Information Retrieval and Extraction

# ANALYSIS OF DOCUMENT RETRIEVAL MODEL FOR NASA CORPUS

Hardik Sharma

# Introduction

A document retrieval model works by first parsing all the documents and then constructing an algorithm for scoring each document based on a given query as input.

Before actually going forward and implementing the retrieval model, we analyse the data present using some used standard visualizations.

We also study some heuristic optimizations to find possible optimizations and the inherent trade-off.

# Task 1, 2, and 3

**Goal :** ***Cleaning the file of punctuation, and complete numbers***

So, setting out this task, we create a `Cleaning` class that implements a `cleanFile` method. The `cleanFile` takes two keyword parameters, `(performStemming, performStopWordRemoval)`, both are `True` by default. For this task, we just clean the first fifteen documents by setting both the arguments as `False`.

Note that tokenization is implemented from scratch, rather than using any external library. This is performed by comparing a regex that filters on basis of any *utf-8* character. After splitting on this bases, we get a list of words present in the given documents.

The implementation of this can be found in `./preprocessing/cleaning.py`.

# Task 4

**Goal :** ***Building Boolean and Vector Models for top p Stems and providing queries to them to find differences between them.***

### Vector Model

The vector model takes performs the scoring as follows.

1. For each term in query, it calculates the term frequency of the term in query. This is the weight of the term in the query. $w_{t,q}$ .
2. For each term based on the `scoringCriteria`, we compute the termDocumentWeight matrix termed as `termScores` property for the `cluster` class.
3. After doing the above, we extrapolate the query vector to the size of corpus and then do the same for the document, to account for all terms in the union of `queryVocabulary` and `clusterVocabulary`.
4. Finally take the dot product of the two and normalize it, to get the final cosine score for each document.

### Boolean Model

The boolean model computes the score exactly as in the case of vector model, only difference is that the weight of the term in the query is either 0 or 1. Similarly, if the word is present in the document the weight is 1, otherwise 0.

### Probabilistic Model

The implementation consists of going through all the terms in the corpus initially and then initializing the relevant and irrelevance Probabilities as follows.

$$relevanceProbability[term] = 0.5 irrelevanceProbability[term] = documentFrequency[term]/N$$

where $term \in$ `self.freqMap`,$N$ is the number of documents in corpus Now, for every document in the corpus, go through each term in the query and compute the following sum:

$$score_{q,d} = \sum_{t \in Q \cap D} ln(\frac{relevanceProbability[t]}{1 - relevanveProbability[t]}) + ln(\frac{1 - irrelevanceProbability[t]}{irrelevanceProbability[t]})$$

where $Q, D$ are the set of terms in query and document respectively and $relevanceProbability[term]$ and $irrelevanceProbability[term]$ are the relevance and irrelevance probabilities, $P(t, D|R = 1)$ and $P(t, D|R = 0)$ Now we update the probabilities as follows:

$$relevanceProbability[term] = \frac{|V_t| + 0.5}{|V| + 1}$$

$$irrelevanceProbability[term] = \frac{n_t - |V_t| + 0.5}{N - |V| + 1}$$

We only do the above for the terms present in the `topR` documents extracted after each iteration. The convergence criteria is when the scores of these `topR` documents do not change. There is also an `numEpochs` limit set to control the model in case, for some reason the values do not converge. On observation, the values converged after only a few iterations, i.e. 2-3 iterations. The `numEpochs` limit is set to 5 by default.

### Latent Semantic Model

The latent semantic model is fairly straightforward to implement. Firstly, the tfIdf scores are calculated for the entire corpus. Using this we get a term-document Matrix `M`.

Then, we add the query's vector representation to this model as another document. Essentially treating the query as a document.

Now, using SVD algorithm in `numpy.linalg.svd`, we get a tuple corresponding to the three matrices obtained as a result of the SVD.

Since the eigenvalues present in the diagonal matrix are sorted in descending order by default, we take the first $l$ columns from the first matrix, $l$ first eigenvalues from the second matrix and $l$ first rows from the third matrix.

Recomputing the product, we get a reduced subspace in which the entire corpus is being represented. We can call this new Matrix as `M'`.

Now, to compute pairwise the score of similarity, we can either take the cosine similarity from the representation of the query, or we can just take the matrix multiplication between the transpose of the `M'` matrix and the `M'` matrix, creating a new `n $\cdot$ n` matrix, where `Score`[i][j] gives the similarity between document i and j, since the query is also a document, we can extract the score using the last row vector or the column vector.

This gives us the final scoring.

## Comparing Models

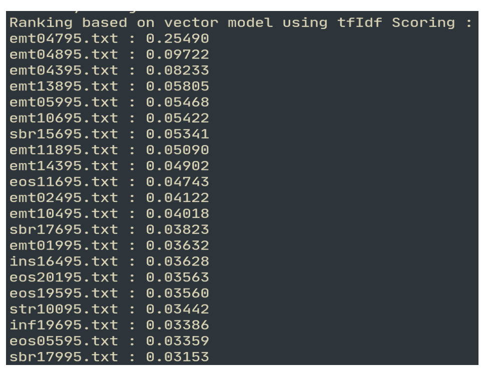
To check for differences we can run curated queries and check the differences between the models.

