



|  |
| --- |
| ElasticSearch  2019 |
|  |
| 15 diciembre  Information repositories – Uniovi 19-20  Written by: Óscar Sánchez Campo (UO265078)  Daniel Finca Martínez (UO264469) |

# Prologue

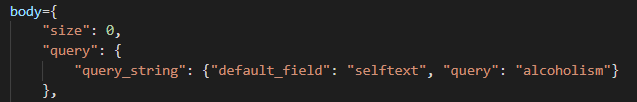
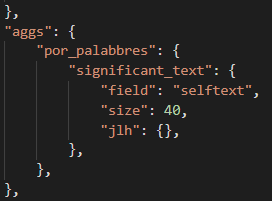
For this project, the provided script “bulkIndexer4.py” has been used to generate the index. Process such as stemming, removing stopwords and n-grams generation are used in such script.

Note: Lowercase operation has been commented out and the reason why this has been done will be explained later on (Exercise 3).

# Exercises

## Exercise 1

In this exercise “alcoholism” topic has been chosen. As an initial query to retrieve significant terms, a simple structure has been used as shown here:



To expand the results provided with such request, Elasticsearch query language aggregations feature is added to above body as stated on the image to the right. Using this, most common words used in posts retrieved from above results are grouped by frequency. These new outcomes then serve as the expanded set of terms about the topic chosen for the theme of the exercise.

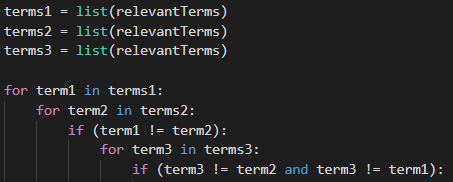
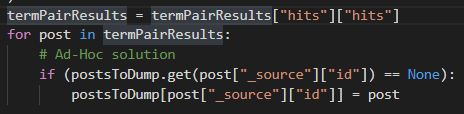
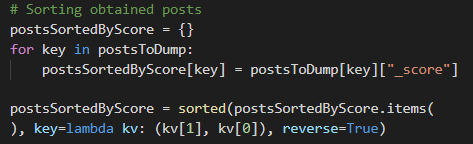
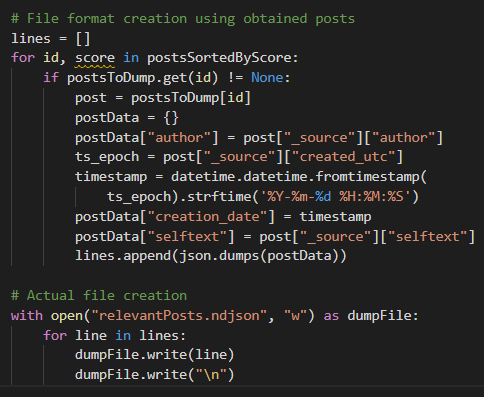
In addition to above aggregation snippet, different similarity metrics have been used. These are gnd, chi\_square and jlh. Different aggregation sizes have been tested to assess their performance. Jlh metric was used with above query. This will be explained later on.

A table can be found below with the relation of the metrics used to the number of words obtained to query the index. To our criteria, jlh has one of the highest precision rates obtained from execution experience.

|  |  |  |  |
| --- | --- | --- | --- |
| Similarity metric  to  Aggregation size | GND | JLH | Chi-square |
| 20 | 7 words – 85% | 7 words – 100% | 7 words – 100% |
| 30 | 8 words – 95% | 8 words – 100% | 9 words – 100% |
| 40 | 13 words – 100% | 14 words – 100% | 13 words – 95% |
| 50 | 16 words – 95% | 17 words – 95% | 17 words – 95% |

As mentioned above, in jlh metric has been used to obtain the results (cell marked).

From the exercise statement, it can be read that the set of posts related to the chosen topic must be obtained and extracted to a file. To do this, these steps have been followed:

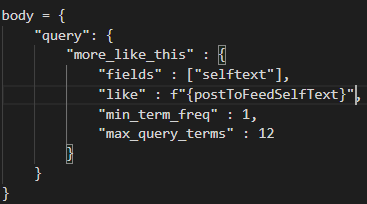
1. Significant terms are taken out from the aggregation previously shown.
2. Then, these are combined in triplets to query the index. In this way, the posts that talk about such combination of terms are retrieved. These tuples are created using three for-loops which avoid same term being queried at the same time, hence being always different.
3. As triplets are generated, they are fed to an Elasticsearch query that looks like this: 
4. As different combinations might generate repeated posts as outcome, this part of the script is used to avoid such issue. Now a dictionary with id to post relation is on our hands to process.
5. Once the previous relation is obtained, the posts are sorted based on their relevance score; using yet again a fresh new dictionary. This can be used whenever the nth most relevant posts want to be obtained.
6. Once sorted, format and output of the posts can be produced. Author, selftext and creation date have been added to the output to comply with the statement of the exercise. This output has been written to an ndJson file. Each line represents a post in json format obtained from stringifying python dictionary objects; by means of json package.

## Exercise 2

In this exercise, the requested task asks for a suggestion on how to implement “*More like this*” queries. This should pretty much simulate the behaviour of gnd similarity metric used in exercise 1. Somewhat alike to what google does when searching any term, suggesting new complete searches like the one on the input.

On top of the terms found from a query to expand relevant terms to our topic, we can feed in the text of the posts found to obtain related ones. This would be essentially the same as doing it using the terms to feed into the “like” parameter. Anyways, here is an explanation of kind of a pseudocode that might be used.

These are the series of steps followed:

1. To obtain relevant documents for the selected topic a single and normal search query is used. To limit the results, the nth most relevant ones might be extracted.
2. Once done, they can be fed into an Elasticsearch request that uses more\_like\_this dsl’s feature. Where the text of each obtained post from step 1 is introduced into the “like” parameter. The syntax was obtained from the links provided in some document from the assignment material. These queries are easy to create using python f”” formatting syntax as shown in the picture included. Where postToFeedSelfText could be a loop iterator taking out text from previously retrieved posts.