Lucidworks

- 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.*

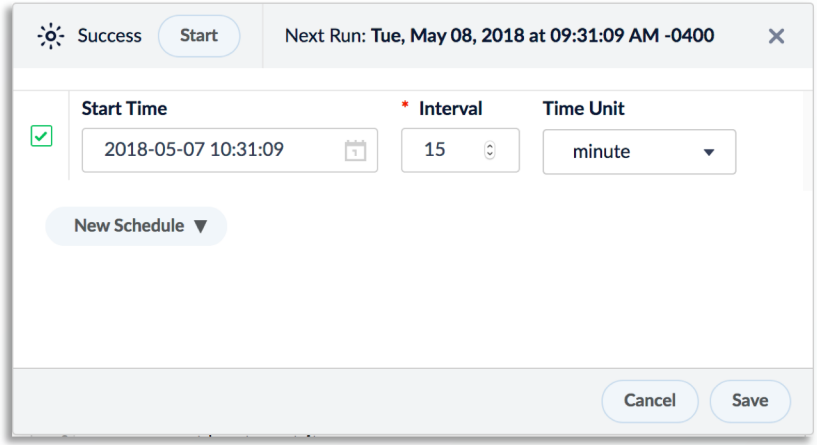


- Click **Save**

Performing Signal Aggregation

Lucidworks

- 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

