Fusion Head-Tail Analysis Reveals Why Users Leave

By Chao Han, VP of Research at Lucidworks

Picture this: You go to a site and type in the query "outdoor rock speaker." You know sort of what you want, but what you see in the top pages are not rock-shaped speakers for outdoors—or even worse, there are no results at all. You bail on that retailer's site and quickly jump to another site, where you see (and buy) exactly what you wanted.

Picture the search engineer under pressure to deliver results. Lucidworks Fusion's head-tail analysis is a powerful tool for finding queries that represent missed opportunities, whether it was because everyone searched and no one clicked or because a few people searched and nothing came back.

Fusion signals and aggregations jobs capture and summarize interesting user events, such as queries, result clicks, purchases, and their associated metadata and metrics. Debuting with Fusion 4, query analytics uses signals to provide head-tail analysis, top-performed queries/tokens list, tail rewriting suggestions, and automatically generated token/phrase misspelling corrections.

Head-tail analysis introduction

Based on our experience analyzing signals, it's not uncommon to see the Pareto effect in query-related signals, that is, approximately 50% to 75% of your traffic might be coming from less than 1% of the queries. This one percent of queries, which create a lot of traffic, are canonically called the " head", and remaining queries constitute the "torso" and "tail".

For example, suppose our main event type of interest is clicks. If we group by unique queries and count the number of clicks each query leads to, head queries are the ones that lead to lots of clicks. Tail queries are the ones that lead to only a few or even zero clicks. To improve your overall conversions, configurations, product catalogs, and SEO/SEM strategies, you need to understand the head and tail distribution, discover reasons for low traffic from torso/tail queries, and find ways to rewrite tail queries to be more effective. Fusion gives you tools to gain powerful insight into your head and tail queries.

How to Run Head-Tail Analysis in Fusion

We will be using an ecommerce dataset from Kaggle 1 to show how to perform head-tail analysis based on signals.

In the **jobs manager**, add a new "Head-n-Tail Analysis" job and fill in the parameters such as follows:

Head-n-Tail Analysis

Perform head tail analysis of queries from raw or aggregated signals collection.

Advanced

* **Spark Job ID**

ecommerce-head-tail

* **Input Collection**

ecommerce_signals

* **Query Field Name**

query_s

Signals data filter query

type_s:click OR type_s:query

* **Event Count Field Name**

count_i

* **Main Event Type**

click

Filtering Event Type

query

Minimum Main Event Count

1

Minimum Filtering Event Count

20

Minimum Query Length

2

Keywords blob name

ecommerce_keyword.csv

In the configuration, the user needs to specify:

- Which collection contains the signals (the Input Collection parameter)
- Which field in the collection contains the query string (the Query Field Name parameter)
- Which field contains the count of the event (for example, if signal data follows the default Fusion setup, count_i is the field that records the count of raw signal, aggr_count_i is the field that records the count after aggregation)

The job allows a user to analyze query performance based on two different events: main event and filtering/secondary event. For example, if you specify the main event to be clicks with minimum count of 0 and the filtering event to be queries with minimum count of 20, then the job will filter on the queries that get searched at least 20 times and check among those popular searched queries to see which ones didn't get clicked at all or only a few times. If you only have one event type, leave the Filtering Event Type parameter empty.

After specifying the configuration, click Run > Start. When the run finishes, you should see the status "Success" to the left of the Start button. If the run fails, you should check the error messages in the job history. If the job history doesn't give you insight into what went wrong, you can debug by submitting the following curl command in a terminal:

tail -f var/log/api/spark-driver-default.log | grep HeadTailAnalysis:

After the run finishes, a series of tables will be output into the output collection. By default, the output collection is your input collection name plus the _suffix _signals_aggr. If your input collection name already has the _signals_aggr suffix, the job won't modify the collection name. If your input collection name ends with _signals, then the job will append _aggr at the end. An example record is as follows:

aggr_id_s	test
aggr_job_id_s	15f645fc8ebT43593355
aggr_type_s	headTailAnalysis
doc_type_s	tail_rewriting
id	e82dcb63-d2e6-4189-9b00-182683edf0d9
score	1
timestamp_tdt	2017-10-28T19:05:28.357Z
version	1582529345273987000
matched_headNoNum_txt	["surveillance camera"]
matched_headQuery_txt	["surveillance camera"]
other_specific_txt	["outdoor"]
reason_code_s	other-specific
suggested_rewriting_txt	["surveillance camera (outdoor)^1.5"]
tailQuery_txt	["outdoor surveillance camera"]
tailTraffic_pl	10

The "doc_type_s" field shows which analytics result table it is and its corresponding table in the reporting part. There are 7 of them:

- overall_distribution: Head Tail Plot
- summary_stat: Summary Stats
- queries_ordered: Query Details
- tokens_ordered: Top Tokens
- queryLength: Query Length
- tail_reasons: Tail Reasons
- tail_rewriting: Head Tail Similarity

