**Project Two – Retrieval Models**

1. **Expectations from each scoring function:**
2. **Scoring Function 1: Okapi TF**

Okapi TF is based on Vector space model or term vector model which is an algebraic model for representing text documents (and any objects, in general) as vectors of identifiers, such as, for example, index terms. In Okapi TF, it ranks the documents based only on their term frequency. So here both the documents as well as the queries are represented as a vector.

So if we have the two documents as below:

Document1 = “John is quicker than Marry”  
Document2 = “Marry is quicker than John”

Okapi TF will have the same vector for both the documents. In this view of a document, known in the literature as the bag of words model, the exact ordering of the terms in a document is ignored but the number of occurrences of each term is material.

It seems here that Okapi TF model would do a better work of a recall rather than concentrating on the precision. This is expected to perform less compared to TF-IDF and Okapi BM25 as both of them consider the inverse document frequency and this model does not.

1. **Scoring Function 2: TF-IDF**

This model has inverse document frequency included in the formula. So with no doubt this model will perform better than the Okapi TF.

Consider any random term “A” that occurs in all our documents where N = 3468. So its IDF component will become log (N/nk) -> nk is the total number of documents in which the term “A” occurs. In our case the term is in all documents. So log(N/nk) = log(1) = 0. This we multiply with the Okapi TF component obtained in (1) above.

Hence, according to this the rare terms in the document would assign higher weights to the documents.

1. **Scoring Function 3: Okapi BM25**

Okapi BM25 is expected to perform better than all of the given scoring functions. This formula has three components. The first is an IDF component that reflects the discriminative power of each word. If we examine cases where df(i) = 0 or N we come to know. The second component is a TF component where we can stress two thing

1. This function is increasing in term frequency, but reaches an asymptotic limit from below.

This means whether a term appears a 100 times or a 1000 times the function will weight it almost the same.

1. There is a correction for document weight. If a document is short, the term frequency for all its words is increased; if a document is long the term frequency for all its words is decreased. The count of each word is measured w.r.t. the document of average length in the collection.

There is a third component which is a QTF(Query term frequency) component which says if a word in the query appears more times than another it should be weighted higher.

Based on the above points the BM25 should perform better than any of the given scoring functions.

1. **Scoring Function 4: Language model with Laplace Smoothing**

Laplace Smoothing will perform bad as compared to Okapi TF, TF-IDF and BM25. This is so because, in this language model the query terms are treated independent from each other i.e conditional independence. Also the queries and the documents are treated as the objects of the same type. We use smoothing for avoiding the estimation problem and overcoming the data sparsity, which means we typically do not have large amounts of text to use for language model probability estimates. The general approach to smoothing is to lower the probability estimates for the words that are not seen in the test. The estimates for unseen words are usually based on the frequency of occurrence of words in the whole document.

So due to the above inconsistencies of not considering inverse document frequency explicitly and producing inconsistency of correctness by assuming documents and queries are same objects, Laplace Smoothing will perform bad as compared to Okapi TF, TF-IDF and BM25.

1. **Scoring Function 5: Language model with Jelinek-Mercer Smoothing**

JM would definitely perform better than Laplace as it does not treat all unseen words equally and takes into account the background probabilities in the entire corpus. Also for making the document and the query objects independent of each other we introduce lamda(λ).

Where Background Probability is equivalent to collection language model probability which is defined as P(qi|C) is the probability for query word I for document collection C, then the estimate we use for an unseen word in document is λ P(qi|C), where λ is the coefficient controlling the probability to unseen words.

I also think that the performance of JM would be close to BM25 but not greater than that.