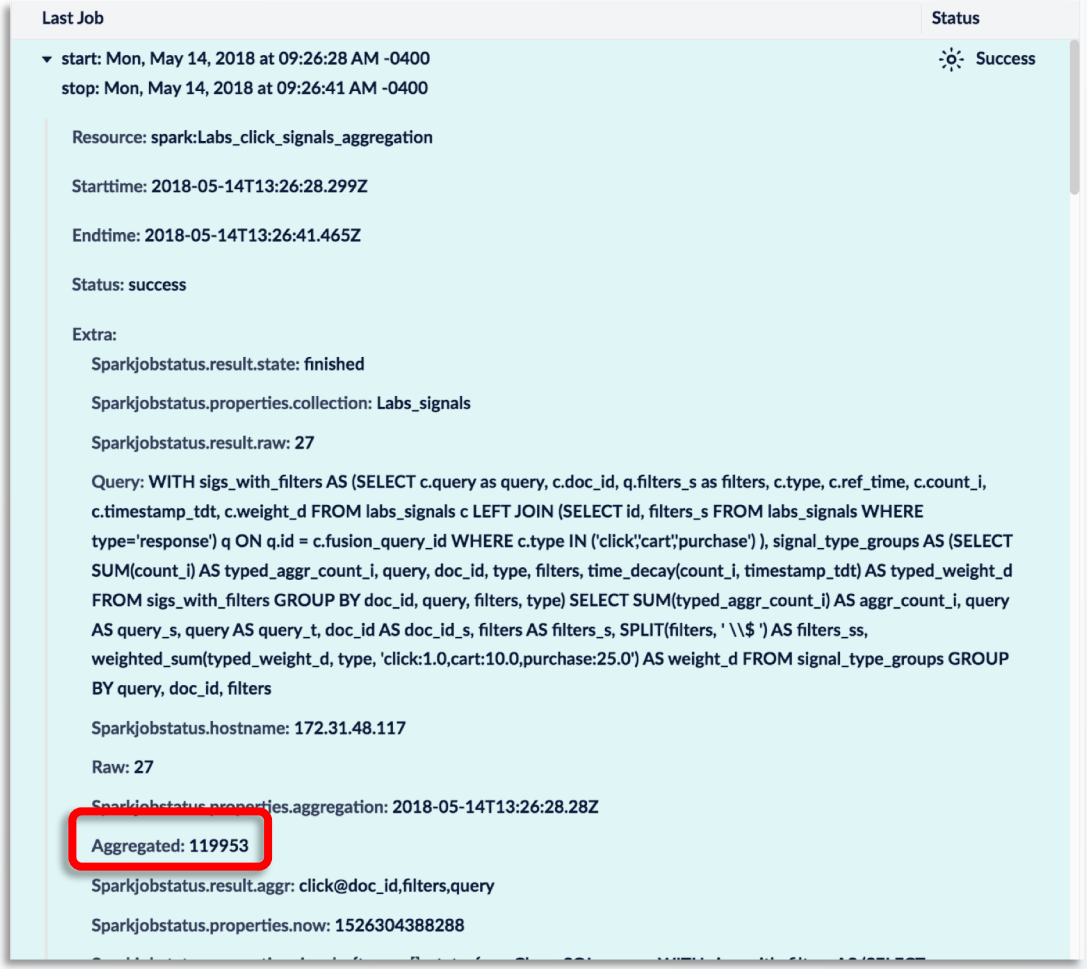
Lucidworks

- 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**

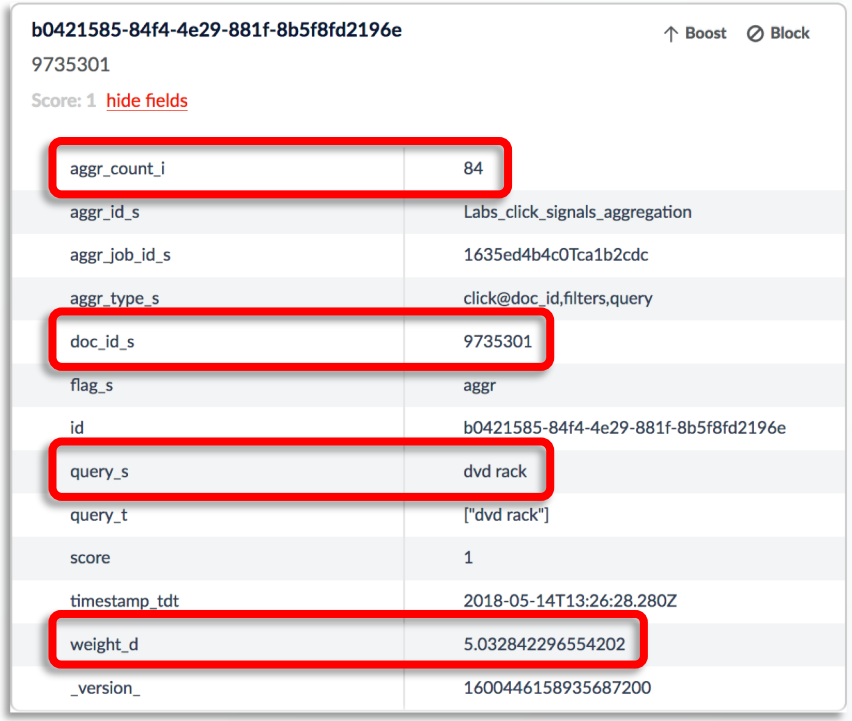
This prevents accidentally reduplicating the aggregated signals every time the job runs.



