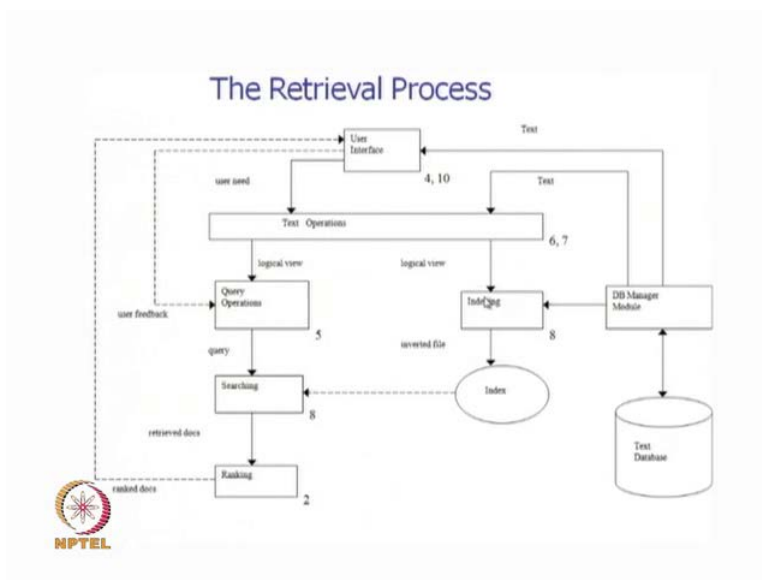


**Natural Language Processing**  
**Prof. Pushpak Bhattacharyya**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Bombay**

**Lecture - 24**  
**IR Models; Boolean Vector**

Today, we are going to discuss, an important topic that of information retrieval model. And in particular will talk about two models namely: Boolean and vector, all these we remember is in the context of the relationship between natural language processing and information retrieval. We have remarked already that both the fields process text and how in response to a users query a document is retrieved that is relevant and satisfies the information need of the user is the main concerned. Now, in this particular lecture, we will essentially do well on, this particular point as to how the terms of a document are weighted, how the terms in the query are weighted and then the relevance document is computed.

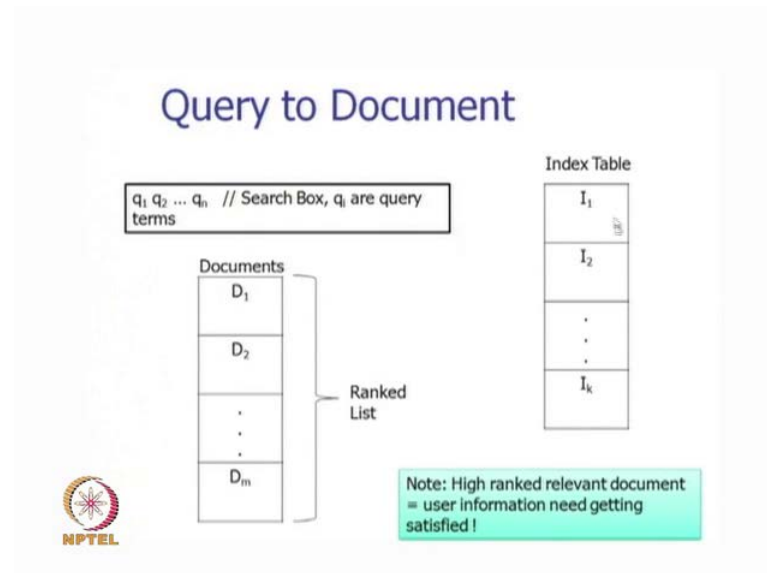
(Refer Slide Time: 01:21)



So, proceeding we just remember that, the retrieval process happens in the following way, there is a user interface and the user information need is expressed in the form of a query. The query is used for searching the document, the documents are ranked and shown back to the user. As a background offline processing, the text data base is created and a data base manager module produces the pages of the information and these are the