### Query 1

**Statement :** ***NASA Project that analysis boundary elements using fracture analysis and dislocations, using the FADD method.***

This query is constructed using the *.key* file of **emt04795.txt**. Below is the result of document scoring by these models.

All the models confidently find the target document. But the certainty of that scoring is reflected in the scores they have given to each of the terms.



```
Ranking based on vector model using tfIdf Scoring :
emt04795.txt : 0.25490
emt04895.txt : 0.09722
emt04395.txt : 0.08233
emt13895.txt : 0.05805
emt05995.txt : 0.05468
emt10695.txt : 0.05422
sbr15695.txt : 0.05341
emt11895.txt : 0.05090
emt14395.txt : 0.04902
eos11695.txt : 0.04743
emt02495.txt : 0.04122
emt10495.txt : 0.04018
sbr17695.txt : 0.03823
emt01995.txt : 0.03632
ins16495.txt : 0.03628
eos20195.txt : 0.03563
eos19595.txt : 0.03560
str10095.txt : 0.03442
inf19695.txt : 0.03386
eos05595.txt : 0.03359
sbr17995.txt : 0.03153
```

Figure 1: Using Vector Model with tfIdf Scoring for Query 1

The tfIdf model confidently separates the relevant file with the rest of the corpus, giving a difference of ( 0.16) between the first two files.

The linear tfIdf model is not so far behind either, even though the difference between the first two scores ( 0.13) is less than that in the case of tfIdf( 0.16), it still gives higher absolute score to the relevant document.

The boolean model gives the file with relatively less confidence than linear tfIdf with an absolute score of (0.25) for the relevant file and the difference between the first two being (0.09).

The scores given by the Probabilistic models and those given by LSM models are shown below :

Both the models correctly retrieve the relevant document within their top 2. The LSM model performing better.

```
Ranking based on vector model using lineartfIdf Scoring :
emt04795.txt : 0.28623
sbr15695.txt : 0.15611
emt04395.txt : 0.15543
emt01995.txt : 0.14103
emt04895.txt : 0.12247
emt11895.txt : 0.09452
emt05995.txt : 0.08349
sbr17995.txt : 0.08046
emt02495.txt : 0.07497
inf21595.txt : 0.06906
emt13895.txt : 0.06792
emt14395.txt : 0.06513
emt10695.txt : 0.06157
eos11695.txt : 0.05107
emt10495.txt : 0.04922
ins16495.txt : 0.04348
sbr17695.txt : 0.04325
emt04995.txt : 0.04175
emt10395.txt : 0.04135
eos19895.txt : 0.03547
emt13495.txt : 0.03509
```

Figure 2: Using Vector Model with linear tfIdf Scoring for Query 1

```
Ranking based on boolean model using boolean Scoring :
emt04795.txt : 0.26149
emt04895.txt : 0.17968
emt04395.txt : 0.17392
emt14395.txt : 0.16584
emt13895.txt : 0.16046
eos19595.txt : 0.15768
emt10695.txt : 0.15756
eos16095.txt : 0.15724
str10095.txt : 0.15594
emt02695.txt : 0.15397
eos11695.txt : 0.15385
ins20595.txt : 0.15207
emt05995.txt : 0.15097
eos06895.txt : 0.14709
eos16995.txt : 0.14511
emt11895.txt : 0.14391
eos20195.txt : 0.14048
ins20795.txt : 0.14048
inf07395.txt : 0.14011
mat14695.txt : 0.14011
eos06695.txt : 0.13916
```

Figure 3: Using Boolean Model for Query 1

```
Ranking based on vector model using BIM :
emt04795.txt : 19.80242
emt04895.txt : 13.82010
emt04395.txt : 13.36593
emt10695.txt : 12.95845
emt13895.txt : 12.95845
emt05995.txt : 11.82805
eos19595.txt : 11.82805
sbr17695.txt : 11.82805
eos05595.txt : 11.17354
str02595.txt : 11.17354
emt14395.txt : 11.01599
eos11695.txt : 11.01599
mat09995.txt : 10.77883
mat14695.txt : 10.77883
mat17295.txt : 10.77883
eos16995.txt : 10.68488
Ranking based on vector model using Latent Semantic Model:
emt04895.txt : 5.96995
emt13895.txt : 5.36432
emt04395.txt : 5.25349
emt01995.txt : 5.02206
emt10695.txt : 4.95494
sbr15695.txt : 4.61982
emt05995.txt : 4.55190
sbr17695.txt : 4.25736
emt10495.txt : 3.67186
emt04795.txt : 3.45436
eos11695.txt : 3.02637
inf19695.txt : 2.81228
ins20495.txt : 2.76232
sbr06295.txt : 2.72767
emt11895.txt : 2.70487
emt02495.txt : 2.62061
```

Figure 4: Probabilistic and LSM Model Scoring for Query 1

### Query 2

**Statement :** ***Does NASA have any ongoing project on non-destructive evaluation that employs electromagnetic probe technology?***