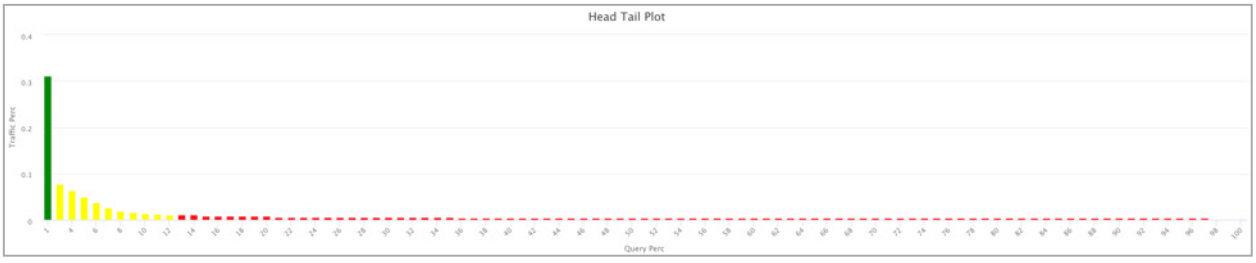
You can facet on this field and filter on the specific table you want to look into.

Head-Tail Analysis Reporting

The generated tables can be visualized and searched in our built-in Insights dashboard (Analytics > App Insights > Analytics > Head Tail analysis):

1. The head-tail distribution plot provides an overview of the query traffic distribution. In order to provide better visualization, the unique queries are in descending order based on traffic and put into bins of 100 queries on the x axis, with the sum of traffic coming from each bin on the y axis. Our backend algorithm automatically suggests where the head, torso, and tail are based on the overall distribution and labels them differently in the plot. If you would rather choose your own threshold you can specify them in the job configuration through the Head/Tail Count Threshold parameter in the Advanced panel.

   In the example plot below, we see a long tail. This indicates very little traffic coming from a large majority of queries (that is, most of our queries are not generating any clicks). If we improve the search results returned by tail queries, we will get many more clicks. Then, collectively, all the tail queries could account for a huge amount of traffic and make the distribution less skewed.



2. Summary statistics about the head-tail distribution.

   In addition to understanding the head and tail distribution, it is valuable to learn how much traffic various query groups produce. Our summary statistics table (which is configurable by the user, the configs are in Advanced area starting with "Top X% Head Query Event Count") can give you insight into possible problem areas in your search engine.

Summary Stats

Statistics	Value
total number of unique queries	22231
top 100 queries leads to ?% total events	21.24%
top 1.0% queries leads to ?% total events	31.06%
number of queries that constitute the top 75.0% of all events	3060
number of queries that constitute the top 50.0% of all events	680
number of queries that constitute the top 25.0% of all events	140
last 1.0% of total events spreaded in how many queries	1613
how many queries leads to zero events.	628
how many queries have less than 5 events	19734
events threshold for tail.	5
events threshold for head.	39

   Note: The numbers such as 100%, 25.0% and 1.0% can be specified by the user in the job configuration (Advanced panel) to answer these specific questions.

3. Query length and token number distribution.

   Another point of interest is learning how users are querying your database. Are most people searching very long strings or very short strings? These distributions will give you insight into how to tune your search engine to be performant on the majority of queries.

Query Length

Avg String Length	One Word Perc	Two Words Perc	Three Words Perc	Four Words Perc	Five Words Perc	Six+ Words Perc
13	17.89	40.06	27.01	10.07	3.36	1.61

4. Detailed Query/Token Traffic Table

   It is also valuable to learn which specific queries are producing the most and least traffic. Our Detailed Query Traffic Table will help you discover which queries are your best performers and which are the worst. You can filter results by issuing a search in the search bar. For example, search "segment_s:tail" to get tail queries or

search “num_events_l:0” to get zero results queries. (Note: field names are listed in the “what is this” toolkit which you can find by hovering over the ? on the UI).

Query Length						
Avg String Length	One Word Perc	Two Words Perc	Three Words Perc	Four Words Perc	Five Words Perc	Six+ Words Perc
13	17.89	40.06	27.01	10.07	3.36	1.61

The “Top Tokens” table lists the number of times each token shown in the queries.

Top Tokens	
Tokens	Count
tv	675
samsung	609
sony	543
case	514
laptop	476
iphone	442
dvd	396
ipod	395
hp	376

Automatic Tail Reason Investigations and Tail Query Rewriting

As we mentioned before, if we could improve the search results for the tail query we would vastly increase our click-through rate (CTR). Typically tail queries underperform because they are a misspelling, or a less frequent variation, on a more popular head query. So, by replacing, or rewriting, a tail query to its corresponding head query we may be able to improve the returned search results for the tail query. Fusion can help you do this.

For each tail query, we try to find its closest matching head queries, and based on the difference between the tail and head queries, we can assign reasons for why any given query is a tail query and prescribe an improvement. Based on our observations on different signal datasets, we summarize tail reasons into several predefined categories while also allowing users to specify their own attributes through a keywords file.

