IR Programming 1

B07902084 資工三 鄭益昀

A Prettier version can be found here: https://hackmd.io/5hUzBULUTi6_kJ1xXuoXyg

Effect of Okapi BM 25 Normalization

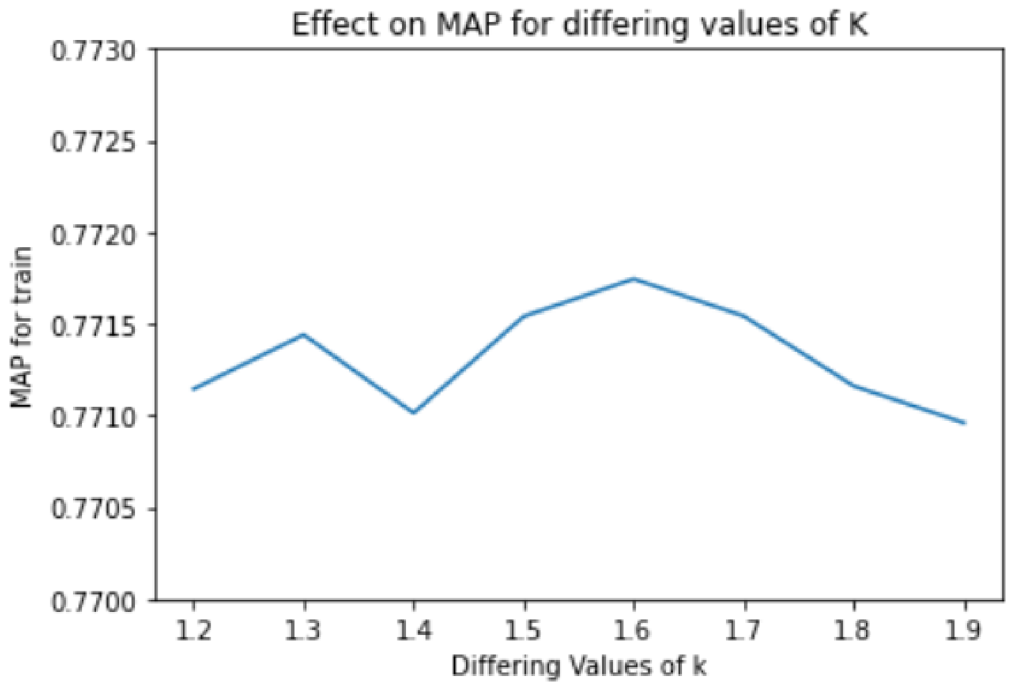
Okapi BM25 normalization is performed using the following formula:

$$\sum_{t \in Q,D} ln \frac{N - df + 0.5}{df + 0.5} \cdot \frac{(k_1 + 1)tf}{k_1((1-b) + b\frac{dl}{avdl}) + tf} \cdot \frac{(k_3 + 1)qtf}{k_3 + qtf}$$

All graphs are generated by running on train queries.