Examining Labs_signals_aggr

Lucidworks

- 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



*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)*

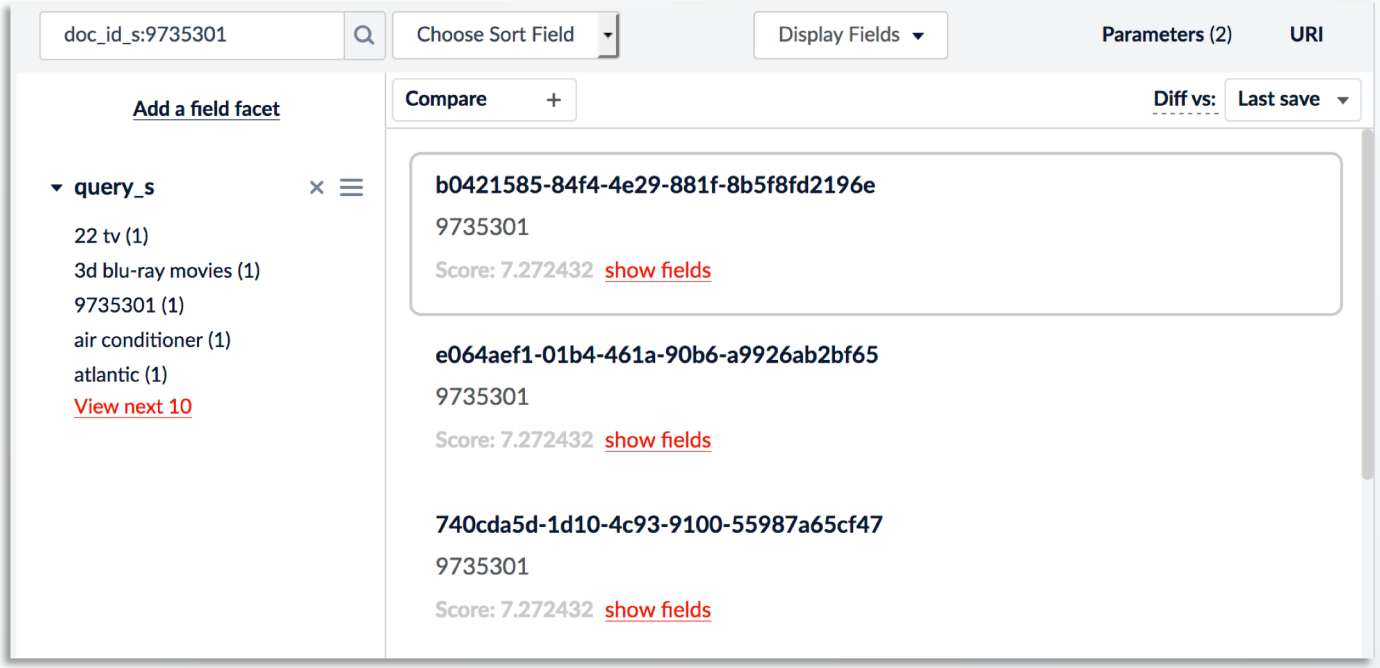
***weight_d** indicates how much to promote this document in response to this query*