The predefined reasons are as follows with examples in the table below:

1. Spelling: Query contains one or more misspellings; we can apply spelling suggestions based on the matching head.

2. Number: Query contains an attribute search on a specific dimension. To normalize these queries we can parse the number to deal with different formatting, and/or pay

attention to unit synonyms or enrich the product catalog. For example, "3x5" should be converted to "3' X 5'" to match the dimension field.

3. Other-specific: Query contains specific descriptive words plus a head query, which means the user is searching for a very specific product or has a specific requirement. We can boost on the specific part to have better relevancy.

4. Other-extra: This is similar to 'other-specific' but the descriptive part may lead to ambiguity, so it requires boosting the head query portion of the query instead of the specific or descriptive words.

5. Rare-term: This is a scenario where the user is searching for a rare item which would require caution with boosting.

6. Re-wording: Query contains a sequence of terms in a less-common order. Flipping the word order to a more common one can change a tail query to a head query, and allows for consistent boosting on the last term in many cases.

7. Stopwords: Query contains stopwords plus head query. We would need to drop stopwords.

Head Tail Similarity

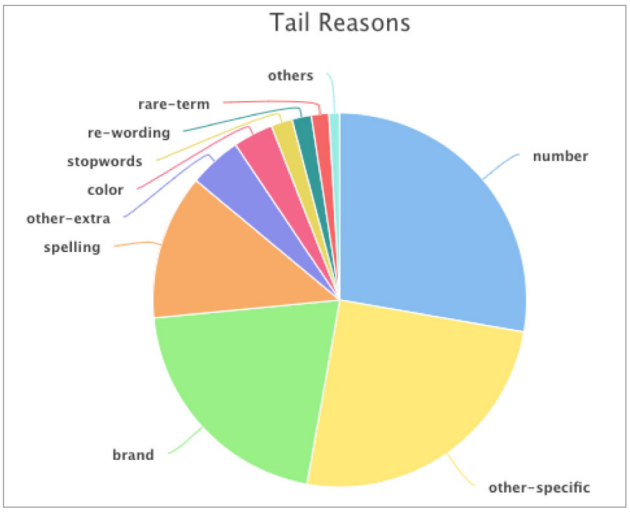
Tail Query	Suggested Rewriting	Matched Head	Reason Code	Matched Head (No Num)	Number	Other Specific	Number Unit	Tail Traffic
chair game	game chair^2	game chair	re-wording	game chair				6
itouch3	itouch 3 itouch3	itouch	number	itouch	3		itouch3	4
laptop hard drive case	(laptop case)^2 hard drive	laptop case	other-extra	laptop case		hard drive		2
outdoor rock speaker	outdoor speaker (rock)^1.5	outdoor speaker	other-specific	outdoor speaker		rock		2
kareokee machine	karaoke machine	karaoke machine	spelling	karaoke machine				2
black ge fridge	fridge ge black OR brand:ge OR color:black	fridge	brand color	fridge				2
dvd r rw	dvd rw	dvd rw	stopwords	dvd rw				2

Users can also customize these results by providing their own dictionary in a CSV file (ecommerce_keyword.csv as in the configuration example) with items such as color, brand, and question. If these are specified, the job will parse the tail query and assign reasons such as "color" or "brand" (as specified in your dictionary) and perform filtering or focused search on these fields. Here is an example of the dictionary.

keyword	type
a	stopword
an	stopword
and	stopword
blue	color
white	color
black	color
hp	brand
samsung	brand
sony	brand

Note: The header of the CSV file must be called "keyword" and "type", and stopwords must be called "stopword" for the program to recognize them. Please upload the CSV file to the blob store and fill in the name shown in blob store to the "Keywords blob name" parameter in the configuration.

A pie chart will be provided to show the proportion of each reason code. Knowing which kind of attributes cause bad search performance can provide guidance on search field boosting setup and ways to enrich the catalog on the corresponding fields.



Note: The suggested rewrites are presented in the "suggested_rewriting" field in the above query rewriting table. Users can change the rewriting approach that fits their needs based on elements provided in other fields, such as the matching head query, color, brand, number, other-specific, term-correction, and number_unit.

Users can also apply the tail query rewriting part at query time. For example, a query parser can be set up to find color, brand, and number, then apply special treatments that are similar to the suggested rewritings. Misspellings and corrections can be put into the Solr synonyms list to perform corrections. If after dropping color, brand, and number, an incoming query matches any tail queries in the rewriting list with reason codes "other-specific" , "other-extra" or "re-wording" , then we can apply the suggested rewriting where different parts of the query get boosted.

Conclusions

The tail query problem is a common issue in most search applications. Fusion provides a unique solution to improve relevancy and recall for tail queries by finding connections to head queries and apply different search strategies based on different reasons. The insights from our dashboard can also provide guidance to help find the root cause and to improve both Solr config and catalog mismatch from queries. Give it a try!