1. **Data Tables, Description and Analysis of the Scoring functions:**
2. **Scoring Function 1: Okapi TF**

|  |  |  |
| --- | --- | --- |
| Query ID | Query | GAP score |
| 202 | uss carl vinson | 1 |
| 214 | capital gains tax rate | 0.487091257 |
| 216 | nicolas cage movies | 0.354021378 |
| 221 | electoral college 2008 results | 0.359660731 |
| 227 | i will survive lyrics | 0.098354753 |
| 230 | world's biggest dog | 0.243779593 |
| 234 | dark chocolate health benefits | 0.51508467 |
| 243 | afghanistan flag | 0.378926225 |
| 246 | civil war battles in South Carolina | 0.221778538 |
| 250 | ford edge problems | 0.054788271 |
|  | AVG | 0.371348542 |

Best Score: We got a perfect GAP score of 1 for “uss carl vinson”. This implies that the relevance for this query is very high. All the terms in this query are very dependent on each other. “carl vinson” is a name of a person which is very likely to come together in any document. So the angle of the vectors formed for the query and document is very acute.

Worst Score: “ford edge problems” all the words can appear across the corpus independently. There is no such dependence on the words. This is the reason why the GAP score is very bad for this query.

1. **Scoring Function 2: TF-IDF**

|  |  |  |
| --- | --- | --- |
| Query ID | Query | GAP score |
| 202 | uss carl vinson | 1 |
| 214 | capital gains tax rate | 0.538791789 |
| 216 | nicolas cage movies | 0.368752834 |
| 221 | electoral college 2008 results | 0.357403124 |
| 227 | i will survive lyrics | 0.112806698 |
| 230 | world's biggest dog | 0.22965442 |
| 234 | dark chocolate health benefits | 0.522931676 |
| 243 | afghanistan flag | 0.391578062 |
| 246 | civil war battles in South Carolina | 0.215587948 |
| 250 | ford edge problems | 0.053989386 |
|  | AVG | 0.379149594 |

* In TF-IDF the score is higher for the terms that rarely appear across the corpus as discussed before. So higher term weights are given to those terms. Here as compare to TF we also calculate the weights based on the term frequency in the document.
* But we multiply this with the inverse document frequency. It perform better due to that as it considers the occurrence of the term w.r.t to the entire corpus. Hence we can see that TF-IDF gives higher precision results as compared to the Okapi TF formula. Hence my prediction that TF-IDF will perform better than Okapi TF is true.

1. **Scoring Function 3: Okapi BM25**

|  |  |  |
| --- | --- | --- |
| Query ID | Query | GAP score |
| 202 | uss carl vinson | 0.007042254 |
| 214 | capital gains tax rate | 0.567507654 |
| 216 | nicolas cage movies | 0.481418084 |
| 221 | electoral college 2008 results | 0.424440054 |
| 227 | i will survive lyrics | 0.241046747 |
| 230 | world's biggest dog | 0.382918312 |
| 234 | dark chocolate health benefits | 0.685653286 |
| 243 | afghanistan flag | 0.42536309 |
| 246 | civil war battles in South Carolina | 0.182825724 |
| 250 | ford edge problems | 0.510957629 |
|  | AVG | 0.390917283 |

* Both the Okapi TF and TF-IDF failed (worst scores) on the query “ford edge problems”, mainly because both the models did not consider the wide spread of the terms in the entire corpus. The vector space model fails in the case of the query if it contains independent terms.
* Whereas BM25 is modeled from a probabilistic binary independence model. BM25 assumes that terms in the query are independent of each other. As we can evidently see that there is a drastic increase in the score of the “ford edge problems” query.
* On the other hand where Okapi TF and TF-IDF scored best(“uss carl vinson”), BM25 failed drastically. And also BM25 is said to give inconsistent results for large documents.
* The middle component in the formula **(1+k1)\*tf(d,i) / K+tf(d,i),** tends to value zero for very large documents having many terms. So this middle term does not distinguish between very large document containing the term and other documents not containing it at all.
* So as per my previous analysis BM25 was better than Okapi TF and TF-IDF.