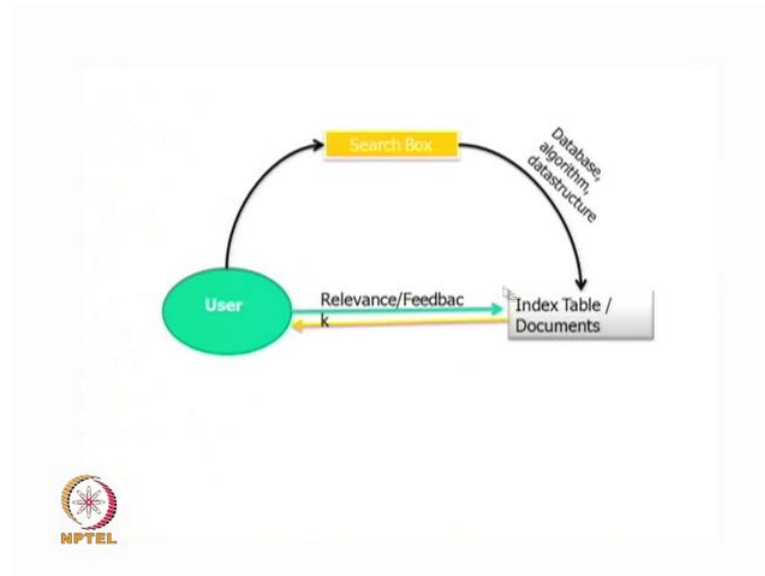
pages which are shown to the user, their indexed and various kind of text operations are carried out to match the users query along with the indexed pages. So, this is the basic semantic there is offline processing on the pages and online processing on the query.

(Refer Slide Time: 02:26)



Now, query to document transfer so, to say happens through a list of documents which are indexed so, there is this index table. The query consist of these n words a particular query shown here q<sub>1</sub>, q<sub>2</sub>, q<sub>n</sub> so, these could be a query like tourism to Taj Mahal which has a three words q<sub>1</sub>, q<sub>2</sub> and q<sub>3</sub> tourism to Taj Mahal. The index table would contained the terms like tourism and then Taj Mahal 2 will not be contained for reason, which was explained last time, 2 is a stop words, it is highly non discriminatory word that means it does not have any specific role to play with respect to the specific information of the document. So, index table contains this k indices I<sub>1</sub> to I<sub>k</sub> and when we have the documents displayed such that, highly relevant document are occupying high rank in this list then the users information need is satisfied. So, we have a rank list of documents here.

(Refer Slide Time: 03:45)



Now, what happens is that the user gives his query to the search box, then data based algorithm data structure all these area of computer science come into play, the index table and documents are created there is a relevance feedback given to the user in terms of the document which have been retrieved. And this closes the loop then progressively the quality of the documents retrieve retrieved improves.

(Refer Slide Time: 04:13)

### How to check quality of retrieval (P, R, F)

- Three parameters
  - Precision  $P = |A \cap O| / |O|$
  - Recall  $R = |A \cap O| / |A|$
  - F<sub>1</sub> score =  $2PR / (P+R)$ 
    - Harmonic mean

The Venn diagram consists of two overlapping circles. The left circle is labeled 'Actual(A)' and the right circle is labeled 'Obtained(O)'. The intersection of the two circles is labeled 'A ^ O'. The NPTEL logo is in the bottom left corner, and a yellow box at the bottom contains the text: 'the above formula are very general. We haven't considered that the documents retrieved are ranked and thus the above expressions need to be'.

We recapitulate, the method to find out accuracy and the quality of the retrieval. There are 3 parameters namely: precision, recall and F score, which are used to calculate a

accuracy of the retrieved document so, you can see here precision is defined in terms of  $A \cap O$  divided by the size of  $O$ , where these diagram is there to explain  $A$  and  $O$ ,  $O$  is the obtained list of document,  $A$  is the actual list of the documents. And we should look up and them as sets rather than lists so,  $A \cap O$  is the documents which are retrieved from the actual set ok. There is this agreement area  $A \cap O$  where actual document and obtained documents agree.