Examining Labs_signals_aggr

Lucidworks

- 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



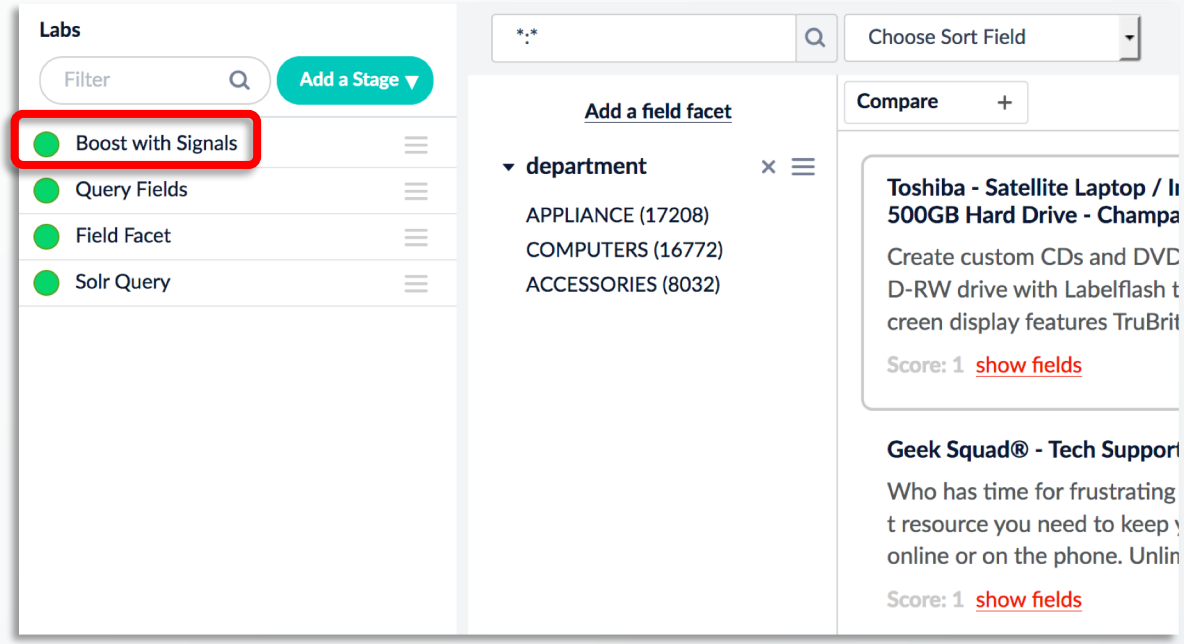
Applying Signal Boosting

Evaluating Signal Boosting

Lucidworks

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

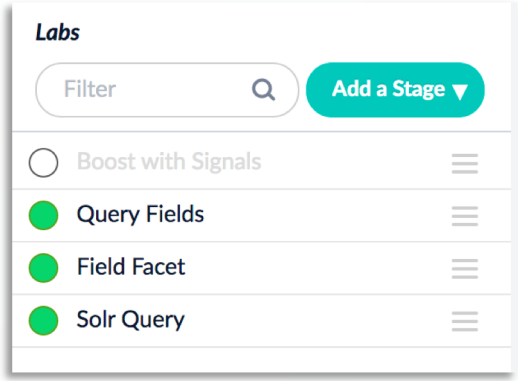
As you can see in the pipeline stages, signal boosting is enabled by default