This query is constructed using the *.key* file of **ins04695.txt**. Below is the result of document scoring by these models.

```
Ranking based on vector model using tfIdf Scoring :
emt13895.txt : 0.09263
ins04695.txt : 0.09085
ins05395.txt : 0.06860
eos06895.txt : 0.05857
emt10695.txt : 0.05831
inf21595.txt : 0.05308
str00795.txt : 0.05079
ins13995.txt : 0.04920
ins05895.txt : 0.04905
eos19995.txt : 0.04892
sbr06195.txt : 0.04643
mat02095.txt : 0.04251
inf07395.txt : 0.04169
ins09595.txt : 0.04144
eos07795.txt : 0.03950
emt10495.txt : 0.03945
emt05995.txt : 0.03713
eos05595.txt : 0.03708
ins05295.txt : 0.03679
str02595.txt : 0.03515
eos00395.txt : 0.03514
```

Figure 5: Using Vector Model with tfIdf Scoring for Query 2

Here, with the exception of boolean model, the other models successfully retrieve the most relevant file with regards to the query : **ins04695.txt**.

```
Ranking based on vector model using lineartfIdf Scoring :
ins04695.txt : 0.19852
emt13895.txt : 0.14700
eos06895.txt : 0.11022
inf21595.txt : 0.10857
ins05395.txt : 0.09918
ins05895.txt : 0.08123
eos19995.txt : 0.07852
ins13995.txt : 0.05925
ins05295.txt : 0.04795
str00795.txt : 0.04319
mip14795.txt : 0.03679
sbr12295.txt : 0.03590
mat02095.txt : 0.03103
emt10695.txt : 0.02948
emt05995.txt : 0.02761
ins01795.txt : 0.02732
inf07395.txt : 0.02701
inf12795.txt : 0.02657
sbr06195.txt : 0.02651
emt04595.txt : 0.02646
emt04495.txt : 0.02477
```

Figure 6: Using Vector Model with linear tfIdf Scoring for Query 2

The tfIdf model is confused with another file, namely **emt13895.txt**, since that file is also somewhat related to electromagnetic phenomena. The important difference is that in this file, the term *electromagnetic* occurs 4 times as much as in the target file, thereby throwing off the model a bit.

The model incorrectly and relatively confidently predicts that this file is the relevant document, since its term frequencies are a bit skewed. It also fails to distinguish between the boundary which should separate the somewhat relevant documents to documents that are visibly unrelated.

```
Ranking based on boolean model using boolean Scoring :
ins04695.txt : 0.12457
emt13895.txt : 0.12123
ins05395.txt : 0.12071
emt10695.txt : 0.11815
inf07395.txt : 0.11035
eos16095.txt : 0.10721
eos05595.txt : 0.10710
eos06895.txt : 0.10660
eos00395.txt : 0.10349
str00795.txt : 0.10050
emt10495.txt : 0.09911
sbr06195.txt : 0.09724
inf21595.txt : 0.09428
ins01795.txt : 0.09366
ins07595.txt : 0.09285
mip14795.txt : 0.09206
eos07195.txt : 0.09167
ins09595.txt : 0.09091
ins13995.txt : 0.09072
mip00595.txt : 0.08980
emt04495.txt : 0.08874
```

Figure 7: Using Boolean Model for Query 2

The first and foremost observation is that these models also seem to think the **emt13895.txt** file to be closely relevant, so the first model didn't perform too bad.

The linear tfIdf model seems to be fairly confident that, even though this file is related to the query, it is not the exact file, since it has significant difference in scores for the two ( 0.03). More than the difference for the second and third position in the tfIdf model.

The boolean model on the other hand, fails to even retrieve the relevant document in the top 15 documents, it is also interesting to note that the boolean model has a fairly even score distribution, it fails to clearly form a boundary for relevant and non-relevant queries.

```
Ranking based on vector model using BIM :
emt13895.txt : 12.37598
eos00395.txt : 11.99124
mat02095.txt : 11.28689
mip01195.txt : 10.80473
emt05995.txt : 10.62938
inf21595.txt : 10.52671
inf07395.txt : 10.44374
ins04695.txt : 10.43963
emt10695.txt : 10.41864
sbr06195.txt : 10.37848
emt10495.txt : 10.27662
str00795.txt : 10.22157
ins13995.txt : 10.21136
eos19895.txt : 10.09614
eos05595.txt : 9.71572
ins14595.txt : 9.59515
Ranking based on vector model using Latent Semantic Model:
ins04695.txt : 5.14258
emt13895.txt : 4.41890
emt10695.txt : 3.77483
inf21595.txt : 3.74853
eos00395.txt : 3.61040
mat02095.txt : 3.60923
str00795.txt : 3.33636
ins13995.txt : 3.31655
sbr06195.txt : 3.28293
ins14595.txt : 3.07775
emt10495.txt : 3.01237
inf12795.txt : 2.52056
ins05295.txt : 2.50972
sbr12295.txt : 2.31308
emt01995.txt : 2.29062
eos19995.txt : 2.21247
```

Figure 8: Using Probabilistic and LSM Scoring for Query 2

The LSM Model correctly finds the most relevant document, also finding the second most relevant document. The confidence seems relatively stronger than that of `lineartfIdf` model.