Now, this precision is a ratio of  $A \cap O$  and  $O$ , this means what is the proportion of obtained documents? That is correct this is precision. Recall is the ratio of  $A \cap O$  and the size of  $A$  so, this is the proportion of actual document which are retrieved actually correct documents which are retrieved  $F$  score is obtained from the harmonic mean of  $P$  and  $R$   $2 \times P \times R$  divided by  $P + R$  and we remark last time that if one improves on harmonic mean, automatically geometric mean and arithmetic mean also improve because, harmonic mean is the least of this 3 means.

(Refer Slide Time: 06:09)

### $P, R, F$ (contd.)

- Precision is easy to calculate, Recall is not.
- Given a known set of pair of  $\langle q, D \rangle$
- Relevance judgement  $\langle q, D \rangle \rightarrow \{0, 1\}$   
(Human evaluation)



Now, precision, recall and  $F$  score are easy to calculate, mathematically but, in actual practice, for information retrieval when document are retrieved and the document list is shown to the user, a rank list of documents is shown to the user. Then the precision is easy to calculate but, recall is not. So, let us understand this particular point, precision is easy to calculate because, we have list of documents in front of us. And let us say we

decided to go up to the k-th position k could be 10, 20, 5 and so on. Normally what is done is shown here I would like to draw a picture.

(Refer Slide Time: 07:00)

Ranked List of Docs.

$P@5$   
— Precision at 5

$D_1$
$D_2$
$D_3$
$D_4$
$D_5$

$D_2, D_4$   
are relevant

$$P@5 = \frac{2}{5} = 40\%$$

NPTEL

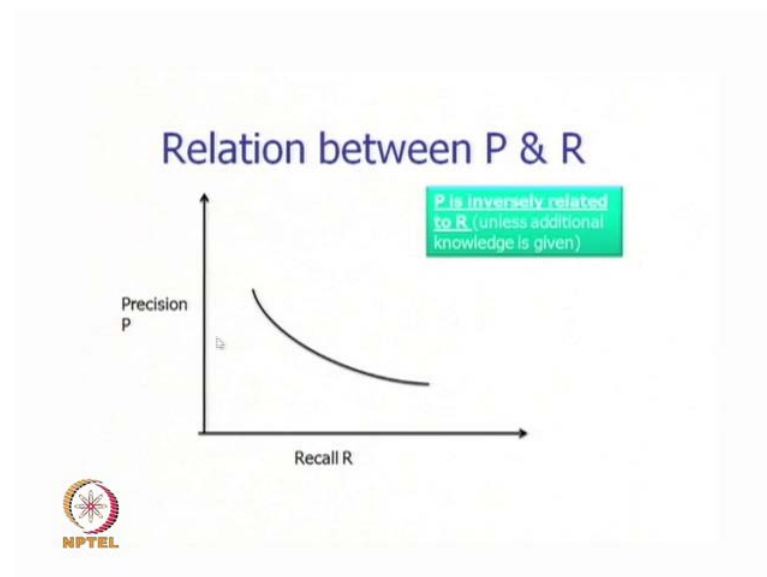
NLP-lect 26

So, we have the ranked list of dogs and here are the documents, D 1, D 2, D 3, D 4 and D 5. Now, the concept of P at 5, that is precision at 5 is P at 5 is that, out of these documents let say D 2 and D 4 are relevant then P at 5 is 2 by 5, which is 40 percent. So, precision is quite easy to calculate but, consider what will happen, when we want to calculate recall. So, how will that be done we have already seen that out of this 5 documents D 2 and D 4 are relevant ok. So, they are from the list of actual relevant documents but, if we want to calculate the recall, then we will have to know the actual list of documents which are relevant.

If that number is known, then in that case we can divide these values these D2 and D 4 by that is 2 by the total number of actual documents. So, at a web scale when we have the whole internet to be considered, then it is impossible to know the actual list of relevant documents as they exist on the web. So, computing recall therefore, it is a very difficult proposition it is impossible to do so, at the scale of the web. So, that is why looking at the slide again, we are commenting that precision is easy to calculate recall is not given a known set of pair of q and D, that is query and document pair, relevance judgment is produce, which has value 0 or 1 so, q, D is 0 if the document D is relevant for the query q and if it is irrelevant for the query q, otherwise it is 1.

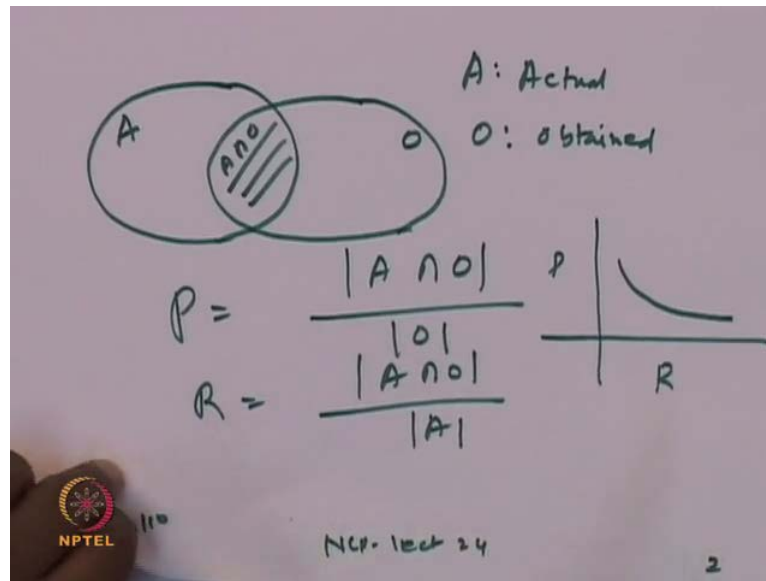
So, 1 indicate this document  $D$  is relevant for the query  $q$  and this is typically done by human evaluation. So, human being actually sit down with a query and produce and launch searches and whichever, document are retrieved they actually read them and from the reading if they see that this document is relevant to the query, then they produce a mark 1, along with the query. And the document and this is the way a relevant pool of judgment is created ok.

(Refer Slide Time: 10:03)



So, now we look at the graph here on the slide, which is the relationship between  $P$  and  $R$ . This is the typical behavior of precision verses recall curve typical behavior for recall verses precision curve so, as recall increases precision falls and wise versa so, precision recall have a inverse relationship. Now, a let us see why this behavior is seen, we would like to show it on the paper

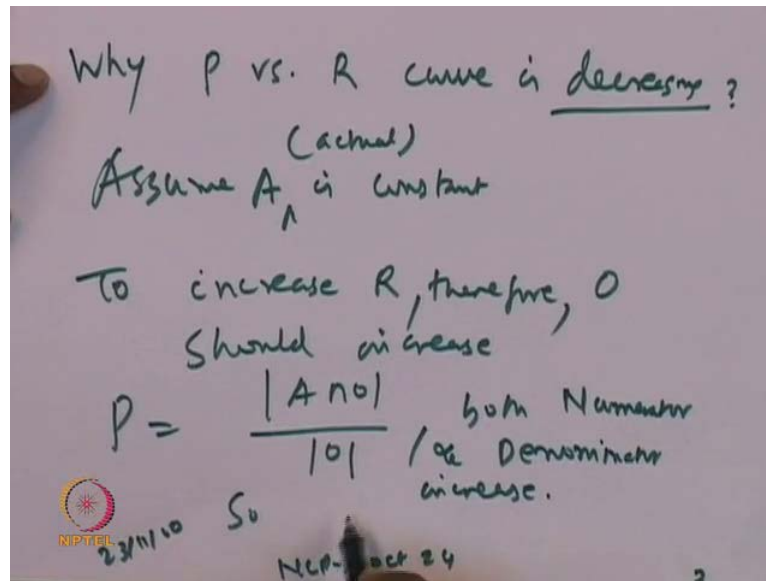
(Refer Slide Time: 10:38)



So, we have already seen that precision and recall are computed from, the actual and obtained entities so, this set is A, this set is O, A is actual and O is obtained. Now, P is equal to A intersection O divided by O, recall on the other hand is A intersection O divided by A now, and we are saying that, the precision verse recall curve, P and R curve comes this way it is following curve. So, this is a typical behavior and it is important to reflect on y, this is the case. Now, let us see if the recall must improve, then either A should go down or A intersection O should go up, that is increase ok.

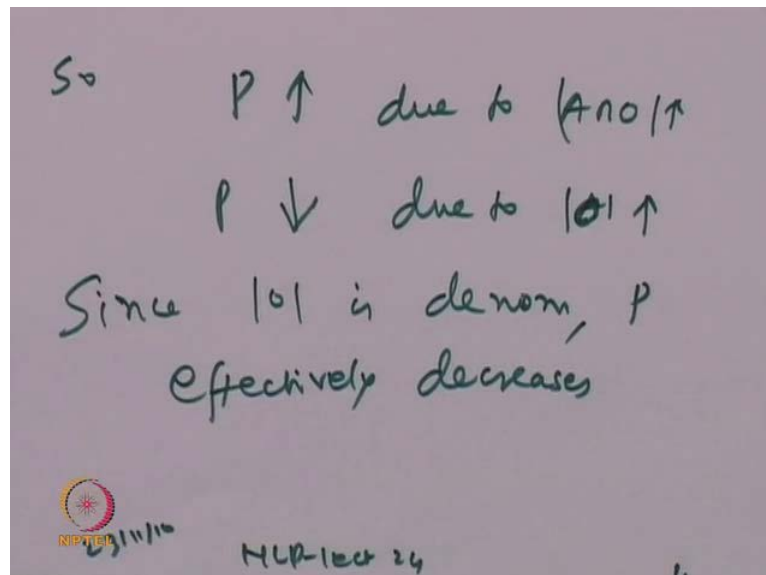
Now, let us assume that the actual set of entities, which are correct that does not change let us suppose A does not change. So, if A does not change the only way recall could go up is by increasing O ok, by increasing O so, as O increases as, the denominator of P increases denominator of P increases numerator also increases but, as we know that the same entity appearing in the numerator and denominator, would make the denominator more predominant then the numerator. What you mean by this? Is that as O increases denominator increasing numerator increasing, the effect will be felt more from the denominator. So, that means the rate at which P will increase due to numerator will be less than the rate at which the denominator, at the rate at which P decreases because, of the denominator.

(Refer Slide Time: 13:08)



So, let us write this points down, why P verses R curve is decreasing? So, assume A is constant, which is the actual set of things A is constant. To increase R therefore, O should increase, to increase R therefore, O should increase. So, P A intersection O by O both numerator and denominator increase.

(Refer Slide Time: 14:05).



So, P goes up due to A intersection O going up, P goes down due to A going up, O going up since, O is denom, P effectively decreases so, that is the reason. Now, another way P could increase is by increasing the overlap between A and O so, both A and O remain



same but, it is just the fact that O goes inside more into A and there by A intersection O increases so, if that, happens then of course, both precision and recall increases but, normally that does not really happen ok. So, going to the slide now we have understood the behavior precision and increasing 1 decreases the other.


So, here we are making an important remark unless additional knowledge is given so, P and R have inverse relationship with each other, as recall increases P would go down so, is that sacrosanct will that always happen, that will not happen if it so happen that the overlap between A and O decrease without disturbing the actual values of A and O. So, that means the intersection area A and intersection area between A and O increases. So, this can be done by increasing the effectiveness of search, by injecting additional knowledge ok. So, which is where the role of natural language processing comes in, in case of information retrieval so, how does one bring the O area closer into the A area.

So that the intersection area increases, that would increase only if we have more effective search and that it is seen typically happens, if we inject more knowledge into the system, if we make use of the text which is retrieved the information content of the text which is retrieved, the meaning of the document and so on. So, that is where language processing plays a role so, probably P versus R curve, which is decreasing can be offset this problem can possibly be solved by introducing language knowledge, domain specific knowledge, meaning information and so on ok. So, let us mark this point here on the slide that P is inversely related to R, unless additional knowledge is given.

(Refer Slide Time: 17:06)

### Precision at rank k

- Choose the top k documents, see how many of them are relevant out of them.
- $P_k = (\# \text{ of relevant documents})/k$
- Mean Average Precision (MAP)  
$$= \frac{1}{L} \sum P_k$$

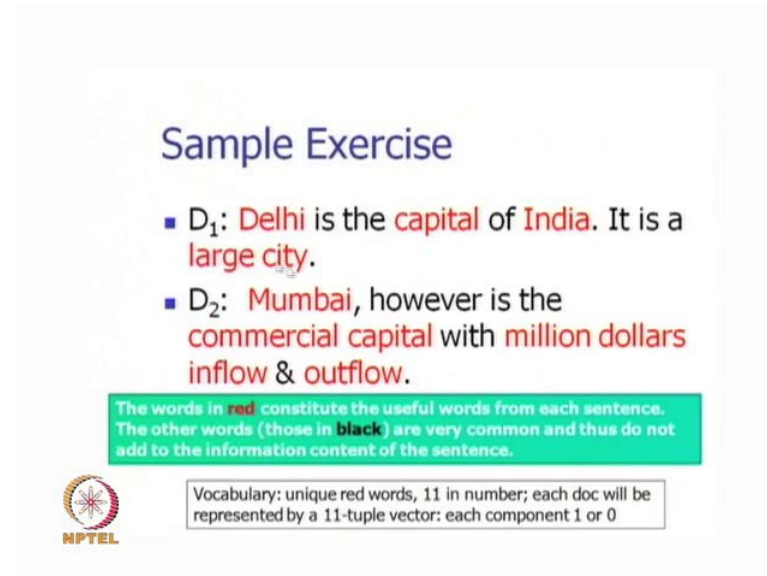


Documents
D <sub>1</sub>
D <sub>2</sub>
⋮
⋮
D <sub>k</sub>
⋮
⋮
D <sub>n</sub>

Now, there is this important parameter, which we started discussing precision at rank k, choose the top k document see how many of them are relevant out of them. So,  $P_k$  is equal to number of relevant documents divided by k, this is  $P_k$  at k precision at rank k. And the mean average precision is obtained mean average precision is obtained in this way, that is  $\frac{1}{L} \sum P_k$ . So, what it means is that, why it is called mean average precision map, there are 2 mean as you can see here 2 averages mean is an average, average is also an average or mean so, what happens is that for a particular retrieval we can vary k.

So, we can keep on calculating different precision, precision at 1 at 2 at 3 at 4 and so on, up to let say about 100 so, we collect all these precision values we collect all these precision values. Then the number of times this precision values have been calculated, these sum of precision are taken and we divide it by number the number times these has been done ok. And we do it for many queries so, the precision values the average precision values of different queries are taken and, when we take the average of this average precision values we get the mean average precision. So, map value is a very important parameter for calculating the effectiveness of the search system.

(Refer Slide Time: 19:04)




**Sample Exercise**

- D<sub>1</sub>: Delhi is the capital of India. It is a large city.
- D<sub>2</sub>: Mumbai, however is the commercial capital with million dollars inflow & outflow.

The words in red constitute the useful words from each sentence. The other words (those in black) are very common and thus do not add to the information content of the sentence.

Vocabulary: unique red words, 11 in number; each doc will be represented by a 11-tuple vector: each component 1 or 0




Now, we do a small sample exercise, suppose our document D 1 the 1st document is Delhi is the capital of India, it is a large city ok. D 2 is another document which contains a text Mumbai however, is the commercial capital with million dollars inflow and outflow. So, 2 pieces of text 2 documents let say Delhi is the capital of India, it is a large city Mumbai however, is a commercial capital with million dollars inflow and outflow. So, the words in red constitutes the useful word from each sentences, the other word those in black are very common and thus do not add to the information content of the sentences.

