Sofia Kyriazi
S1790986
Information retrieval: report2 - week 6-7
25/10/2016

Hypothesis

A custom analyzer that uses a pattern in order to separate the document to tokens*, and with the filters of stemming in dutch, turning to lower_case and shingle combination of pairs (min:2 max:2) has a better performance than the default analyzer of elasticsearch.

*Separators are considered to be whitespaces, symbols, uppercase, lowercase, digits.

Analyzer + Mapping

Before taking on this approach, the previous approach was to use the ngram and the edge Ngram tokenizer for the document. Both of which seemed to fail in producing good scores but both had a good step between the scores in the sequence of documents, which lead to the assumption that the shorter the words the simpler it was to detect some similarity between the query and the original document. To continue with, in the documentation of the elasticsearch, the camel analyzer manipulates a string that has similarities to the documents in the collection.txt. This motivated the development of the following analyzer. Again, some filters were set to produce even better clarity in the results avoiding long words, such as stemming in the Dutch language, the lowercase was also used in order to avoid mistakes with capital cases from the queries side. Another filter that was used was the shingle filter to create pairs of stemmers of the original words to increase the accuracy in matching. Using the hypothesis that combinations of dimensions, for example, would be more accurate to the real result than detecting a single number in the dimensions of the original document. The asciifolding filter was used to replace any characters that don't belong to the standard dictionary.
The mapping is the analyzer for the one field that is used to represent the total document, named pdesc.
This is how the analyzer looks:

```
{
  "settings": {
        "number_of_shards": 1,
        "analysis": {
        "tokenizer": {
        "ngram_tokenizer": {
        "type": "pattern",
                "pattern":
"([^\\p{L}\\d]+)|(?<=\\D)(?=\\d)|(?<=\\d)(?=\\D)|(?<=[\\p{L}&&[^\\p{Lu}]])(?=\\p{Lu})|(?<=\\p{Lu
})(?=\\p{Lu}[\\p{L}&&[^\\p{Lu}]])"
        }
        },
        "analyzer": {
        "ngram_tokenizer_analyzer": {
        "type": "custom",
```

```
            "tokenizer": "ngram_tokenizer",
            "filter": [
                    "lowercase","asciifolding","my_stemmer","other_filter"
            ]
            }
            },
            "filter" : {
                    "my_stemmer" : {
                    "type" : "stemmer",
                    "language": "dutch"
                    },
                    "other_filter":{
                    "type":"shingle",
                    "max_shingle_size":2,
                    "min_shingle_size":2,
                    "output_unigrams":"true"
                    }
            }
            }
  },
  "mappings": {
            "products": {
            "properties": {
            "pdesc": {
            "type": "string",
            "term_vector": "yes",
            "analyzer": "ngram_tokenizer_analyzer"
            }
            }
            }
  }
}
```

The results from this execution are the following:

map all 0.4067	recip_rank all 0.406667
P1 all 0.310000	S1 all 0.310000
P5 all 0.101600	S5 all 0.508000
P30 all 11.133333	S30 all 0.668000
P1000 all 11.133333	S1000 all 0.730000