Sofia Kyriazi
S1790986
Information Retrieval: report 3 - week 6-7

Hypothesis

1.We divide the document to p-code, p-desc, p-dimensions. 2.A camel analyzer with a pattern to separate strings to tokens in every symbol, whitespace, change of case, digit has a better performance if it applies to the different p-fields we provide above compared to the baseline search, default by elasticsearch and to only one field that contains the whole document. 3. Finally, we apply a bool type of query that requests matches on each field and sets a weight to each match.

Preparation Phase

Observing the given collection of the documents, we can notice the following and make assumptions:

- The documents do not follow a specific order: meaning that dimensions, characteristics, product names are not in order.
- The queries and the documents are not following a specific pattern, but they use lower and uppercase in the names. Meaning that for this it would be better to set a filter on elasticsearch to convert everything to lowercase.
- We capture the dimensions with a script applying regular expressions in python, in the following patterns that were detected.
    - NumberXNumber or numberxnumber (m, mm, kg, g, l, ml, kl)
    - Number (m, mm, kg, g, l, ml, kl)(with or without whitespace)
    - Word = Number (m, mm, kg, g, l, ml, kl) (with or without whitespaces)
- Python segment:
    - strs.append('x'.join(re.findall(r'\w+\s*mm\b',a)))
    - strs.append('x'.join(re.findall(r'\w+\s*mt\b',a)))
    - strs.append('x'.join(re.findall(r'\w+\s*MM\b',a)))
    - strs.append('x'.join(re.findall(r'\b\d+x*\d+\b',a)))
    - strs.append('x'.join(re.findall(r'\b\d+X*\d+\b',a)))
    - strs.append('x'.join(re.findall(r'\b\d+\s*mm\b',a)))
    - strs.append('x'.join(re.findall(r'\b\d+\s*MM\b',a)))
    - strs.append('x'.join(re.findall(r'\w+\s*kg\b',a)))
    - strs.append('x'.join(re.findall(r'\w+\s*KG\b',a)))
    - strs.append('x'.join(re.findall(r'\w+\s*ml\b',a)))
- strs.append('x'.join(re.findall(r'\w+\s*ML\b',a)))
    - strs.append('x'.join(re.findall(r'\w+\s*gr\b',a)))
    - strs.append('x'.join(re.findall(r'\w+\s*GR\b',a)))

  This way we find a combination of the appearances of the above mentioned patterns and combine them in a string to assign them to the p-dimension variable. The best separation though it might not work for all the documents would be to isolate such cases and create a different field inside the document with name “p-dimensions”

Sofia Kyriazi
S1790986
Information Retrieval: report 3 - week 6-7

- In our effort to detect codes inside the documents, we follow the following basic algorithm.
    - If a word contains a combination of characters along with digits or the appearance of symbols and that is the largest appearance of that combination in the document then that is considered to be a code.
    - That values is then assigned to the p-code field of the document.
- Python
    - not(i.isdigit()) and not(i.isalpha()) and len(code)<len(previous code)
- Codes are usually sequences of letters with digits mixed and it would be better if we could use another field to capture the code in the document as pcode.

Analyzer Phase

In this phase the camel analyzer as was mentioned in the elasticsearch documentation was used:

```
        "analyzer":
        "camel":
        "type": "pattern",
        "pattern":
"([^\\p{L}\\d]+)|(?<=\\D)(?=\\d)|(?<=\\d)(?=\\D)|(?<=[\\p{L}&&[^\\p{Lu}]])(?=\\p{Lu})|(?<=\\p{Lu
})(?=\\p{Lu}[\\p{L}&&[^\\p{Lu}]])"
```

Mapping of test case 1

```
        "mappings": {
        "products": {
        "properties": {
        "pdesc": {
        "type": "string",
        "term_vector": "yes",
        "analyzer": "standard"
        },
        "pcode": {
        "type": "string",
        "term_vector": "yes",
        "analyzer": "camel"
        },
        "pdim": {
        "type": "string",
        "term_vector": "yes",
        "analyzer": "camel"
```

Sofia Kyriazi
S1790986
Information Retrieval: report 3 - week 6-7

This was followed to keep the search at least at the same performance with the baseline search given that the p-desc contains all the document, so no harm could be done in the search. The main hypothesis is that the user would give more emphasis on the code and the dimensions while searching and so the analyzer better be applied to the pcode and pdim fields.

Search Phase test case 1

After uploading the documents with the aforementioned format, we execute the query script which transforms the question in the same way as the documents were processed and the parameters of the query are set up in the following format. Giving more weight to the code and the dimensions of the document, rather than the description:

```
"sort":["_score"],
"size": 100,
"query": {
"bool": {
"should":
        "match": {
        "pdesc": {
        "query": terms,
        "boost": 1

"match": {
"pcode": {
        "query": code,
        "boost": 2

"match": {
"pdim": {
        "query": dim,
        "boost": 2
```

Results Phase test case 1

The above search produced the following results

```
map    all    0.1799
P1    all    0.136000
P5    all    0.044000
P30    all    6.066667
P1000    all    6.066667
recip_rank    all    0.179871
S1    all    0.136000
S5    all    0.220000
```

S30 all 0.364000
S1000 all 0.466000

Due to our unexpected results, there had to be a new test checking if a change in parameters would make a difference in the hypothesis.

Mapping test case 2

Setting camel analyser to all the fields.

```
"mappings": {
"products": {
"properties": {
"pdesc": {
"type": "string",
"term_vector": "yes",
"analyzer": "camel"
},
"pcode": {
"type": "string",
"term_vector": "yes",
"analyzer": "camel"
},
"pdim": {
"type": "string",
"term_vector": "yes",
"analyzer": "camel"
```

Search Phase test case 2

In this case of the

```
"query": {
"bool": {
"should": [
{
        "match": {
        "pdesc": {
        "query": terms,
        "boost": 2
"match": {
"pcode": {
        "query": code,
        "boost": 1
"match": {
"pdim": {
```

```
            "query": dim,
            "boost": 1
```

Results Phase test case 2

We run the evaluation script and in this case the results are the following:

```
map    all    0.2854
P1    all    0.218000
P5    all    0.071200
P30    all    8.066667
P1000    all    8.066667
recip_rank    all    0.285376
S1    all    0.218000
S5    all    0.356000
S30    all    0.484000
S1000    all    0.570000
```

We can see that this produces an improvement compared to the baseline search

Suggestions/Improvements

The hypothesis starts with a logical approach on giving weights to the product id and code but in practice it doesn't seem to work in the expected way. This can lead to some assumptions such as not knowing whether the user inserted the queries giving more importance to codes and dimensions, or the standard analyzer doesn't work well with the codes and dimensions.

The prefered strategy to follow for this approach would be to use different analyzers, for the pcode and pdim as in a combination of the camel and the shingel and the camel only for the descriptions, but during the development, I had problems with having two analyzers in the same settings for one index.