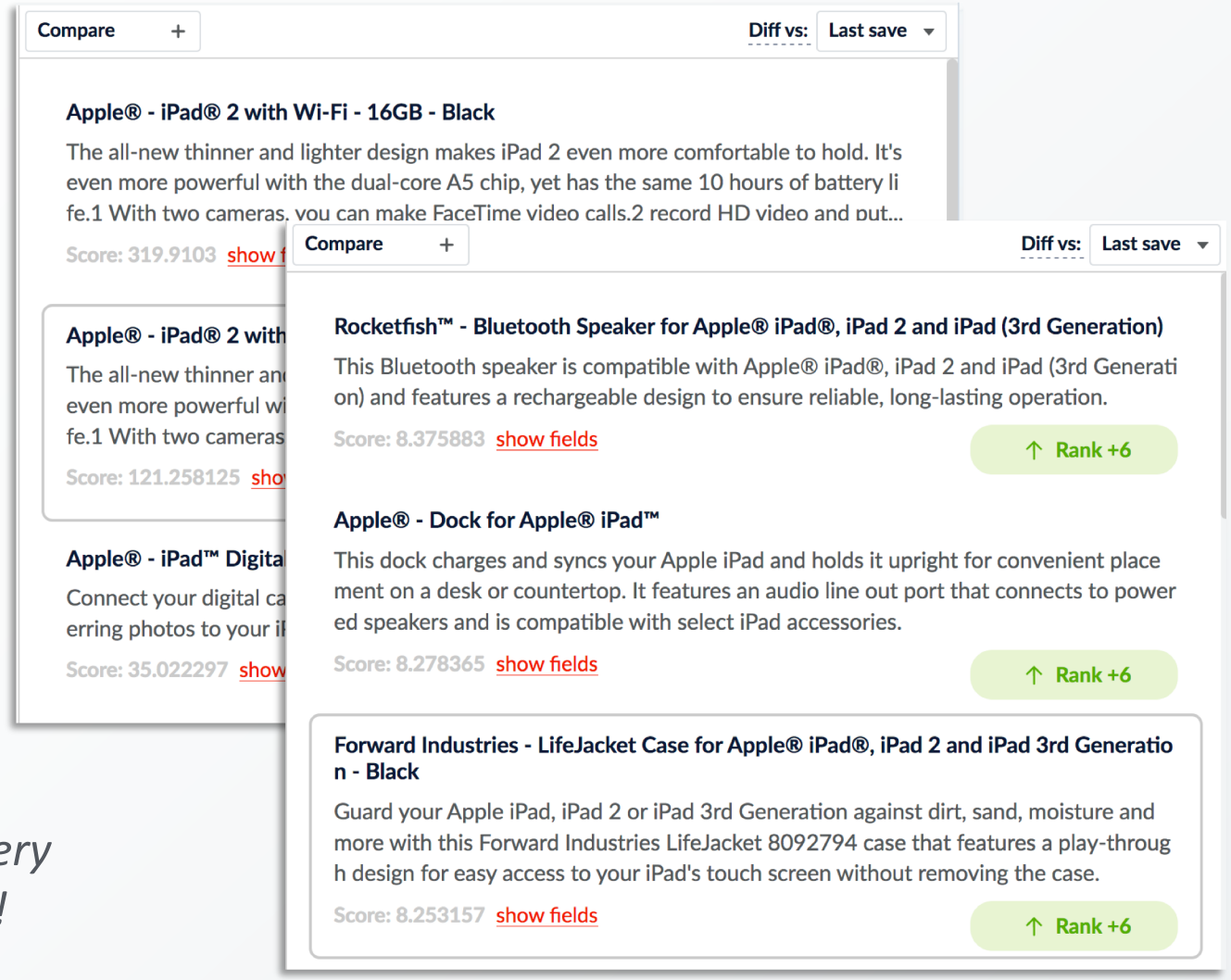


Evaluating Signal Boosting

Lucidworks

- 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!