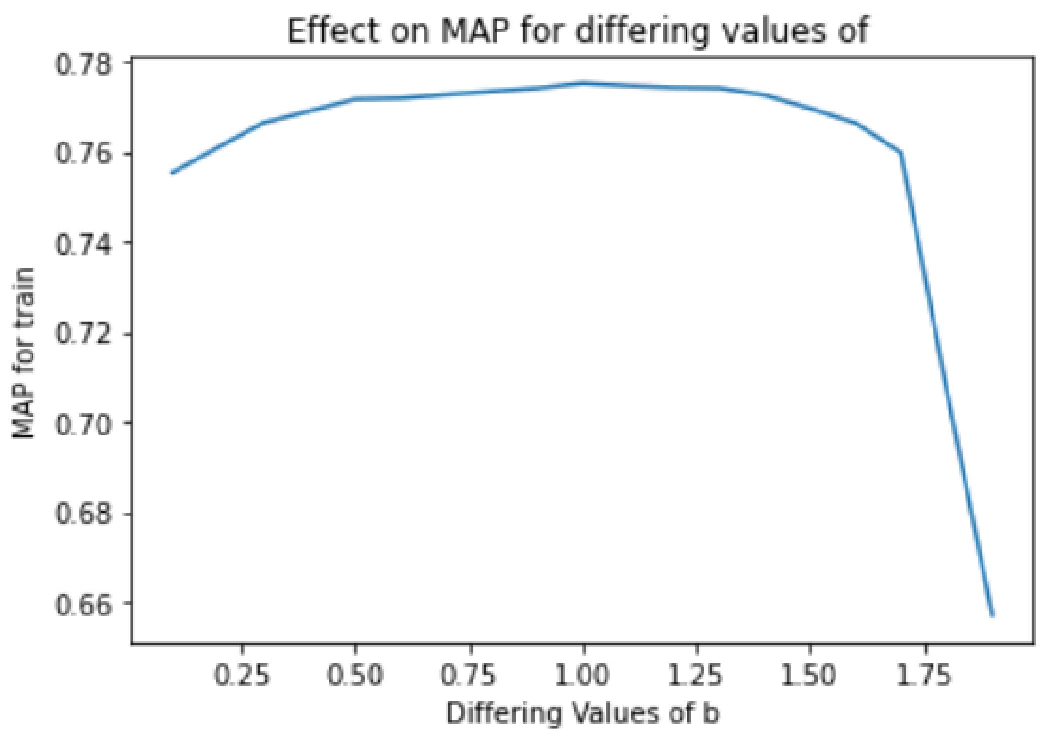
Effect of k_1

b = 0.5
k_3 = 100



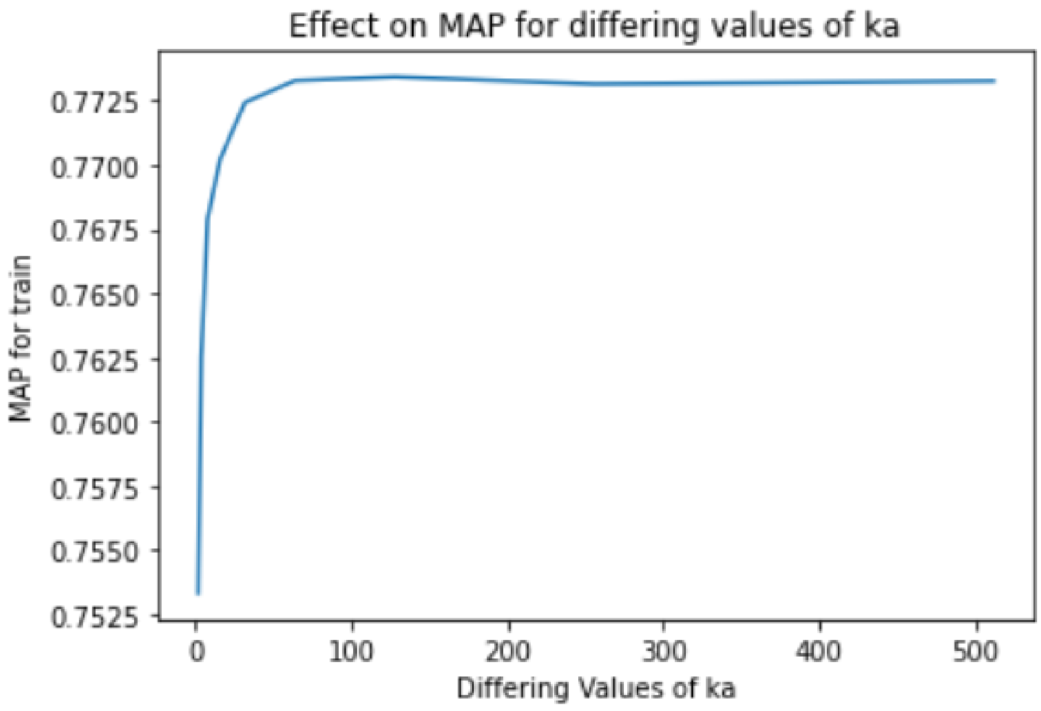
Effect of b

k_1 = 1.5
k_3 = 100



Effect of b

k_1 = 1.5
b = 0.5



Effect of Rocchio Feedback

Rocchio Feedback is performed based on the below formula:

$$q_m = \alpha q + \frac{\beta}{|D_r|} \sum_{\forall d_j \in D_r} d_j - \frac{\gamma}{|D_n|} \sum_{\forall d_j \in D_n} d_j$$

On Choosing "relevant" and "nonrelevant"

I defined relevant and nonrelevant as the top C_r and bottom C_nr documents, as generated by our intial query. A problem arise, however, that a large portion of nonrelevant document all score similar low level scores (from stub words etc), and doesn't seem to be relevant. Therefore γ was set to 0.

