CPEG 457 Assignment 2

Shane Cincotta

April 6, 2020

Part 1

1

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

Figure 1: Okapi Mathematical Formula

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

Figure 2: TF-IDF Mathematical Formula

Both of the formulas for the Oakpi and TF-IDF retrieval methods incorporate the total number of documents in the collection, the term's frequency in the document, the terms frequency in the query, the number of documents that contain the term, the document length and the average document length. Also, both formulas use $C * \frac{dl}{avdl}$ in the denominator, where C is a constant, dl is the document length and $avdl$ is the average document length.

On the other hand, the formulas also contain unshared terms. The Okapi formula has 3 distinct contstants, k_1, k_3 and b, while the TF-IDF formula has 1 distinc constant, s.

2

```
double dirMu = [0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0];
for(all){
  score += ((1 + log(1 + log(tf[i])))/((1-dirMu) + dirMu*(docLength/docLengthAvg))) * qf[i] * log((docN+1)/DF[i]);

}
```

Figure 3: TF-IDF Formula

Figure 3 shows the implemented TF-IDF formula with the value of b set by the variable dirMu.

```
double dirMu = [0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0];
for(all){
  score += log((docN - DF[i] + 0.5)/(DF[i] +0.5)) * ((2.2*tf[i]) / (1.2*((1-dirMu) + dirMu * (docLength/docLengthAvg)) + tf[i])) * ((1001*qf[i])/(1000+qf[i]));

}
```

Figure 4: Okapi Formula

Figure 4 shows the implemented TF-IDF formula with the value of s set by the variable dirMu.

3

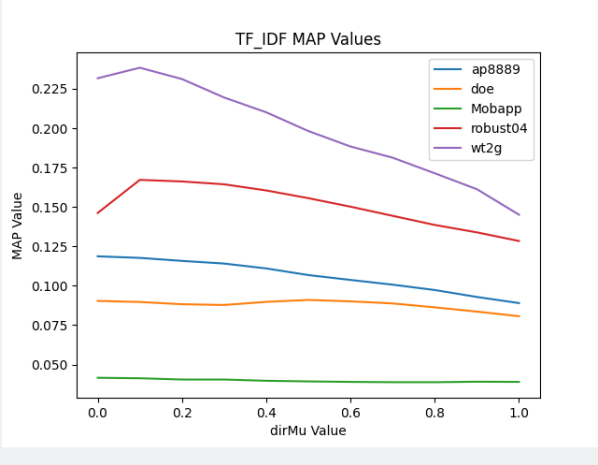

Figure 5: TF-IDF Map Values

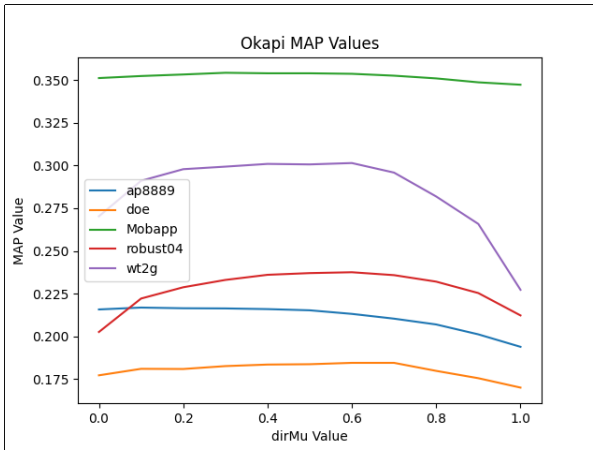

Figure 6: Okapi Map Values

Figures 5 and 6 show the performace of the above retrieval formulas with 11 dirMu values (0-1) tested against 5 collections.

According to figures 5 and 6, when optimally tuned, the Okapi formula produced the best results against every collection.

Additionally, the formulas do appear to have different sensitivities to the setting of the dirMu parameter. According to the TF-IDF results, an increase in the dirMu parameter generally correlates with a decrease of the MAP value in a semi-linear fashion.

This contrasts with the Okapi results. According to figure 6, an increase in the dirMu parameter generally correlates with a increase of the MAP value. This trend continues until dirMu reaches about 0.6, at which point an increase of the dirMu parameter generally results in a decrease of the MAP value.