### Query 3

**Statement :** ***What nasa projects are focused on the development of networks, internet and world wide web?***

This query is constructed using the *.key* file of **inf11595.txt**. Below is the result of document scoring by these models.

```
Ranking based on vector model using tfIdf Scoring :
inf12795.txt : 0.12016
inf21595.txt : 0.10395
inf11595.txt : 0.10384
emt14395.txt : 0.10364
eos16095.txt : 0.09732
emt21795.txt : 0.08619
inf02895.txt : 0.07647
inf12995.txt : 0.06038
inf17395.txt : 0.04691
inf07395.txt : 0.04117
inf19695.txt : 0.03871
emt17495.txt : 0.03649
eos11695.txt : 0.03536
emt13295.txt : 0.03485
str10095.txt : 0.03237
sbr21395.txt : 0.03193
inf17195.txt : 0.03066
eos20195.txt : 0.02958
ins14595.txt : 0.02858
sbr17695.txt : 0.02565
ins13995.txt : 0.02509
```

Figure 9: Using Vector Model with tfIdf Scoring for Query 3

At first glance it seems that apart from the linear tf-idf scoring, the other models get confused with a certain other file (**inf12795.txt**). But upon furthur inspection of the keywords present in (**inf12795.key**), we can see that this file is also relevant to the given query!

```
Ranking based on vector model using lineartfIdf Scoring :
inf11595.txt : 0.23539
inf12795.txt : 0.18438
inf21595.txt : 0.12144
eos16095.txt : 0.09685
emt17495.txt : 0.09223
emt21795.txt : 0.07783
sbr21395.txt : 0.07569
emt14395.txt : 0.06899
inf02895.txt : 0.05714
ins13995.txt : 0.03036
inf19695.txt : 0.02743
inf12995.txt : 0.02632
eos11695.txt : 0.02624
inf17395.txt : 0.02333
sbr17695.txt : 0.02312
inf07395.txt : 0.02223
eos20195.txt : 0.02021
str10095.txt : 0.02016
mat02095.txt : 0.01985
emt13295.txt : 0.01980
sbr18095.txt : 0.01883
```

Figure 10: Using Vector Model with linear tfIdf Scoring for Query 3

For the tfIdf scoring, we see that it successfully manages to capture both the documents in the top 3 retrieved documents. Even though the third document is not very relevant, it is present as a noise due to high number of common stop words in the query and that document.

```
Ranking based on boolean model using boolean Scoring :
eos16095.txt : 0.24398
emt14395.txt : 0.23158
inf12795.txt : 0.21381
inf21595.txt : 0.20698
inf02895.txt : 0.20212
inf12995.txt : 0.19983
inf17395.txt : 0.18732
emt21795.txt : 0.18607
inf11595.txt : 0.18163
mat00695.txt : 0.16865
eos03795.txt : 0.16781
inf07395.txt : 0.16771
str10095.txt : 0.16591
mip09195.txt : 0.16485
ins07595.txt : 0.16279
ins21195.txt : 0.16279
mip00595.txt : 0.16179
eos11695.txt : 0.16113
ins14595.txt : 0.16113
eos06895.txt : 0.15975
inf17195.txt : 0.15885
```

Figure 11: Using Boolean Model for Query 3

The boolean model on the other hand fails to retrive the documents in any structured manner, the target document is ranked 9th in the list, whereas the document which could be confused with, is ranked 3rd, meaning there are 7 noisy and irrelevant documents in the top 9 documents, this is not at all an ideal retrival.

The models overall suffer from the noise of stop words, even through all this, the linear tfIdf model still manages to give the target document as the most valid document all of the time. This is amazing, as it essentially searches for relevant terms even in a sea of stop words that are present to throw it off.

The probabilistic model performs fairly decent, getting the correct files for the first two queries and missing slightly for the last one. The LSM scoring on the other hand, predicts all the queries along with the linearTfIdf model.

```
Ranking based on vector model using BIM :
inf21595.txt : 15.40467
eos16095.txt : 15.37364
emt13295.txt : 14.03252
emt14395.txt : 14.03252
inf19695.txt : 14.03252
inf12795.txt : 13.91982
inf11595.txt : 13.64494
eos11695.txt : 12.82539
inf02895.txt : 12.82539
inf17395.txt : 12.82539
inf07395.txt : 12.38549
inf17195.txt : 12.38549
sbr17995.txt : 12.38549
eos20195.txt : 12.35446
ins04695.txt : 12.35446
ins14595.txt : 12.35446
Ranking based on vector model using Latent Semantic Model:
inf11595.txt : 5.76311
inf21595.txt : 4.66828
inf12795.txt : 4.57714
inf02895.txt : 2.85867
inf19695.txt : 2.63771
emt21795.txt : 2.27342
sbr21395.txt : 2.20714
ins14595.txt : 1.72950
inf17395.txt : 1.70204
sbr18095.txt : 1.66214
inf21695.txt : 1.55942
inf12995.txt : 1.55194
ins13995.txt : 1.46854
emt13295.txt : 1.40700
eos11695.txt : 1.36580
str00795.txt : 1.30627
```

Figure 12: Using Probabilistic and LSM Scoring for Query 3

# Task 5

**Goal :** ***Do Tasks 3 and 4, but this time, remove stopWords from the corpus while cleaning it.***

The `termScores` and `termProbabilityScores` for the corpus are calculated using the same techniques.

## Comparing Models

To check for differences we can run curated queries again, as we did in Task 4 and then check the differences between the models.

### Query 1

**Statement :** ***NASA Project that analysis boundary elements using fracture analysis and dislocations, using the FADD method.***

This query is constructed using the *.key* file of **emt04795.txt**. Below is the result of document scoring by these models.

All the models confidently find the target document. But the certainity of that scoring is reflected in the scores they have given to each of the terms.

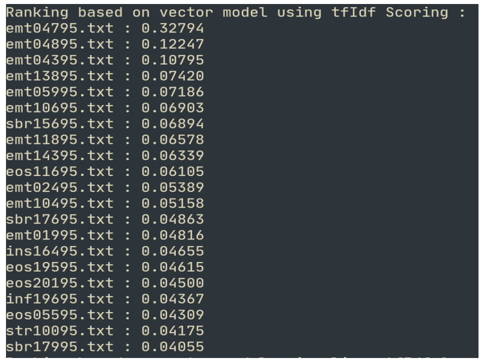
The tfIdf model confidently separates the relevant file with the rest of the corpus, giving a difference of whopping ( 0.2) between the first two files.

The linear tfIdf model is not so far behind either, even though the difference between the first two scores is less than that in the case of tfIdf, it still gives higher absolute score to the relevant document.

The boolean model gives the file with relatively less confidence with an absolute score of only (0.24) for the relevant file and the difference between the first two being (0.11).

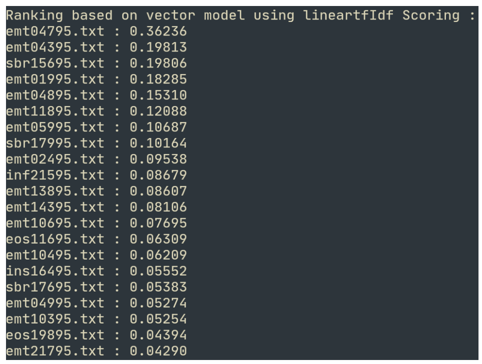
### Query 2

**Statement :** ***Does NASA have any ongoing project on non-destructive evaluation that employs electromagnetic probe technology?***



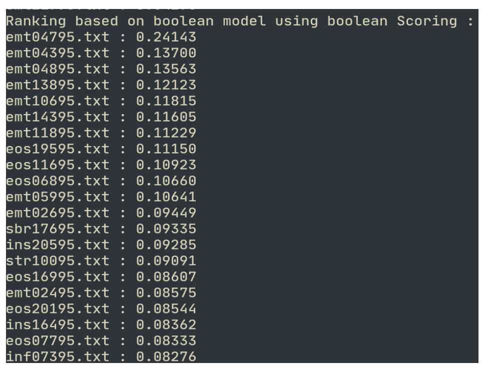
```
Ranking based on vector model using tfIdf Scoring :
emt04795.txt : 0.32794
emt04895.txt : 0.12247
emt04395.txt : 0.10795
emt13895.txt : 0.07420
emt05995.txt : 0.07186
emt10695.txt : 0.06903
sbr15695.txt : 0.06894
emt11895.txt : 0.06578
emt14395.txt : 0.06339
eos11695.txt : 0.06105
emt02495.txt : 0.05389
emt10495.txt : 0.05158
sbr17695.txt : 0.04863
emt01995.txt : 0.04816
ins16495.txt : 0.04655
eos19595.txt : 0.04615
eos20195.txt : 0.04500
inf19695.txt : 0.04367
eos05595.txt : 0.04309
str10095.txt : 0.04175
sbr17995.txt : 0.04055
```

Figure 13: Using Vector Model with tfIdf Scoring for Query 1



```
Ranking based on vector model using lineartfIdf Scoring :
emt04795.txt : 0.36236
emt04395.txt : 0.19813
sbr15695.txt : 0.19806
emt01995.txt : 0.18285
emt04895.txt : 0.15310
emt11895.txt : 0.12088
emt05995.txt : 0.10687
sbr17995.txt : 0.10164
emt02495.txt : 0.09538
inf21595.txt : 0.08679
emt13895.txt : 0.08607
emt14395.txt : 0.08106
emt10695.txt : 0.07695
eos11695.txt : 0.06309
emt10495.txt : 0.06209
ins16495.txt : 0.05552
sbr17695.txt : 0.05383
emt04995.txt : 0.05274
emt10395.txt : 0.05254
eos19895.txt : 0.04394
emt21795.txt : 0.04290
```

Figure 14: Using Vector Model with linear tfIdf Scoring for Query 1



```
Ranking based on boolean model using boolean Scoring :
emt04795.txt : 0.24143
emt04395.txt : 0.13700
emt04895.txt : 0.13563
emt13895.txt : 0.12123
emt10695.txt : 0.11815
emt14395.txt : 0.11605
emt11895.txt : 0.11229
eos19595.txt : 0.11150
eos11695.txt : 0.10923
eos06895.txt : 0.10660
emt05995.txt : 0.10641
emt02695.txt : 0.09449
sbr17695.txt : 0.09335
ins20595.txt : 0.09285
str10095.txt : 0.09091
eos16995.txt : 0.08607
emt02495.txt : 0.08575
eos20195.txt : 0.08544
ins16495.txt : 0.08362
eos07795.txt : 0.08333
inf07395.txt : 0.08276
```

Figure 15: Using Boolean Model for Query 1

This query is constructed using the *.key* file of **ins04695.txt**. Below is the result of document scoring by these models.

Here, with the exception of tfIdf, the other models successfully recognize that the query is most relevant to the file **ins04695.txt**.

The tfIdf model is confused with another file, namely **emt13895.txt**, since that file is also somewhat related to electromagnetic phenomena. The important difference is that in this file, the term *electromagnetic* occurs 4 times as much as in the target file, thereby throwing off the model a bit. Still the model is able to evaluate that the first two documents are much more likely to be the target than the rest, since the difference between the first two is merely ( 0.0017) but the difference between the second and the third document is ( 0.03), a gigantic factor of 17.

Even though the other models correctly predicted the target file, how sure are they ? The first and foremost observation is that these models also seem to think the **emt13895.txt** file to be closely relevant, so the first model didn't perform too bad. The linear tfIdf model seems to be confident that, even though this file is related to the query, it is not the exact file, since it has significant difference

```
Ranking based on vector model using BIM :
inf21595.txt : 15.40467
eos16095.txt : 15.37364
emt13295.txt : 14.03252
emt14395.txt : 14.03252
inf19695.txt : 14.03252
inf12795.txt : 13.91982
inf11595.txt : 13.64494
eos11695.txt : 12.82539
inf02895.txt : 12.82539
inf17395.txt : 12.82539
inf07395.txt : 12.38549
inf17195.txt : 12.38549
sbr17995.txt : 12.38549
eos20195.txt : 12.35446
ins04695.txt : 12.35446
ins14595.txt : 12.35446
Ranking based on vector model using Latent Semantic Model:
inf11595.txt : 5.76311
inf21595.txt : 4.66828
inf12795.txt : 4.57714
inf02895.txt : 2.85867
inf19695.txt : 2.63771
emt21795.txt : 2.27342
sbr21395.txt : 2.20714
ins14595.txt : 1.72950
inf17395.txt : 1.70204
sbr18095.txt : 1.66214
inf21695.txt : 1.55942
inf12995.txt : 1.55194
ins13995.txt : 1.46854
emt13295.txt : 1.40700
eos11695.txt : 1.36580
str00795.txt : 1.30627
```

Figure 16: Using Probabilistic and LSM Scoring for Query 1

```
Ranking based on vector model using tfIdf Scoring :
emt13895.txt : 0.09263
ins04695.txt : 0.09085
ins05395.txt : 0.06860
eos06895.txt : 0.05857
emt10695.txt : 0.05831
inf21595.txt : 0.05308
str00795.txt : 0.05079
ins13995.txt : 0.04920
ins05895.txt : 0.04905
eos19995.txt : 0.04892
sbr06195.txt : 0.04643
mat02095.txt : 0.04251
inf07395.txt : 0.04169
ins09595.txt : 0.04144
eos07795.txt : 0.03950
emt10495.txt : 0.03945
emt05995.txt : 0.03713
eos05595.txt : 0.03708
ins05295.txt : 0.03679
str02595.txt : 0.03515
eos00395.txt : 0.03514
```

Figure 17: Using Vector Model with tfIdf Scoring for Query 2

```
Ranking based on vector model using lineartfIdf Scoring :
ins04695.txt : 0.19852
emt13895.txt : 0.14700
eos06895.txt : 0.11022
inf21595.txt : 0.10857
ins05395.txt : 0.09918
ins05895.txt : 0.08123
eos19995.txt : 0.07852
ins13995.txt : 0.05925
ins05295.txt : 0.04795
str00795.txt : 0.04319
mip14795.txt : 0.03679
sbr12295.txt : 0.03590
mat02095.txt : 0.03103
emt10695.txt : 0.02948
emt05995.txt : 0.02761
ins01795.txt : 0.02732
inf07395.txt : 0.02701
inf12795.txt : 0.02657
sbr06195.txt : 0.02651
emt04595.txt : 0.02646
emt04495.txt : 0.02477
```

Figure 18: Using Vector Model with linear tfIdf Scoring for Query 2

in scores for the two ( 0.05). More than the difference for the second and third position in the tfIdf model.

```
Ranking based on boolean model using boolean Scoring :
ins04695.txt : 0.12457
emt13895.txt : 0.12123
ins05395.txt : 0.12071
emt10695.txt : 0.11815
inf07395.txt : 0.11035
eos16095.txt : 0.10721
eos05595.txt : 0.10710
eos06895.txt : 0.10660
eos00395.txt : 0.10349
str00795.txt : 0.10050
emt10495.txt : 0.09911
sbr06195.txt : 0.09724
inf21595.txt : 0.09428
ins01795.txt : 0.09366
ins07595.txt : 0.09285
mip14795.txt : 0.09206
eos07195.txt : 0.09167
ins09595.txt : 0.09091
ins13995.txt : 0.09072
mip00595.txt : 0.08980
emt04495.txt : 0.08874
```

Figure 19: Using Boolean Model for Query 2

```
Ranking based on vector model using BIM :
emt04795.txt : 13.80343
emt04895.txt : 7.98099
emt04395.txt : 7.36694
emt10695.txt : 6.50789
emt13895.txt : 6.50789
emt05995.txt : 5.37749
emt11895.txt : 5.37749
eos19595.txt : 5.37749
sbr17695.txt : 5.37749
eos05595.txt : 5.17028
str02595.txt : 5.17028
emt14395.txt : 4.95209
eos11695.txt : 4.95209
eos07795.txt : 4.82785
sbr15695.txt : 4.82785
emt10195.txt : 4.59777
Ranking based on vector model using Latent Semantic Model:
emt04895.txt : 6.43603
emt13895.txt : 5.70940
emt10695.txt : 5.18007
emt04395.txt : 4.65169
sbr15695.txt : 4.61003
emt01995.txt : 4.28704
emt05995.txt : 4.06628
sbr17695.txt : 4.02927
emt04795.txt : 3.68546
emt10495.txt : 3.45898
emt11895.txt : 2.69760
eos07795.txt : 2.68853
eos11695.txt : 2.51318
str02595.txt : 2.51044
sbr06295.txt : 2.47187
emt21795.txt : 2.46863
```

Figure 20: Using Probabilistic and LSM Scoring for Query 2

The boolean model on the other hand, figures out the top 2 relevant files, but isn't too sure about which one is the best, since even the following rankings have similar scores.

### Query 3

**Statement :** ***What nasa projects are focused on the development of networks, internet and world wide web?***

This query is constructed using the *.key* file of **inf11595.txt**. Below is the result of document scoring by these models.

At first glance it seems that apart from the linear tf-idf scoring, the other models get confused with a certain other file (**inf12795.txt**). But upon furthur inspection of the keywords present in (**inf12795.key**), we can see that this file is also relevant to the given query!

And even when the models gets confused with this file, we see that they score both the files in top 2 only and that too with marginal difference, the linear tf-idf model, which weighs term frequency more than document frequency, of course confidently predicts the source file as the most relevant file.

So even for queries that have multiple almost equally relevant articles, the models not only retrieve the relevant files out of the corpus, but also confidently distinguishes them from the other files. This

```
Ranking based on vector model using tfIdf Scoring :
inf12795.txt : 0.15928
inf11595.txt : 0.13724
eos16095.txt : 0.12449
emt21795.txt : 0.11206
inf21595.txt : 0.10925
emt14395.txt : 0.10656
inf02895.txt : 0.10130
inf12995.txt : 0.07866
inf17395.txt : 0.06186
emt17495.txt : 0.04700
eos11695.txt : 0.04660
sbr21395.txt : 0.04164
str10095.txt : 0.04029
eos20195.txt : 0.03769
ins14595.txt : 0.03731
sbr17695.txt : 0.03281
ins13995.txt : 0.03210
sbr18095.txt : 0.03131
emt10695.txt : 0.02753
mat02095.txt : 0.02731
inf07395.txt : 0.02601
```

Figure 21: Using Vector Model with tfIdf Scoring for Query 3

```
Ranking based on vector model using lineartfIdf Scoring :
inf11595.txt : 0.30587
inf12795.txt : 0.24080
inf21595.txt : 0.13578
eos16095.txt : 0.12425
emt17495.txt : 0.11803
emt21795.txt : 0.10063
sbr21395.txt : 0.09869
inf02895.txt : 0.07281
emt14395.txt : 0.06986
ins13995.txt : 0.03807
eos11695.txt : 0.03370
inf12995.txt : 0.03257
inf17395.txt : 0.02938
sbr17695.txt : 0.02815
eos20195.txt : 0.02463
mat02095.txt : 0.02451
str10095.txt : 0.02424
sbr18095.txt : 0.02380
ins16495.txt : 0.02058
inf19695.txt : 0.02010
ins20495.txt : 0.01983
```

Figure 22: Using Vector Model with linear tfIdf Scoring for uery 3

```
Ranking based on boolean model using boolean Scoring
inf12795.txt : 0.18803
eos16095.txt : 0.17869
inf02895.txt : 0.17689
emt14395.txt : 0.17408
inf21595.txt : 0.16499
inf12995.txt : 0.16276
inf11595.txt : 0.15696
emt21795.txt : 0.14949
inf17395.txt : 0.14618
ins14595.txt : 0.10997
eos11695.txt : 0.10923
ins21195.txt : 0.09759
mat00695.txt : 0.09535
emt10695.txt : 0.09452
sbr18095.txt : 0.09381
mip09195.txt : 0.09366
sbr17695.txt : 0.09335
ins07595.txt : 0.09285
str10095.txt : 0.09091
ins13995.txt : 0.09072
mip00595.txt : 0.08980
```

Figure 23: Using Boolean Model for Query 3

behaviour is best shown by the linear tf-idf model, which assigns high scores to the first two documents, and the scores after that drop significantly.

On the other hand, this behaviour is least followed by the boolean model, since it scores almost all the documents equally. With the first 5 documents having a span of mere ( 0.016) compared to the span in vector model using linear tf-idf ( 0.167) which is greater by a factor of 10. The tfidf model, resides in the middle, having a span of ( 0.50). Approximately 3 times as good as the boolean model, and approximately 3 times worse than the linear tfIdf model.

Observing the results, we notice that the probabilistic model's rankings remain largely consistent even after excluding stop words. This consistency stems from the increased prominence of essential keywords within the top P stems, as they now surface more prominently following the removal of stopwords. Consequently, we observe a notable improvement in the retrieval of the most relevant document, now situated at the top. Beyond the top results, we also witness significant fluctuations in rankings after the first three, primarily because the absence of stopwords allows for more pertinent keywords to feature prominently among the top P stems, enhancing document differentiation.

```
Ranking based on vector model using BIM :
inf12795.txt : 13.36483
inf11595.txt : 13.15850
inf02895.txt : 12.46957
inf21595.txt : 12.46957
emt21795.txt : 11.58719
emt14395.txt : 10.89826
inf12995.txt : 10.69193
eos16095.txt : 9.14726
inf17395.txt : 7.78814
ins14595.txt : 6.98743
eos11695.txt : 6.59902
emt10695.txt : 6.21683
ins13995.txt : 6.21683
mat02095.txt : 6.21683
sbr18095.txt : 6.21683
sbr17695.txt : 6.01050
Ranking based on vector model using Latent Semantic Model:
inf12795.txt : 10.05626
inf11595.txt : 8.94181
emt21795.txt : 6.21777
inf21595.txt : 5.99006
inf02895.txt : 5.95694
inf12995.txt : 4.55607
sbr21395.txt : 2.24273
ins14595.txt : 2.18170
sbr18095.txt : 2.17653
ins13995.txt : 1.95754
inf17395.txt : 1.89875
mat02095.txt : 1.89107
sbr17695.txt : 1.79376
emt14395.txt : 1.71609
eos11695.txt : 1.55593
emt10695.txt : 1.41039
```

Figure 24: Using Probabilistic and LSM Scoring for Query 3

Likewise, in the case of the LSI model, the upper echelon of rankings exhibits minimal alterations, while the lower rankings undergo substantial changes. This transformation results in the retrieval of documents with much greater semantic similarity to the query than previously observed. This transformation is facilitated by the increased relative importance of relevant keywords due to the absence of stopwords, as they acquire higher tf-idf weights in this context.

## Conclusion Of Comparison and Final Remarks

The tfidf model is accurately able to distinguish between documents that are relevant and documents that are not, but it is inaccurate when it comes to ranking those documents in the exact order of relevance.

The boolean model on the other hand, is able to compare between two documents with ease as to which one is better than the other, giving an accurate idea of importance, but it doesn't seem to able to generalize the result to classify clusters of documents as relevant or non-relevant, giving a very close knitted similarity.

The linear tfIdf model seems to act as the best of both worlds for these queries, not only it confidently categorises relevant and non-relevant documents, it is also able to rank the relevant with greater confidence than the other two.

The probabilistic model closely resembles the boolean model introduced in assignment 1, primarily because it focuses on determining whether a specific term is present within a document or not. However, where it diverges from the boolean model is its incorporation of factors like the probability of relevance and the document frequency of terms. While the boolean model primarily relied on straightforward keyword matching, the probabilistic model takes into account the likelihood of a term's relevance and its rarity within irrelevant documents, thereby optimizing the odds of retrieval. Consequently, it fulfills the same overarching objective as the boolean model but does so with heightened precision.

In contrast, the LSI model shares common ground with the vector model in its utilization of tf-idf weights. Nevertheless, it distinguishes itself by incorporating semantic relationships between terms through the measurement of linear correlations among them. Instead of solely relying on frequency-based metrics like tf-idf scores, the LSI model captures term correlations and distills valuable information while filtering out data noise. This proves especially advantageous in scenarios where the focus extends beyond individual keywords to understanding the collective meaning of term combinations within a given context. The LSI model is proficient at recognizing contextual nuances and delivering documents that align in meaning with the query.

As anticipated, the probabilistic model yields rankings akin to those of the boolean model, while the LSI model produces rankings resembling those of the vector model.