02/04/18

INFORMATION RETRIEVAL SYSTEM

UNIT- I

* Information Retrieval System:-

Information retrieval system is decoded to find relevant document not finding simple matches to patterns.

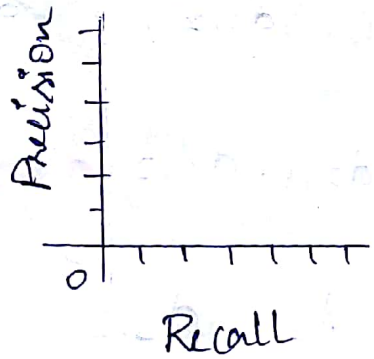
* Precision:-

Precision is the ratio of number of relevant documents to the total no. of documents retrieved.

→ Precision provides an indication of the quality of the answer set.

* Recall:-

Recall considers the total no. of relevant documents retrieved to the total no. of documents in the collection that are delivered to be relevant.



* Retrieval Strategies :-

1. Vector Space Model :

Q : "gold silver truck"

D_1 : "Shipment of gold damaged in a fire"

D_2 : "Delivery of silver arrived in a silver truck".

D_3 : "Shipment of gold arrived in a truck"

The vector space model computes a measure of similarity by defining a vector that represents the query. This model involves constructing a vector that represents the term in the document and another vector that represents the term in the query

Q → $\langle q_0, q_1, q_2, \ldots q_n \rangle$

D_1 → $\langle d_{1,0}, d_{1,1}, d_{1,2}, \ldots d_{1,n} \rangle$

D_2 → $\langle d_{2,0}, d_{2,1}, d_{2,2}, \ldots d_{2,n} \rangle$

D_3 → $\langle d_{3,0}, d_{3,1}, d_{3,2}, \ldots d_{3,n} \rangle$

from the figure

A component of each vector is required for each distinct term in the collection.

→ To construct a vector that corresponds to each document, we consider the following formula.

t = no. of distinct Terms in the document collection.

tf_{ij} = the no. of occurences of term t_j (term j) in the document D_i

This is referred to as term frequency

df_j = no. of documents which contains t_j.

$$idf_j = \log\left(\frac{d}{df_j}\right)$$

↳ Inverse Document frequency

where d = total no. of documents

Similarity coefficient

$$sc(Q, D_i) = \sum_{j=1}^{t} w_{qj} \times d_{ij}$$

docid a arrived damaged delivery fire gold in of

D_1	0	0	0.477	0	0.477	0.176	0	0
D_2	0	0.176	0	0.477	0	0	0	0
D_3	0	0.176	0	0	0	0.176	0	0
Q	0	0	0	0	0	0.176	0	0

shipment	silver	truck
0.176	0	0
0	0.477	0.176
0.176	0	0.176
0	0.477	0.176.

idfa ~~(D_1)~~ = $\log\left(\frac{a}{df_i}\right)$

idfa (D_1) = $\log\left(\frac{3}{3}\right) = \log 1 = 0$

idf arrived = $\log\left(\frac{2}{3}\right) = 0.176$

idf damaged = $\log\left(\frac{1}{3}\right) = 0.477$

idf delivery = $\log\left(\frac{1}{3}\right) = 0.477$

idf fire = $\log\left(\frac{1}{3}\right) = 0.477$

idf gold = $\log\left(\frac{2}{3}\right) = 0.176$

idf in = $\log\left(\frac{3}{3}\right) = \log 1 = 0$

idf of = $\log\left(\frac{3}{3}\right) = \log 1 = 0$

idf shipment = $\log\left(\frac{2}{3}\right) = 0.176$.

idf silver = $\log\left(\frac{1}{3}\right) = 0.477$

idf truck = $\log\left(\frac{2}{3}\right) = 0.176$

↳ Here we need to calculate only the words in document. But the word is also present in the query. Hence we write the value in the query also.

$SC(Q, D_1) = (0)(0) + 0 + 0(0.477) + (0)(0)$
$+ (0)(0.477) + (0.176)(0.176)$
$+ (0)(0) + (0)(0) + (0.176)(0)$
$+ (0)(0) + (0)(0) + (0.176)(0) = 0.031$
$+ (0)(0.477) + 0(0.176)$

$SC(Q, D_2) = 0.486.$

$SC(Q, D_3) = 0.062.$

Hence, ranking could be.