The results also vary widely depending on the collection used. For example, the Okapi formula has a roughly steady MAP value of 0.35 when tested with the Mobapp collection. But the formula has a roughly steady MAP value of 0.18 when tested with the doe collection, around a 50 percent decrease. Another interesting observation is that there does not seem to be a constant rank regarding difficulty of the collections, that is, just because Okapi did poorly with one collection relative to the other collections, does not mean that TF-IDF will also perform poorly. For example, the Okapi method performed the best with the Mobapp collection, yet the TF-IDF method performed the worst with that same collection.

Part 2

BM25Plus

For my first retrieval function, I implemented the BM25Plus method. Figure 7 shows the mathematical formula, figure 8 shows the formula translated to code and figures 9 and 10 show graphs of the results.

$$\sum_{t \in Q \cap D} c(t,Q) \left[\frac{1 + \log\left(1 + \log(c(t,D))\right)}{1 - s + s\frac{|D|}{avdl}} + \delta \right] \log \frac{N+1}{df(t)}$$

Figure 7:

```
double dirMu = [0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0];
for(all){
  score += qf[i] * (((1 + log10(1 + log10(tf[i]))) / (1-dirMu + (dirMu*(docLength/docLengthAvg))))+0.5) * log10((termN + 1)/ DF[i]
}
```

Figure 8:

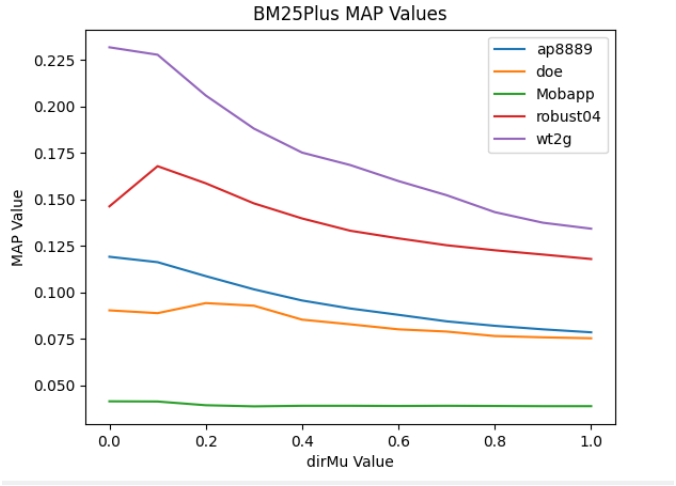

Figure 9:

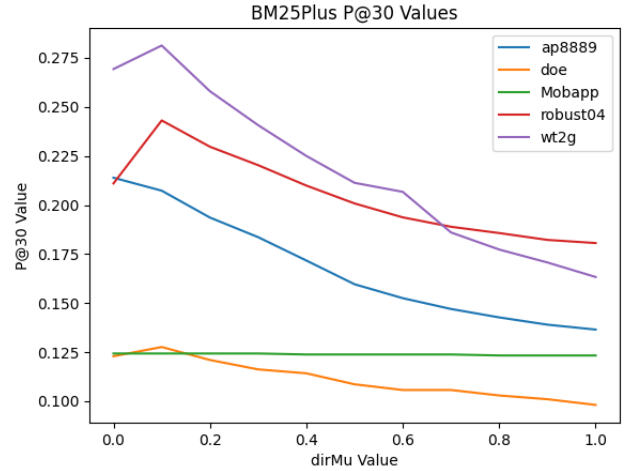

Figure 10:

PL2

For my second retrieval function, I implemented the PL2 method. Figures 11 and 12 show the mathematical formula, figure 13 shows the formula translated to code and figures 14 and 15 show graphs of the results.

$$\frac{(k_1+1)\cdot tfn_t^D}{k_1+tfn_t^D}\cdot\log\frac{N+1}{df(t)}$$

Figure 11:

$$tfn_t^D = c(t,D)\cdot\log_2\left(1+c\cdot\frac{avdl}{|D|}\right)$$

Figure 12:

```
double dirMu = [0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0];
for(all){
  score += (((1000+1) * (tf[i] * log2(1 + dirMu * (docLengthAvg/docLength))))/(1000 + ((tf[i] * log2(1 + dirMu * (docLengthAvg/doc

}
```

Figure 13:

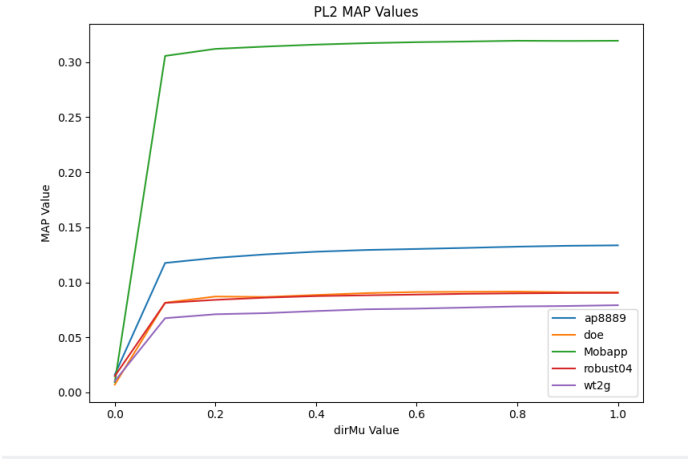

Figure 14:

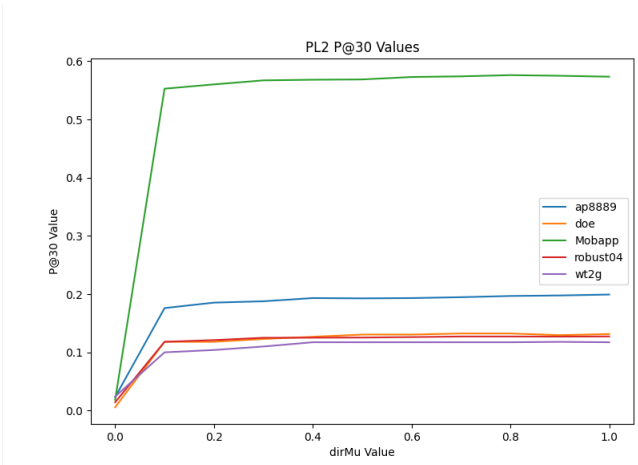

Figure 15:

F1-Log

For my last retrieval function, I implemented the F1-Log method. Figure 16 shows the mathematical formula, figure 17 shows the formula translated to code and figures 18 and 19 show graphs of the results.

$$\sum_{t \in Q \cap D} C_t^Q \cdot TF(C_t^D) \cdot LN(|D|) \cdot LW(t)$$

Figure 16:

```
double dirMu = [0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0];
for(all){
  score += qf[i] * (1+log((1+log(tf[i])))) * ((docLengthAvg + dirMu)/(docLengthAvg + (docLength*dirMu))) * (log((1 + docN)/(DF[i])

}
```

Figure 17:

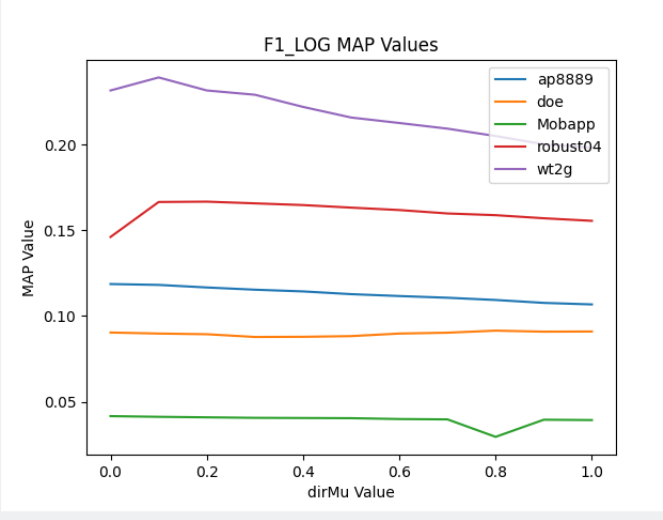

Figure 18:

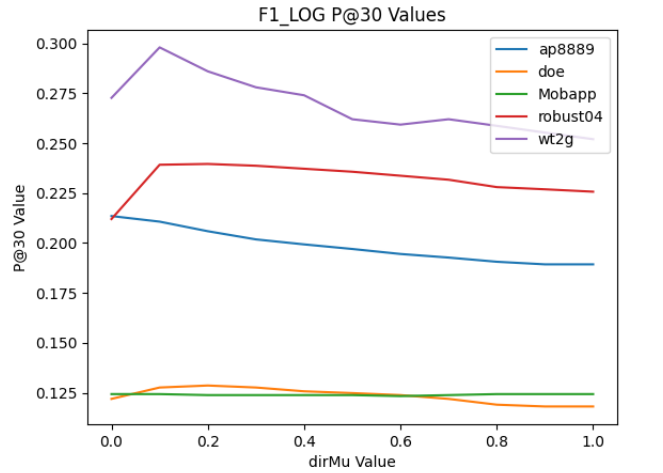

Figure 19: