<u>IR Challenge - Raz Ben Aharon</u>

In this task, we were given an index with the following stats:

```
Repository statistics:
documents:       528155
unique terms:    664605
total terms:     253367449
```

The index includes stopwords.

We also got relevance judgments for the first 50 queries.

The first thing I did was run a grid search on some of the retrieval algorithms we learned in class, using Indri — specifically OKAPI BM25 and Dirichlet smoothing. The goal was to find the best hyperparameters based on MAP score. I ended up running 4 different searches, both with and without pseudo relevance feedback.

The top MAP scores were:

MAP	Configuration
0.2339	train_dir_mu1000_terms30_w0.6
0.2333	train_dir_mu1000_terms30_w0.4
0.2328	train_dir_mu1000_terms20_w0.6
0.2328	train_dir_mu1000_terms20_w0.4
0.2312	train_dir_mu1000_terms10_w0.4
0.2311	train_dir_mu1500_terms30_w0.6
0.231	train_dir_mu1500_terms30_w0.4
0.2308	train_dir_mu1000_terms20_w0.8
0.2308	train_dir_mu1000_terms10_w0.8
0.2305	okapi_no_prf_k10.8_b0.5
0.2304	train_dir_mu1000_terms30_w0.8
0.2304	train_dir_mu1000_terms10_w0.6
0.2298	baseline_dir_mu1000
0.2297	okapi_no_prf_k11.2_b0.5

Conclusions:

Dirichlet with pseudo relevance feedback gave the best MAP scores on the training set across all four models.

In addition, pseudo relevance feedback didn't really help OKAPI BM25 — in fact, it often made things worse.

Final hybrid model

The next step I took to improve the model was to try combining BM25 and Dirichlet.

The idea is to use Dirichlet with pseudo relevance feedback to extract the highest-probability terms from the most relevant documents — specifically, by building a feedback language model based on the top-ranked results from the initial query retrieved using the Dirichlet language model. These top terms, which are expected to capture the main concepts of the query, were then used for **query expansion** in BM25.

For each query (301–350), I ran the following Indri command to retrieve the top 50 terms from the top 10 documents using pseudo relevance feedback (PRF).

```
<rule>method:dir,mu=1000</rule>
<fbDocs>10</fbDocs>
<fbTerms>50</fbTerms>
<fbOrigWeight>(varied)</fbOrigWeight>
<printQuery>true</printQuery>
```

I then filtered out stopwords using NLTK’s English stopword list, as well as any words shorter than 3 letters. From the remaining terms, I selected the top 20 most relevant ones and renormalized their weights, so they sum to 1.

Next, I ran OKAPI BM25 using the best parameters I found without PRF: **k_1 = 0.8, b = 0.5**.
I added query expansion using a mix where **60%** of the weight comes from the original query and **40%** from the expansion terms extracted with Dirichlet PRF.

This setup gave me the best MAP score so far: **0.2373**.

```
<parameters>
  <index>/data/IRCompetition/ROBUSTindex/</index>
  <trecFormat>true</trecFormat>
  <runID>OK_k10p8_b0p5_fbOrig0p40_fbMix0p60</runID>
  <rule>method:okapi,k1:0.8,b:0.5</rule>
  <query>
    <number>{qid}</number>
    <text>
      #weight(
        0.60  #combine({original_query_terms})
        0.40  #weight(
          {w1:.6f} "term1"
          {w2:.6f} "term2"
        )
      )
    </text>
  </query>
</parameters>
```

* At first, I removed stopwords from the query, but that turned out to be a big mistake — the MAP score dropped significantly (to around 0.12). This happened because the index includes stopwords, and removing them from the query created a mismatch that reduced performance.

So the three models I ended up choosing are:

1. **Final hybrid model** – BM25 with Dirichlet PRF-based expansion.
2. **Dirichlet with PRF** – $\mu = 1000$, fbDocs = 10, fbTerms = 30, fbOrigWeight = 0.6
3. **BM25 baseline** – $k_1 = 0.8$, $b = 0.5$ (no expansion)

Note: I used ChatGPT throughout this project