1. **Scoring Function 4: Language model with Laplace Smoothing**

|  |  |  |
| --- | --- | --- |
| Query ID | Query | GAP score |
| 202 | uss carl vinson | 0.005208333 |
| 214 | capital gains tax rate | 0.504038522 |
| 216 | nicolas cage movies | 0.470812002 |
| 221 | electoral college 2008 results | 0.419645428 |
| 227 | i will survive lyrics | 0.109762634 |
| 230 | world's biggest dog | 0.186795839 |
| 234 | dark chocolate health benefits | 0.549190703 |
| 243 | afghanistan flag | 0.132673938 |
| 246 | civil war battles in South Carolina | 0.098901558 |
| 250 | ford edge problems | 0.118314002 |
|  | AVG | 0.259534296 |

* Seeing on all the scores as well as the average score given by Laplace and comparing them with the previous scoring functions, Laplace has given the worst results by far as predicted. This type of smoothing is also called add one smoothing. What it does it adds 1 to every term which has no frequency in the document. All the zero probabilities do not occur in this case. And we also normalize the denominator by |V| (Vocabulary size).
* Here in Laplace smoothing the query terms which do not occur in the document have the same probability contribution regardless of their frequency occurring in the corpus. This leads to an understanding that this model does not take into importance of the query term and assigns a smoothing factor of 1 to the tf. This leads to the poor precision of the query with comparison to other models.

1. **Scoring Function 5: Language model with Jelinek-Mercer Smoothing**

|  |  |  |
| --- | --- | --- |
| Query ID | Query | GAP score |
| 202 | uss carl vinson | 1 |
| 214 | capital gains tax rate | 0.55684368 |
| 216 | nicolas cage movies | 0.4145165 |
| 221 | electoral college 2008 results | 0.357245253 |
| 227 | i will survive lyrics | 0.180517579 |
| 230 | world's biggest dog | 0.307715856 |
| 234 | dark chocolate health benefits | 0.579889062 |
| 243 | afghanistan flag | 0.428314521 |
| 246 | civil war battles in South Carolina | 0.230419166 |
| 250 | ford edge problems | 0.20294262 |
|  | AVG | 0.425840424 |

Comparing the above results with Laplace we see that we have much better readings in JM model as predicted correctly. But as expected by me that the GAP score of JM will be less than BM25 proves me wrong. Instead I have got better results as compared to BM25 even JM does not contain a inverse document frequency term. As given in the estimations the JM model does not treat all unseen words equally and takes into account the background probabilities in the entire corpus. The parameter lambda λ is responsible for this as explained below.

* Small values of λ produce less smoothing, and consequently tends to act more like a Boolean AND since the absence of any query word will penalize the score substantially.
* In addition, the relative weighting of words, as measures by the maximum likelihood estimates, will be important in determining the score.
* As λ approaches 1m the relative weighting will be less important, and the query acts more like a Boolean OR. In TREC evaluations, it has shown that values of λ around 0.1 work well for short queries and values around 0.7 work well for longer queries. In our program we have taken it to be 0.2.
* Short queries tend to contain only significant words, and a low λ value will favor documents that contain all the query words.
* With much longer queries missing a word is much less important, and a high λ places more emphasis on documents that contain a number of the high probability words.

1. **Was there a method which was significantly better than the others?**

As per the average GAP scores we have ::

1. Okapi TF -> 0.371348542
2. TF-IDF -> 0.379149594
3. BM25 -> 0.390917283
4. Laplace Smoothing -> 0.259534296
5. Jelinek-Mercer(JM) Smoothing -> 0.425840424

As seen from the above JM performs the best out of the lot. Which I did not expect. BM25 should have performed better than the JM model. After researching a bit I also found that JM have performed better than BM25 in TREC8 collection of documents but loses out on TREC9 and TREC10 collection of documents.

1. **Was there a query which was significantly harder than the others?**

* The query “i will survive lyrics” performed the worst for all the scoring functions.
* This was due to the terms “i” and “will” we one of the stop words and they were filtered out before giving them to any scoring function.
* This converts the query only to the form “survive lyrics” which on its own is not a song. So over here the terms “i will” were necessary to score the documents properly.