Getting Started With Elastic Search

Lets start with an understanding of other search processes with a standard SQL search engine built into a website for example – our first issue is query calls have to be super specific in writing or they will not even work –

Next – we might have to make multiple query’s just to get down order of words case and other issues that make it easy for your user to miss out on there search’s

Last- even if you finally build a robust set of query’s that gets everything you want it will probably also end up getting you a bunch of things you don’t want.

WHY – SQL is a database not a search engine – elastic search is a search engine it is for information retrieval.

How can we get started with elastic search –

Download it and then run it through your terminal with a curl request to get the whole thing up and running from the command line

Now to use you want to set up a shemea to use a config to search – ids – dates – locations

You can specify things like strings and index’s – but also through analyzers you can tell it what it should expect to be in the English language – this will all be built up as a Jason file in the end

Then can build iterative events to or batch calls to move all the data you need into elastic search

Simplest way to search this new data is the match all query

You get back time it took and all docs that match no order on match with this yet

Next using a term query can create a filter to match a certain form a English text

The match phrase query can be used to match phrases – of your target language or you can make a linked query of several phrases to match and get your results that way

These query’s can be made more complex with Booleans that allow use to use - must = and – should = or -finally - must not = not

This helps you find thing’s easier – next lets talk relevance – TF – term frequency – IDF inverse document frequency.

Term frequency is how often a term occurs / inverse document frequency is how often the term is found among certain documents.

When is it good to use elastic search and when is it not –

1 Getting data analysis in

This is called analysis – tokenization is the first thing we do when dealing with English – next casing is big as far as getting the right data and then Stop Wording is the last thing that we are concerned with – Stemming is that last tool chops up words to get ends and beginnings

2- indexing – what happens is that elastic search cuts everything up and says these documents have these tokens – effectively a search engine full query returns results in the form of a python dictionary.

3- Search supports AND OR – AND OR – so it has many forms of data retrieval to match your needs

4- For prioritization of results you have a priority queue – remember this makes search’s much longer to return results so it is non ideal for search’s where you expect to return 10,000 results you can set it up to only do ten for a top ten list and then nothing else is in a priority order they are just worse than top ten

5- Getting data out aggregation – for all results return interesting information add to aggregator

Return results from aggregator.

The results work well for general aggregation but on top of that if you turn this data on it’s side you get really nice histograms as well. This information can also be great in spark charts to see how many search’s are launched in a certain day – you can get tons of analytics about search’s from elastic search

Aggregations can be nested arbitrarily if you want –

This will lead to data buckets for your primary search with sub buckets for your other arbitrarily set nested aggregation search’s like city – towns and people.