As for relevant document, I chose the top 50 documents as relevant. But this is a hyperparameter that can be tuned. β is set to 0.75 for my project.

Rocchio Performances

I did not produce a graph of using Rocchio Feedback as in training data, using rocchio feedback consistently decreases training data performance. This same trend can be seen on private scoreboard. However, I did try a few method to increase performance (though none worked).

Aside from hyperparameter tuning, I think the most interesting approach I used was to have multiple rounds of rocchio feedback, or use a known, well-tested (from kaggle) ranked list as the feedback vector. For example, I took the best approach public scoreboard score's ranking list and use the top 50 document as rocchio.

However, the above method never surpassed the initial upload, and seems to be ineffective (at least as of the hyperparameters I chose).

Other Relevant information

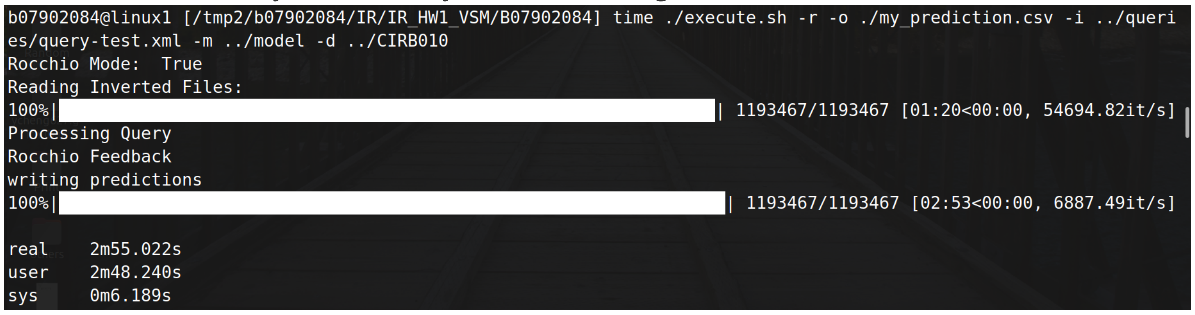
How I got top 3 in Kaggle Public (and absolutely tanked private scoreboard)

Similar to ML, ensemble seems to be a key solution to many problems. While this is not classification, I still tried to use ensemble by averaging the ranks of multiple models, and hoping this model will increase accuracy.

The result is that, by averaging these documents:
`file_names = [    "prediction-0.79304.csv",    "prediction-0.79076.csv",    "prediction-0.79020.csv",    "prediction-0.79002.csv",    "prediction-0.78790.csv"]`, I was able to get a ranked list of 0.79813, which is 0.005 higher than my highest single submission (and moved me from 9th place to 3rd place).

Time Taken (tested on linux1)

Note that this is heavily influenced by CPU load. Timing was done at 60% CPU load.



```
b07902084@linux1 [/tmp2/b07902084/IR/IR_HW1_VSM/B07902084] time ./execute.sh -r -o ./my_prediction.csv -i ../queri
es/query-test.xml -m ../model -d ../CIRB010
Rocchio Mode:  True
Reading Inverted Files:
100%|████████████████████████████████████████████████████████████████| 1193467/1193467 [01:20<00:00, 54694.82it/s]
Processing Query
Rocchio Feedback
writing predictions
100%|█████████████████████████████████████████████████████████████████| 1193467/1193467 [02:53<00:00, 6887.49it/s]

real    2m55.022s
user    2m48.240s
sys     0m6.189s
```

Discussion: What you learn in the homework

Time complexity and speedup in python

My First program was written using basically plain python list and dictionary. This was convenient as I can store grams in tuple and use it as keys. This turns out to be terrible as I tried to do rocchio feedback, which required 2 layers of for loop and took over 10 minute to complete.

This led to my improvement of using scipy and sparse matrix (as usual NP matrix was never an option given the size of data). This resulted in over 5x speedup, and allowed for a program that can run in 5 minute.

Effect of Feedback

Feedback relies heavily on the assumption that we only base the feedbacks on "relevant" information. When we have a good relevant set, our feedback can improve the ranking, but when we have a bad relevant set, this can actually hurt our experiment.

This can be seen in training data, which has a MAP of 0.4 in one of the queries. Performing Rocchio on this document decreases the MAP even more, and thus should be avoided.