EXTRA CREDIT(PART 1)

Goal: *" Design and implement a synthetic spelling-error generator. The underlying error model is simply a character shuffler that randomly shuffles non-boundary characters in a given token (a query term). Assume that the error rate on a query level is no higher than 40%, that is, in a 10-term query, 4 terms (or fewer) are to be affected by synthetic noise. Longer tokens should have higher probability of being affected by the noise than shorter ones. Report the overall effectiveness levels when running noisy queries and compare them to those of the noise free-baseline."*

Synopsis:

For part 1 we are inducing random error based on the below pseudo code :

Step 1. Generate tokens from the query and generate a map of tokens with key as tokens and value as the length of the token.
Step 2. Sort the map in decreasing order of the length of tokens.
Step 3. Compute the L= number of tokens, then calculate X= [floor(40% * L)]
Step 4. Filter top X tokens sorted in decreasing order of their length for inducing errors.
Step 5. Use a random shuffler to shuffle the characters of the token leaving the first and last position of the token.
Step 6. Replace the error induced tokens with the original tokens in the query and return the new query generated.
Step 7. Repeat this for all queries.

Now use the error induced queries as an input to BM25 model and generate an evaluation report based on the new ranking generated from the BM25 result.

The observed overall evaluation based on the comparison between original queries and error induced queries is as follows.

Mean Average Precision and Mean Reciprocal Rank.

Runs	MAP	MRR
BM25 with original queries	0.34	0.79
BM25 with error queries	0.20	0.55

Conclusion: From above result, it is clear that spelling errors in queries could affect the overall effectiveness of the retrieved results over a great margin.

EXTRA CREDIT(PART 2)

Goal: " *Design and implement a soft matching query handler that those not correct the synthetic noise but rather minimizes its impact on effectiveness as much as possible. Your soft matching model MUST NOT (ATTEMPT TO) reverse the noise generator model ,but rather be designed independently as if the error model is unknown .You may use any (combination) of the indexing and retrieval, and query handling techniques that you've learned throughout the course or other ones. Compare overall effectiveness to those in the previous task and with noise free baseline (you may include the overall results of all three runs in one table.*

Synopsis:

For part 2 we are designing the query handler in the following way :

For Soft Matching Query Handler, we tried to implement the procedure for error correction in queries mentioned in the book "Search Engines Information Retrieval in practice by W.Bruce Croft,Donald Metzler, Trevor Strohman" with a few modifications. We are considering the chances of an error occurring and the best possible correction for the error as probabilities. We then try to calculate the probability of the correct word *w* given that we can see that the error word is *e.* This is represented as *P(w|e).* So, in order to find the corrections with maximum value of this probability, we can use *P(e|w)P(w) i.e. the product of error model probability and language model probability.* Again, we tried with a unigram language model but it provided us with less than useful result. So, we have considered language model as a mixture of probability that the word occurs in text i.e. *P(w)* and the probability that it occurs following the previous word i.e. $P(w|w_p)$, where w_p is the probability that the previous word occurs in the language model. Thus the formula is given as :

$$\lambda P(w) + (1- \lambda)P(w|w_p)$$

Where λ is a parameter that specifies the relative importance of the two probabilities and $P(w|w_p)$ is the probability of the word *w* following the previous word w_p.
Since we had a query log provided , we included the query logs as well, into the language model to emphasize on context of the query.

The steps we implemented are as follows(BM25 was used as fetching algorithm in all the scenarios):

Step-1. Created collection of possible words in English.
Step-2. Created a Term Frequency Dictionary which contains all the unigram terms in the corpus as well in the query and their respective frequencies
Step-3. Created a Bigram Index for both the corpus and the query log. This is required for obtaining the $P(w|w_p)$ component in the above equation.
Step-4. Now, for each term in each query, we are first checking if exists in the dictionary or not.

Not being in dictionary will prove it to be an error. Also, we are considering the word to be valid if it exists in the Term Frequency Dictionary. This is because words like "IBM" can exists in the corpus.

Step-5. If a word does not exist either in the collection of English words and corpus, we try to predict the best possible correction in the following way:

a. According to the examples given in the book, we assume that spelling errors occur mostly in characters apart from the first one and also we are assuming the error word will be of same length as the correct word and thus we filtered unigram terms from the English dictionary based on the above two assumptions.

b. Now for each term in the filtered list, we are calculating the distance between the error term and the current term form the list using *"Damerau–Levenshtein algorithm for edit distance"*. This algorithm was used as it incorporates transpositions also. If the distance found is less than o 0.6 times the length of the term from the dictionary, then this term is selected as possible candidate. This parameter 0.6 is selected empirically. We tried with other values such as 0.5,0.7 etc. but 0.6 gave us the best results in terms of Mean Average Precision and Mean Reciprocal Rank.

Following are the results for 3 values shown in tabular format:

Values	MAP	MRR
0.5	0.28	0.65
0.6	0.28	0.69
0.7	0.27	0.69

These values may vary between a range of 0.03-0.04

c. Now the language model for this candidate term is calculated by dividing the term frequency by the total number of words in the corpus. A constant of 0.2 is multiplied with this amount. This constant is considered to be value of λ in the equation and this value has been selected empirically .

d. Now the candidate term and the previous term to the error term in the query is searched as a pair in a Bi-Gram index formed from the corpus. If found, the Bi-Gram language model is calculated as the total number of occurrence of this pair in the corpus divided by the total number of bi-gram terms found in the corpus. A constant of 0.7 is multiplied to this value. This constant comes from the residual weightage left after calculating language model from Term Frequency Dictionary i.e. (1-0.2). The rest (0.8-0.7) will be assigned to the Bi-gram language model created from the query log. This distribution is again selected empirically and the best result was obtained in terms of Mean Average Precision and Mean Reciprocal Rank.

Outputs for 3 runs are displayed as follows:

Weightage for Corpus Bigram Index	Weightage for Query Bigram Index	MAP	MRR
0.6	0.2	0.26	0.64
0.5	0.3	0.27	0.65
0.7	0.1	0.28	0.68

These values may vary between a range of 0.03-0.04

Although by definition (followed from the text book), the residual probability i.e. (1-0.2) should be assigned to both Corpus Bi-Gram language model and Query Bi-gram model in some distribution but we tried with assigning the entire residual weight (0.8) to only Corpus Language Model and it fetched better results in some runs ,for example, we observed a MRR value of 0.73 in few runs but also it fetched 0.68 in some runs. So, we decided to go with the approach of distributing the weightage as it is more logical and more stable.

Step-6. The same procedure as mentioned above is followed for Query Bi-gram index and as stated earlier, a constant value of 0.1 is multiplied.

Step-7. If the candidate word is not found in the corpus, a very low value (1/total number of unigram terms occurring in corpus) is assigned to the candidate term.

Step-8. Now the candidate terms are sorted in non-increasing order based on their weighted language model values and the term with highest value is chosen to be the alternative to the error term.

Step-9. Now the next pair of unigram terms is selected from the query which included the corrected term in the previous iteration.

Conclusion:

This approach has enabled us to reduce the effect of errors in queries in terms of overall retrieved results. The improvement is measured in terms of Man Average Precision and Mean Reciprocal Ranks. Following is the improvement in tabular format:

Runs	MAP	MRR
BM25 with original queries	0.34	0.79
BM25 with error queries	0.20	0.56
BM25 with Corrected Queries	0.28	0.68

These values may vary between a range of 0.03-0.04
This variation is because of randomness of error generation