D_2 D_3 D_1

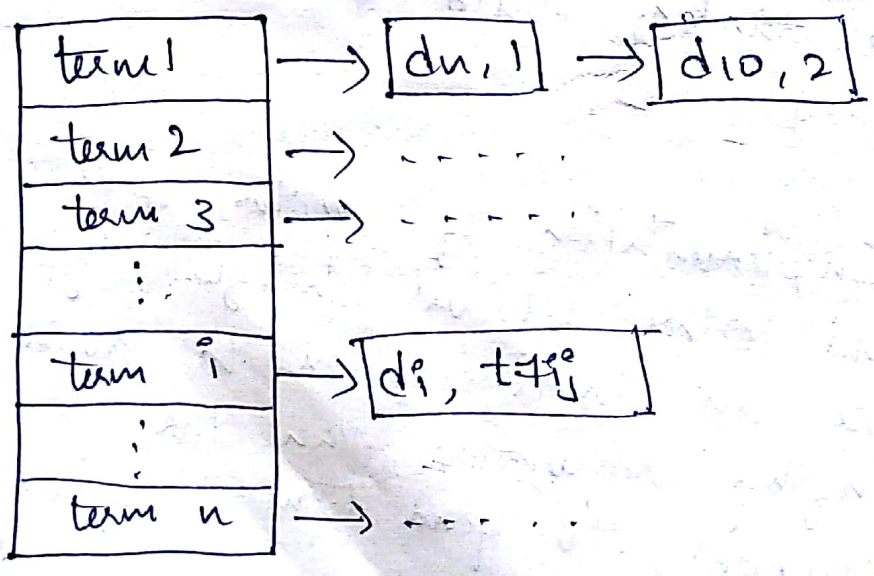
Implementation of vector space model and other retrieval strategies: typically use an invert index to avoid a lengthy sequential scan through every document, to find term in query. Instead an inverted index is generated

prior to the user issuing any queries.

→ The structure of the inverted index which contains an entry for each of the end terms is stored in a structure called the index. For each term a pointer references a logical linked list called the posting list. The posting list contains an entry for each unique document that contains the term, the posting list contains the both a doc. Identifier and the term frequency.

→ The posting list in the figure indicates that the term t_1 appears once in document 1 and twice in doc 10.

→ An entry for an arbitary term t_j indicates that it occurs tf times in document d_j.



→ Many variations were studied in the following weight for term 'j' in document 'i' was defined as w_{ij}

$$w_{ij} = \frac{(\log tf_{ij} + 1.0) * idf_j}{\sum_{j=1}^{t} [(\log tf_{ij} + 1.0) * idf_j]^2}$$

→ This weight is that a single matching term with the high term frequency can skew the effect of remaining matches between a query and a given document.

*Similarity Measure:-

Several different means of comparing a query vector with doc. vector have been implemented.

→ These are well documented and are presented here simply as a quick review.

→ The most common of this are the cosine measure where cosine of angle between the query and document vector is

$$s(Q, D_i) = \frac{\sum_{j=1}^{t} w_{qj} d_{ij}}{\sqrt{\sum_{j=1}^{t} (d_{ij})^2 \sum_{j=1}^{t} (w_{qj})^2}}$$

where $\sqrt{\sum_{j=1}^{t} (w_{qj})^2}$ appears the computation

for every document.

The Dice co-efficient is defined as

$$Sc(Q, D_i) = \frac{2 \sum_{j=1}^{t} w_{qj}\, d_{ij}}{\sum_{j=1}^{t} (d_{ij})^2 \sum_{j=1}^{t} (w_{qj})^2}$$

The Jaccard co-efficient is defined as

$$Sc(Q, D_i) = \frac{\sum_{j=1}^{t} w_{qj}\, d_{ij}}{\sum_{j=1}^{t} (d_{ij})^2 + \sum_{j=1}^{t} (w_{qj})^2 - \sum_{j=1}^{t} q_j\, d_{ij}}$$

* Probabilistic Retrieval Strategy :-

The probabilistic model computes the similarity quotient between a query and a document as the probability the document will be relevant to query.

This reduces the relevant ranking problem to an application of probability theory. The probability theory can be used to compute a measure of relevance between a query and a document. Here two different fundamental approaches were proposed

first ⟶ It relies an user patterns to predict relevance.

Second → It uses each term in query as a clues as to whether or not a document is relevant. The original work on the use of probability theory to retrieve document can be traced. This work develope an area of research where the probability that a document will be relevant, giving a particular term estimated.

* Simple Term Weight :-

$$P\left(\frac{A}{BC}\right) = \frac{P(BC/A)\,P(A)}{P(BC)}$$

By using Bayes Theorem.

P(Sunny

$$q = P(\text{win}, \text{sunny}, \text{good\_short stop})$$

$$q = \frac{P(\text{win}/\text{sunny}, \text{good\_short stop})}{P(\text{sunny}, \text{good\_short stop})} \times P(\text{win})$$

$$1-q = \frac{P(\text{Sunny}, \text{good\_short stop}/\text{lose})\,P(\text{lose})}{P(\text{Sunny}, \text{good\_short stop})}$$

$$q = \frac{P(\text{sunny}, \text{good\_short stop}/\text{win}) \times P(\text{win})}{P(\text{sunny}, \text{good\_short stop})}$$

$$1-q = \frac{P(\text{sunny}, \text{good\_short stop}/\text{lose})\,P(\text{lose})}{P(\text{sunny}, \text{good\_short stop})}$$

$$\frac{\alpha}{1-\alpha} = \frac{P(\text{Sunny}, \text{good\_shortstop} \mid \text{win}) * P(\text{win})}{P(\text{Sunny}, \text{good\_shortstop} \mid \text{lose}) P(\text{lose})} \longrightarrow (1)$$

Solving for first term

$$\frac{P(\text{Sunny}, \text{good\_shortstop} \mid \text{win})}{P(\text{Sunny}, \text{good\_shortstop} \mid \text{lose})}$$

$$= \frac{P(\text{Sunny} \mid \text{win})\ P(\text{good\_shortstop} \mid \text{win})}{P(\text{Sunny} \mid \text{lose})\ P(\text{good\_shortstop} \mid \text{lose})} \longrightarrow (2)$$

$$\frac{\beta}{1-\beta} = \frac{P(\text{Sunny} \mid \text{win})\ P(\text{win}) / P(\text{Sunny})}{P(\text{Sunny} \mid \text{lose})\ P(\text{lose}) / P(\text{Sunny})}$$

$$\frac{\beta}{1-\beta} = \frac{P(\text{Sunny} \mid \text{win})}{P(\text{Sunny} \mid \text{lose})} * \frac{P(\text{win})}{P(\text{lose})}$$

$$\frac{\gamma}{1-\gamma} = \frac{P(\text{good\_shortstop} \mid \text{win})}{P(\text{good\_shortstop} \mid \text{lose})} * \frac{P(\text{win})}{P(\text{lose})}$$

$$\frac{P(\text{Sunny} \mid \text{win})}{P(\text{Sunny} \mid \text{lose})} = \frac{\beta}{1-\beta} * \frac{P(\text{lose})}{P(\text{win})}$$

$$\frac{P(\text{good-short stop} \mid \text{win})}{P(\text{good-short stop} \mid \text{lose})} = \frac{r}{1-r} \times \frac{P(\text{lose})}{P(\text{win})}$$

from eqn (2), (1)

$$\frac{q'}{1-\alpha} = \frac{\beta}{1-\beta} \times \frac{P(\text{lose})}{P(\text{win})} \times \frac{r}{1-r} \times \frac{P(\text{lose})}{P(\text{win})} \times \frac{P(\text{win})}{P(\text{lose})}$$

$$\frac{\alpha}{1-\alpha} = \frac{\beta}{1-\beta} \times \frac{r}{1-r} \times \frac{P(\text{lose})}{P(\text{win})}$$

$$\frac{\alpha}{1-\alpha} = \frac{0.6}{1-0.6} \times \frac{0.75}{1-0.75} \times \frac{0.5}{0.5}$$

where $\beta = 0.6$, $r = 0.75$

$$\frac{\alpha}{1-\alpha} = 1.5 \times 3 \times 1 = 4.5$$

$$\frac{\alpha}{1-\alpha} = 4.5$$

$$\alpha = (1-\alpha)4.5$$

$$\alpha = 4.5 - 4.5\alpha$$

$$5.5\alpha = 4.5$$

$$\alpha = \frac{4.5}{5.5}$$

$$\alpha = 0.818.$$

→ The use of term weights is based on the probability ranking principle which assumes that optimal effectiveness occurs when documents are ranked based on an estimate of the probability of their relevants to a query

*

→ The key is to assigned probabilities to components of the query and then use each of these as evidence in computing the final probability that a document is relevant to the query.

→ The term queries are assigned weights which corresponds to the probability that a particular term in match with a given query will retrieve a relevant document. The weight for each term in the query are combined to obtain a final vision of relevance

Ex:- A soft ball called salamanders will win one of its games. We might observe based on the past experience that they usually win on sunny days when they best shortstop plays. This means that two pieces of evidence outdoor conditions and presence of good shortstop might be used.

→ The probabilistic model computes the similarity between the query and a document as the probability that the document will be relevant to the query. This reduces the relevance ranking problem to an application of probability theory.

→ Two fundamentally different approaches were proposed

→ The first lies on usage patterns

Example Problem:

Q: Gold Silver Truck

D_1: "Shipment of gold damaged in a fire".

D_2: "Delivery of silver arrived in a silver truck".

D_3: "Shipment of gold arrived in a truck".

N = no. of documents collected

n = no. of documents that contain

R = no. of relevant documents for query.

r = no. of relevant documents

→ Choosing 'I', and 'O', yields the following weights

$$w_1 = \log\left(\frac{r/R}{n/N}\right)$$

→ Choose I_2 and O_1 yields the following weights

$$w_2 = \log\left(\frac{r/R}{\frac{n-r}{N-R}}\right)$$

→ Choose I_1 and O_2

$$w_3 = \log\left(\frac{r}{\frac{R-r}{n/N-n}}\right)$$

→ choose I_2 and D_2

$$w_4 = \log\left(\frac{\frac{r}{R-r}}{\frac{n-r}{(N-n)-(R-r)}}\right)$$

→ When incomplete relevance information is available, 0.5 is added to weight to account for the uncertainity involved in estimating relevance.

→ The modify weight function appeared as

$$w = \log\left(\frac{\frac{r+0.5}{(R-r)+0.5}}{\frac{(n-r)+0.5}{(N-n)-(R-r)+0.5}}\right)$$

Ans frequencies of each term

	Gold	Silver	Tancle
N	3	3	3
n	2	1	2
R	2	2	2
r	1	1	2

w_1 for Gold:

$$\log\left(\frac{1/2}{2/3}\right) \Rightarrow \log\left(\frac{3}{4}\right)$$

$\frac{r+0.5}{R+1}$ $\frac{}{n+1}$ $\frac{}{N+2}$

$$w_2 \Rightarrow \log\left(\frac{1/2}{1/1}\right) \Rightarrow \log 1/2$$

$w_3 \Rightarrow \log$

Gold:

$$w_1 = \log\left(\frac{\frac{1.5}{3}}{\frac{3}{5}}\right) \Rightarrow \log\left(\frac{15}{30} \times \frac{5}{3}\right)$$

$$\Rightarrow \log\left(\frac{5}{6}\right) = -0.0791$$

Silver:

$$w_1 = \log\left(\frac{\frac{15}{30}}{2.5}\right) \Rightarrow \log\left(\frac{5}{4}\right) = 0.097$$

Truck:

$$w_1 = \log\left(\frac{\frac{25}{30}}{\frac{3}{5}}\right) \Rightarrow \log\left(\frac{25}{30} \times \frac{5}{3}\right)$$

$$= \log\left(\frac{25}{18}\right) = 0.143$$

Gold:

$$w_2 = \log\left(\frac{\frac{1+0.5}{3}}{\frac{(2+1)-(1+0.5)}{(3+2)-(2+1)}}\right) = \frac{1.5}{3} \times \frac{2}{1.5}$$

$$\Rightarrow -0.1760$$

Silver:

$$w_2 = \log\left(\frac{\frac{1.5}{3}}{\frac{0.5}{2}}\right) \Rightarrow 0.301$$

Truck:

$$w_2 = \log\left(\frac{\frac{2.5}{3}}{\frac{0.5}{2}}\right) \Rightarrow 0.523$$

Gold :

$$w_3 = \log\left(\frac{\frac{1.5}{1.5}}{\frac{3}{2}}\right) \Rightarrow -0.1760$$

Silver :

$$w_3 = \log\left(\frac{\frac{1.5}{1.5}}{\frac{2}{3}}\right) \Rightarrow 0.1760$$

Truck :

$$w_3 = \log\left(\frac{2.5/0.5}{3/2}\right) = 0.522$$

Gold :

$$w_4 = \log\left(\frac{1.5/1.5}{1.5/2-1.5}\right) \Rightarrow -0.4771$$

Silver :

$$w_4 = \log\left(\frac{1.5/1.5}{\frac{0.5}{3-1.5}}\right) = 0.4771$$

Truck

$$w_4 = \log\left(\frac{\frac{2.5}{0.5}}{\frac{0.5}{2-0.5}}\right) \Rightarrow 1.1760.$$

Term Weights :-

	w_1	w_2	w_3	w_4
Gold	-0.0791	-0.1760	-0.1760	-0.4771
Silver	0.097	0.301	0.1760	0.4771
Truck	0.143	0.523	0.522	1.1760

Document Weights

	w_1	w_2	w_3	w_4
Gold	-0.079	-0.176	-0.176	-0.4771
Silver	0.240	0.829	0.698	1.6531
Truck	0.064	0.347	0.347	0.699

→ for doc. D_1, gold is only term to appear, so the weight for D_1 is just the weight for gold

→ for D_2 silver and truck appears. So, the weight for D_2 is sum of weights of silver and truck.

→ for D_3 gold and truck appears, so, the weights for D_3 is sum of weights of gold and truck.

Result: Initial test of

* Non-Binary Independence Model:-

In non binary independence model. we calculate a seperate probability for each term frequency.

→ Consider 10 documents in which document 1 contains the term blue once and the document 2 contains 10 occurances of the term blue.

→ Assume both documents one and two are relevant and the other 8 documents are not relevant.

→ With the simple term weight model we would compute the

$$P(\text{Relevant} \mid \text{one blue}) = 0.2 = \frac{2}{10} = 0.2$$

Because blue occurs in 2 out of 10 relevant documents

→ Hence we conclude the probability that blue will occur one time

→ Because it occurs one time in document 1. ie $P(1 \mid R) = 0.1$

→ Probability that blue occurs 10 times is $P(10 \mid R) = 0.1$

→ Because it occurs 10 times in one out of 10 documents.

→ The probability that a term will result in a non-relevant document is also used.

→ The final weight is computed as the ratio of the probability a term will occur Tf times (Time frequency) in relevant documents to the probability that the term will occur Tf times in non-relevant documents.

$$\log\left(\frac{P(d_i|R)}{P(d_i|N)}\right)$$

Where d_i = document contains occurrences of i^{th} term.

Q: Gold silver Truck

d_1: Shipment of gold damaged in a fire

d_2: Delivery of silver arrived in a silver truck

d_3: Shipment of gold arrived in a truck

* (doc id a arrived damaged delivery fire gold in of shipment silver truck)

doc id	a	arrived	damaged	delivery	fire	gold	in	of	shipment	silver	truck
D_1	1	0	1	0	1	1	1	1	1	0	0
D_2	1	1	0	1	0	0	1	1	0	2	1
D_3	1	1	0	0	0	0	1	1	1	0	1
Q	0	0	0	0	0	0	0	0	0	1	1

doc'd	a	arrived	damaged	delivery	fire	gold	in	of	shipment	silver	truck
D_1	1/7	0	1/7	0	1/7	1/7	1/7	1/7	1/7	0	0
D_2	1/8	1/8	0	1/8	0	0	1/8	1/8	0	2/8	1/8
D_3	1/7	1/7	0	0	0	1/7	1/7	1/7	1/7	0	1/7

→ The training data include both relevant and non-relevant documents.

→ We assume that document 2 and three are relevant and document 1 is not relevant

→ The terms present in the query gold, silver, truck. for D_1 weight of gold is

$$\log\left[\frac{P(1/7|R)}{P(1/7|N)}\right] = \log\frac{1/2}{1/1}$$

→ from these two relevant documents one has frequency of 1/7 and the other doesnot so, $P(1/7|R) = 1/2$.

→ However the only non-relevant document has gold with a frequency of 1/7. So, $P(1/7|N) = 1$.

→ Weights for each term and given term frequency can be computed in this way for each term in document

→ Vector can be constructed and similarity coefficient can be computed between query and each document

* Poisson's Model:-

This model developed a probabilistic model which uses poisson's distribution to estimate probabilities of relevance and incorporate term frequency and document name.

$$w = \log \frac{P(1-r)}{r(1-p)}$$

where, p = probability that term is represent given in that document is relevant

r = probability that the term is represent given that document is non-relevant.

To find term frequencies we are using ptf (probability of term frequency). and the waiting then becomes

$$w = \log \left(\frac{Ptf\ (rel)}{rtf\ (no)} \right)$$

→ where q_{tf} is probability of term frequency of non-relevant document and subscript '0' indicate absence. This assumption is made that term randomly occur within the document according to poisson's distribution.

→ The term frequency and document length that have similar values from observation. To incorporate the term frequency we use another function.

$$w' = w \frac{tf}{k_1 + tf}$$

where w is standard probabilistic weight and k_1 is unknown constant whose value depends on the collection and must be determined experimentally. The simplest mean to account document length is to modify the equation given above for w' by substituting value of $k_1 = \frac{(k_1)(d)}{\Delta}$

where d = document length

Δ = average document length.

$$w_1 = w \left[\frac{tf}{\frac{k_1(d)}{\Delta} + tf} \right]$$

→ The symmetry between documents and query is used to calculate the query term frequency in same way similar to the document frequency.

→ A tuning parameter K_1 is used to scale the effect of document term frequency. Similarly another parameter K_3 is used to scale the query term frequency qtf. finally closeness match to the poison estimation can be attempted with additional term possessing scaling factor of K_2

$$K_2\left[|Q|\left(\frac{\Delta - d}{\Delta + d}\right)\right]$$

→ this term K_2 is constant i.e., experimentally determined. $|Q|$ is the no. of query terms. Now this term enables a high value of K_2 to given addition emphasis to documents that are shorter than average.

→ These additional result in similarity coefficient

$$SC(Q, D_i) = \sum_{j=1}^{t} \log\left[\frac{r/R - r}{\frac{n-r}{(N-n)-(R-r)}}\right] \left[\frac{tf_{ij}}{\frac{K_1 dl_i}{\Delta} + tf_{ij}}\right] \times \left[\frac{qtf_i}{K_3 + qtf_j}\right] \times \left[K_2 |Q| \left(\frac{\Delta - dl_i}{\Delta + dl_i}\right)\right]$$

where
N = No. of documents in collection
n = No. of documents indexed by a given term
R = No. of relevant documents for query

r = No. of relevant docs indexed by the given term.

tf_{ij} = Term frequency of term j in document i.

qtf_j = Term frequency of term j in query 'Q'.

dl_i = No. of terms in document i.

$|Q|$ = No. of terms in the query.

Δ = Average document length.

k_1, k_2, k_3 = Tuning parameters.

→ k_1, k_2, k_3 have effect of reducing the impact of term frequency and query term frequency - If either '0' of k_1 and k_3, the effect is to eliminate the quantity

→ Large values of k_1 and k_3 result in significantly reducing the size of the first term. Including a factor of k_1+1, k_3+1 in the numerator doesnot effect the overall ranking, because these factors apply equally to all documents

$$K = k\left[(1-b) + b\left(\frac{dl_i}{\Delta}\right)\right].$$

$$SC(Q, D_i) = \sum_{j=1}^{t} \log\left[\frac{\frac{r}{R-r}}{\frac{n-r}{(N-n)-(R-r)}}\right] * \left[\frac{(k_1+1)tf_{ij}}{K+tf_{ij}}\right]$$

$$* \left[\frac{(k_3+1)qtf_{ij}}{k_3+qtf_{ij}}\right] + \left[k_2|Q|\left(\frac{\Delta-dl_i}{\Delta+dl_i}\right)\right]$$

For experiments conducted in these values were ~~taken~~ taken as $k_1=1$, $k_2=$ $k_3=8$, $b=0.6$.

→ Small values of K_1, K_2, K_3 have the effect of reducing the impact of term frequency and query term frequency.

→ If either '0' of K_1 and K_3, the effect is to eliminate the quantity

→ Large values of K_1 and K_3 result in significantly reducing the size of the first term.

→ Including a factor of K_1+1 and K_3+1 in the numerator doesnot effect the overall ranking because these factors apply equally to all the documents.

$$K = K_1\left((1-b) + b\left(\frac{dl}{\Delta}\right)\right)$$

$$SC(Q, D_1) = \sum_{j=1}^{t} \log\left(\frac{\frac{r}{R-r}}{\frac{(n-r)}{(N-n)-(R-r)}}\right) \times \left(\frac{(K_1+1)tf_j}{K+tf_j}\right)$$

$$\times\left(\frac{(K_3+1)\,qtf}{K_3+qtf_j}\right) + \left(K_2|Q|\frac{\Delta-dl}{\Delta+dl}\right)$$

for experiments conducted in these value were taken as

$K_1 = 1$
$K_2 = 0$
$K_3 = 8$
$b = 0.6$

for:-

q: Gold Silver Truck

d_1: Shipment of gold damaged in a fire

d_2: Delivery of silver arrived in a silver truck.

d_3: Shipment of gold arrived in a truck

Using the above documents and query with previously completed ~~IDF~~ as computed. IDF as:

gold -0.477

Silver 0.477

truck 1.176.

$$\text{avg } dl = \frac{22}{3} = 7.33$$

Using the same parameters k_1, k_2, k_3.

~~for~~ ~~df~~ k_1. IDF computed for

d df

$df_1 = 7$

$df_2 = 7$

$df_3 = 7$

→ for document d_1 the only match is gold, which appears with a term frequency $tf = 1$, so the similarity coefficient for Sc for $d1$ is just

the value of gold.

→ Now that we have the value of k it is possible to compute the similarity co-efficient of d1.

→ The coefficient is a summation for all terms, but only one term gold will have a non-zero value.

→ We start with the value of $\boxed{w_4 = -0.477}$

$$SC = \left[\frac{(1+1)1}{0.97371}\right]\left[\frac{(8+1)(1)}{8+1}\right] \times (0.477) + \left[0\left(\frac{\Delta - dk}{\Delta + dk}\right)\right].$$

for document d_2

→ The terms that match that query : silver and truck - result in non-zero values in the summation.

value K :

$$K = K_1 \left((1-b) + b \left(\frac{dl_i}{\Delta} \right) \right)$$

$$K = 1 \left((1-0.6) + 0.6 \left(\frac{8}{7.33} \right) \right)$$

$$K = 1 \left(0.4 + 0.6 \left(\frac{8}{7.33} \right) \right)$$

$$K = 1.055$$

*Language Models :-

A statistical model is a probabilistic mechanism for generating a piece of text.

→ It defines a distribution over all possible word sequences.

→ The simplest language model is unigram language model which is essentially a word distribution.

→ The core idea is that documents can be ranked on likely note of generating the query

→ Consider the spoken document recognition. If the speakers work with the words in a document.

→ What is the likelyhood they say the words in the documents query?

$$SC(Q, D_i) = P(Q, M_{D_i})$$

where empty i is language model

→ ~~The~~ probability of the query is -

→ The probability of query is calculated as product of the probabilities of both the terms present in the query and the terms absent.

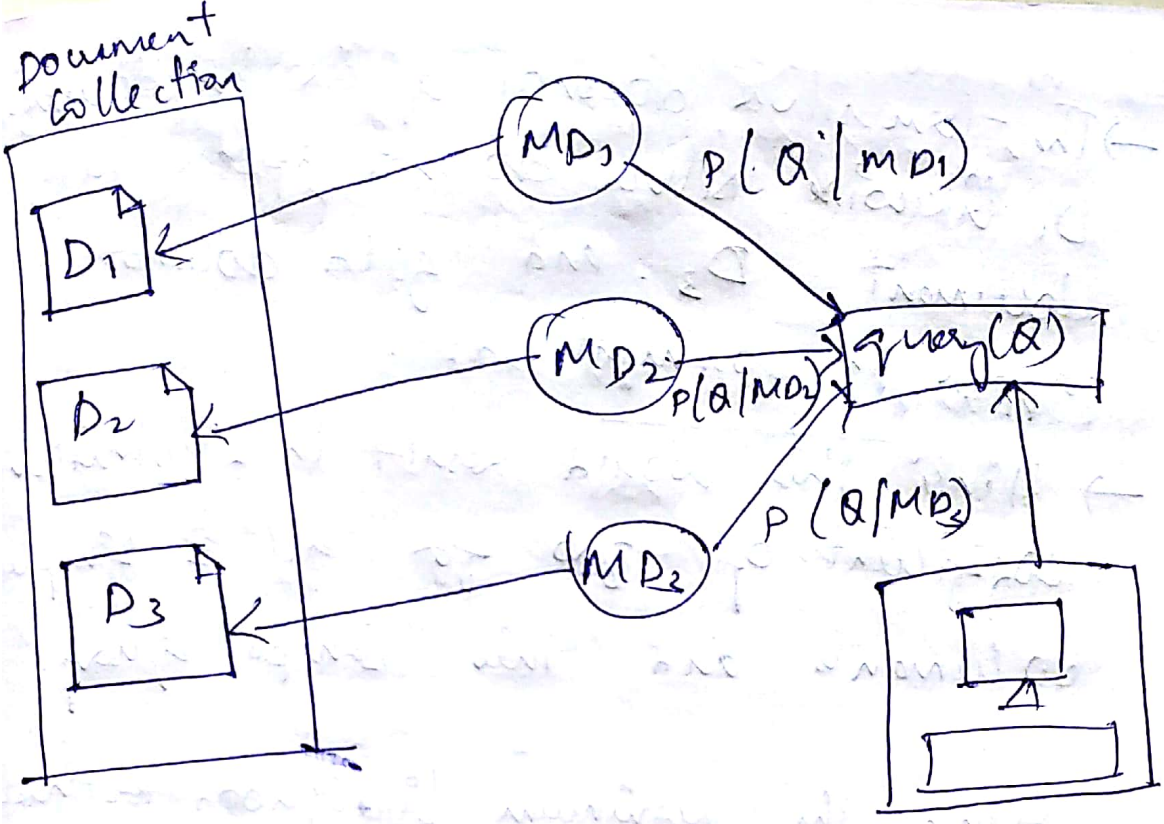
$$SC(Q, D_i) = \prod_{t_j \in Q} P(t_j | M_{D_i}) \prod_{t_j \notin Q} (1 - P(t_j | M_{D_i}))$$

$P(t_j | M_{D_i})$ can be estimated in many different ways.

$$P(t_j | M_{D_i}) = P_{ml}(t_j | M_{D_i}).$$

where $P_{ml}(t_j | M_{D_i})$ the maximum likelyhood estimate of the term distribution is given by $P_{ml}(t_j | M_{D_i}) = \frac{tf(t_j, D_i)}{dl_{D_i}}$.

where dl_{D_i} is the document length of document D_i.



→ The basic idea illustrated in the figure, the similarity measure will work but it has a big problem.

→ If a term in the query doesn't occur in a document the whole similarity measures become 'zero'.

→ Consider a small running example of a query and three documents.

Q : gold silver truck

D_1 : Shipment of gold damaged in a fire

D_2 : Delivery of silver arrived in a silver truck

D_3 : Shipment of gold arrived in a truck.

→ The term silva doesnot appear in document D_1 likewise silva doesnot appear in document D_3. and gold doesnot appear in document d_2.

→ Hence this would result in a similarity coefficient of zero for all 3 sample documents and this sample query

→ Hence the maximum likelyhood estimate for

$$P_{ml}(Silva | M_{Di}) = \frac{tf(Silva, Di)}{d(Di} = 0$$

* Smoothing :-

To avoid the problem caused by terms in the query not present in a document, various smoothing approaches exits which estimate non-zero values for this terms.

→ One approach assumes that the query term occur in this model, but simply at no higher a rate than the chance of it occuring in any other document.

→ Smoothing is the ratio of

$$\frac{C_{ft}}{C_S}$$

where C_{ft} → The no. of occurences of term t
C_S → Total terms collection in entire document.

→ In our example the estimate for silver

$$= \frac{2}{22} = 0.091.$$

To use a large values the estimate value will become.

$$P_{avg} = \frac{\sum_{d \in d_{ft}} P_{ml}(t/M_d)}{d_{ft}}$$

d_{ft} = document frequency of term 't'.

→ To improve effectiveness of estimates for term weight it is possible to minimize the risk involved in one estimation. We first define $\bar{f}_t$ as mean term frequency of term 't' in the document.

$$\bar{f}_t = P_{avg}(t) \times d_{ft}$$

The risk can be applied by calculating

$$R_{td} = \left(\frac{1.0}{1.0 + f_t}\right) \times \left(\frac{\bar{f}_t}{1.0 + \bar{f}_t}\right)$$

→ The first similarity measure described by using language models in information retrieval uses the smoothing ratio $\frac{cf_t}{CS}$.

* Language Model Example:-

There are 22 tokens i.e., $CS = 22$. The total no. of tokens in document d_1, d_2, d_3 are

	D_1	D_2	D_3
d_i	7	8	7

df_t, we use same term ordering in document d_1, d_2, d_3. So, we have to consider the table

	a	arrived	damaged	delivery	fine	gold	in	of
df_t	3	2	1	1	1	2	3	3
cf_t	3	2	1	1	1	2	3	3

	shipment	silver	truck
df_t	2	1	2
cf_t	2	2	2

cf_t is term count of token 't'.

	a	arrived	damaged	delivery	fire	gold	in	of	shipment	silver	truck
D_1	1	0	1	0	1	1	1	1	1	0	0
D_2	1	1	0	1	0	0	1	1	0	2	1
D_3	1	1	0	0	0	1	1	1	1	0	1

→ The raw term frequency of term t in document 'd'.

$$Pml(t|M_{Di}) = \frac{tf(t, Di)}{dl_{Di}}$$

Maximum Likelihood for Each Term is

$Pml(t|M_{Di})$	D_1	D_2	D_3
a	1/7 = 0.143	1/8 = 0.125	1/7 = 0.143
arrived	0	0.125	0.143
damaged	0.143	0.5	0
delivery	0	0.125	0
fire	0.143	0	0
gold	0.143	0	0.143
in	0.143	0.125	0.143
of	0.143	0.125	0.143
Shipment	0.143	0	0.143
silver	0	0.25	0
truck	0	0.143	0.143

→ Second we calculate mean probability of term in document which contains the term.

$$P_{avg}(t)_{11} = \frac{\sum_d (t \in d)\ P_{ml}(t|M_d)}{\cdot 1, dft}$$

for the term arrived, it only appear in document d_2, d_3.

$$P_{avg}(arrived) = \frac{P_{ml}(arrived|M_{D_2}) + P_{ml}(arrived|M_{D_3})}{df\ arrived}$$

- $P_{ml}(arrived|M_{D_2}) = 0.125$
- $P_{ml}(arrived|M_{D_3}) = 0.143$

$$= \frac{0.125 + 0.143}{2} = \frac{0.268}{2} = 0.134.$$

* Mean Term frequency :-

ft	a	arrived	damaged	delivery	fire	gold	in	of	shipment	silver	term
D_1	0.958	0.938	1.000	0.875	1.000	0.958	0.958	1.000	1.000	1.750	0.958
D_2	1.046	1.071	1.143	1.000	1.143	1.096	1.046	1.143	1.143	2.000	1.046
D_3	0.958	0.938	1.000	0.875	1.000	1.000	0.958	0.958	1.000	1.750	0.958

Risk factory $R_{t,d} = \left(\frac{1.0}{1.0 \times ft}\right) \times \frac{ft}{1.0 \times ft}$

→ The risk value of the document can be collected in the table.

R t, d	D_1	D_2	D_3
a	0.250	0.249	0.250
arrived	0.516	0.250	0.250
damaged	0.250	0.467	0.500
delivery	0.533	0.250	0.533
fire	0.250	0.467	0.500
gold	0.250	0.467	0.250
in	0.250	0.249	0.250
of	0.250	0.249	0.250
Shipment	0.250	0.467	0.250
Silver	0.364	0.148	0.364
truck	0.516	0.249	0.250

→ We use the risk value as a mining parameter to calculate

$$P(Q|M_d)$$

→ The probility of producing the query for a given document models has mentioned before. It consists of two step.

→ Initially we calculate $P(t/M_s)$ and for all other term occurances. The smoothing estimate $P(t/M_d) = \frac{Cft}{Cs}$ is used

$P(t/t_d)$ D_1 D_2 D_3

a

$$P(t|M_{di}) = \begin{cases} P_{ml}(t,d)^{(1-R)(t,d)} \times P_{avg}^{(t,d)} & ; \text{if } tf(t,d) > 0 \\ \frac{Cft}{Cs} & \text{otherwise} \end{cases}$$

finally we compute a final measure of similarity

$P(t/t_d)$	D_1	D_2	D_3
a	0.141	0.128	0.141
arrived	0091	0.127	0.141
damaged	0.143	0.045	0.045
delivery	0.045	0.125	0.045
fire	0.143	0.045	0.045
gold	0.143	0.091	0.145
in	0.141	0.128	0.141
of	0.141	0.128	0.141
Shipment	0.143	0.091	0.143
Silver	0.091	0.250	0.091
truck	0.091	0.127	0.141

$$P(Q|M_d) = \prod_{t \in Q} P(t|M_d) \times \prod_{t \notin Q} (1.0 - P(t|M_d))$$

Similarity using language Model

	D_1	D_2	D_3
$P(Q\|M_d)$	0.000409	0.001214	0.000748