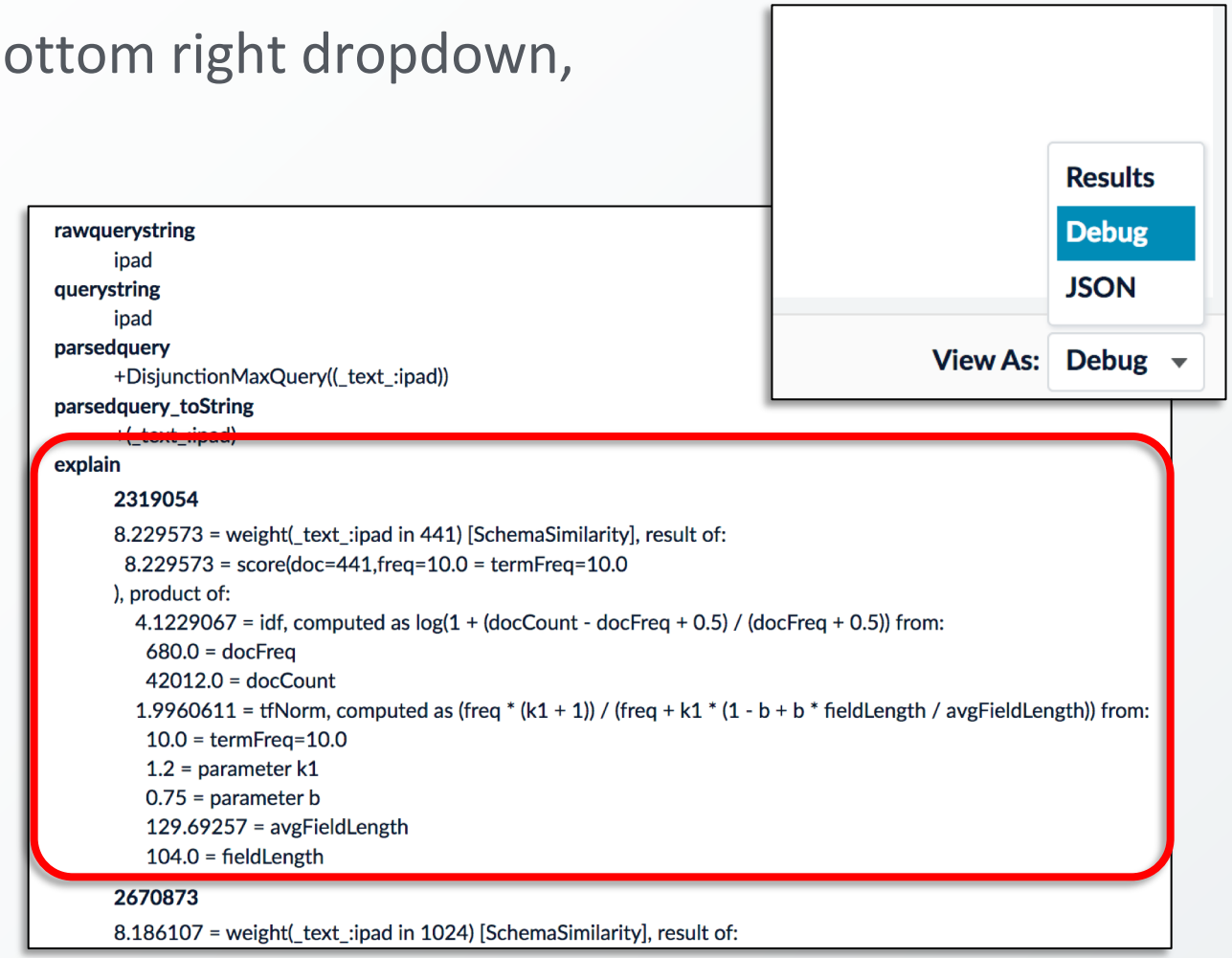
Optional:
Analyzing Signal Boosting

Analyzing Signal Boosting

Lucidworks

- 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*



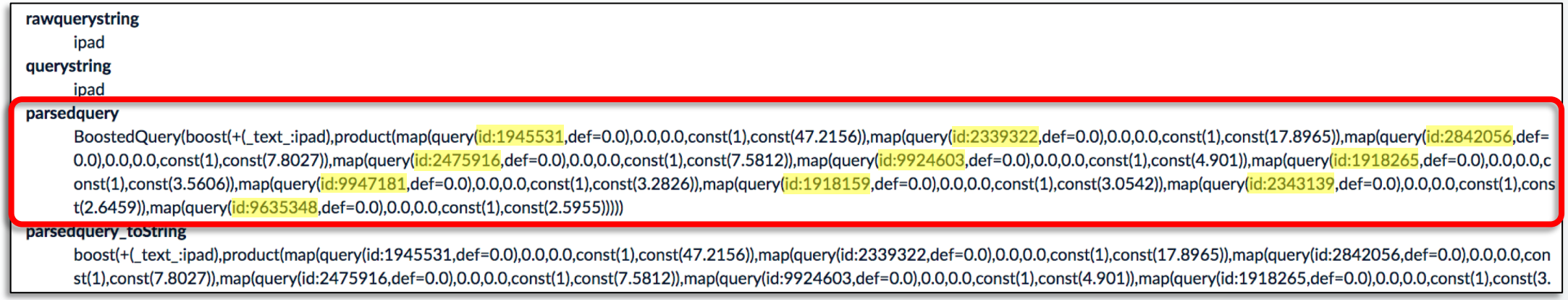
Evaluating Signal Boosting

Lucidworks

- 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*



Examining the Boost with Signals stage

Lucidworks

*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,const(1),const(47.2156)),map(query(id:2339322,def=0.0),0.0,0.0,const(1),const(17.8965)),map(query(id:2842056,def=0.0),0.0,0.0,const(1),const(7.8027)),map(query(id:2475916,def=0.0),0.0,0.0,const(1),const(7.5812)),map(query(id:9924603,def=0.0),0.0,0.0,const(1),const(4.901)),map(query(id:1918265,def=0.0),0.0,0.0,const(1),const(3.5606)),map(query(id:9947181,def=0.0),0.0,0.0,const(1),const(3.2826)),map(query(id:1918159,def=0.0),0.0,0.0,const(1),const(3.0542)),map(query(id:2343139,def=0.0),0.0,0.0,const(1),const(2.6459)),map(query(id:9635348,def=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

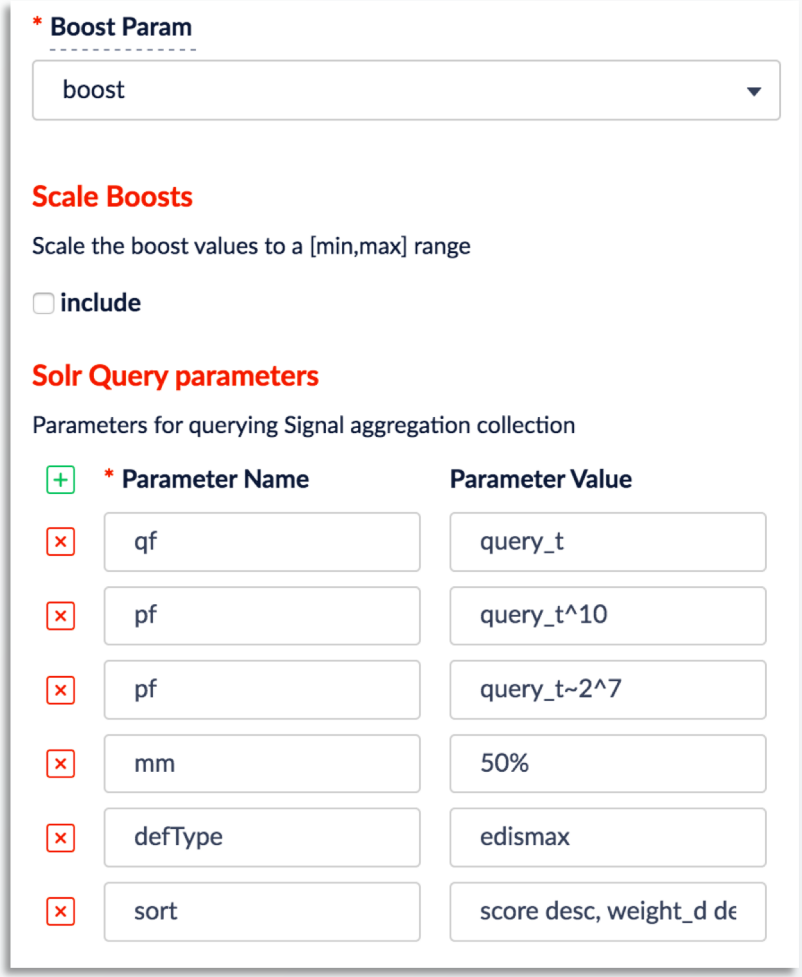
Understanding the Boost with Signals Stage

Lucidworks

- 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.



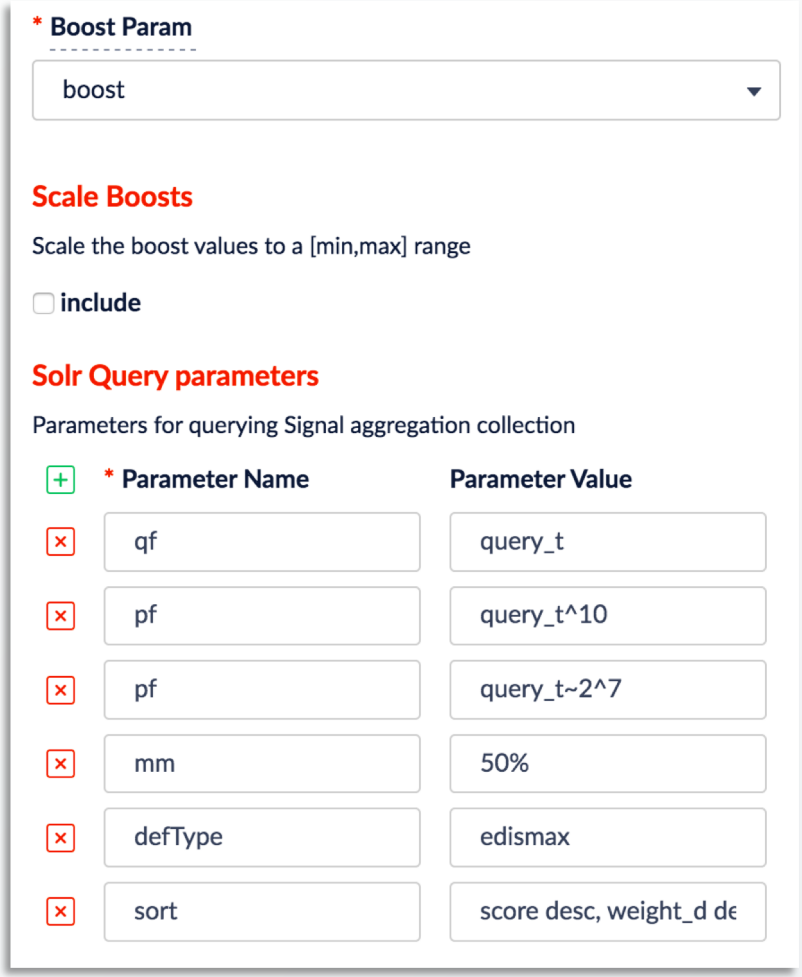
Understanding the Boost with Signals Stage, cont.

Lucidworks

- **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



Understanding the Boost with Signals Stage, cont.

Lucidworks

- 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 **weigeht**, which is a measure of how many times the click pattern occurred