So, things like the they appear in all documents all sentences so, the, the presence of the does not make D 1 anything special and similarly, does not make D 2 anything special. So, the vocabulary is constituted of the red words which appear in this document, which means they are the words which are stop words content words and are important for the documents so, here the red words are unique red words are eleven in number each document will be represented by a eleven tuple vector each component being 1 or 0.

So, if the Delhi is the 1st word in the vocabulary, then the vector of D 1 will have a 1 in position 1, then if capital is 2nd word in the vocabulary then Delhi will have this document D 1 will have a 1 in the position of capital ok. Similarly, for India large and city, Mumbai will come after that we will have a 1 there, commercial capital will come after that with million dollars inflow and outflow. So, for each of these words we have a

position fixed in a vector and will make that position on or off, depending upon the appearance of the word in that vector or not.

(Refer Slide Time: 21:23)



### Definition of IR Model


An IR model is a quadruple  
 $[D, Q, F, R(q_i, d_j)]$   
Where,  
 $D$ : documents  
 $Q$ : Queries  
 $F$ : Framework for modeling document, query and their relationships  
 $R(.,.)$ : Ranking function returning a real no. expressing the relevance of  $d_j$  with  $q_i$

So, this is the way, the vectors are created and what we do having obtained those vectors is the point under discussion now and we have to discuss what is called the IR model. And IR model is formally defined as a quadruple which has 4 components  $D$  is the set of documents,  $Q$  is the set of queries,  $F$  is the frame work for modeling document query and their relationship. So,  $F$  is a frame work which represents the document and the queries and  $R(q_i, d_j)$  is a ranking function, it returns a real number expressing the relevance of  $d_j$  with  $q_i$  ok. So,  $R(q_i, d_j)$  captures the relevance of  $q_i$  with the document query and the document ok. So, this 4 tuple let us remember the set of documents, the set of queries the frame work for modeling the document and the queries and their relationship and  $R$  is the ranking function.

(Refer Slide Time: 22:32)

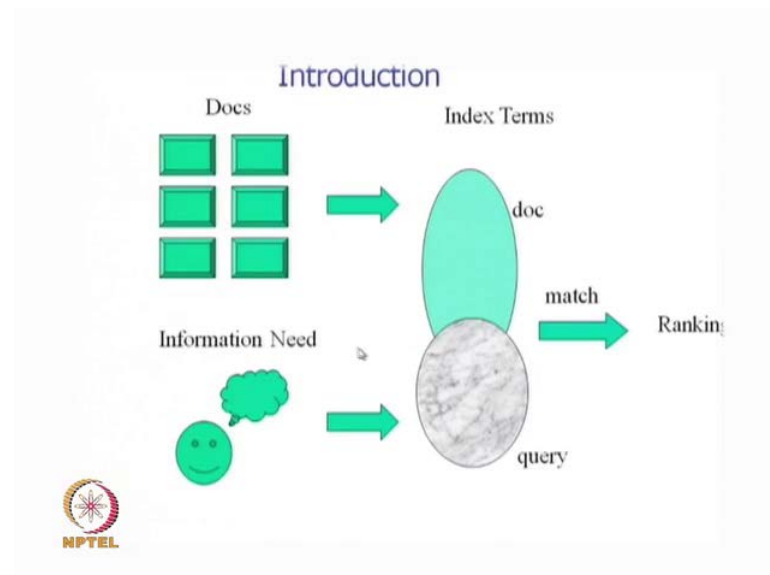
## Index Terms

- Keywords representing a document
- Semantics of the word helps remember the main theme of the document
- Generally nouns
- Assign numerical weights to index terms to indicate their importance



The index terms are created from the documents; these are the key words which represent a document. Semantics of the word helps remember the main theme of the document. And generally the index terms are nouns, the preposition conjunction articles etcetera, being present in all documents get dropped, which is why they become top wards. And we assign numerical weights to the index terms to indicate their importance.

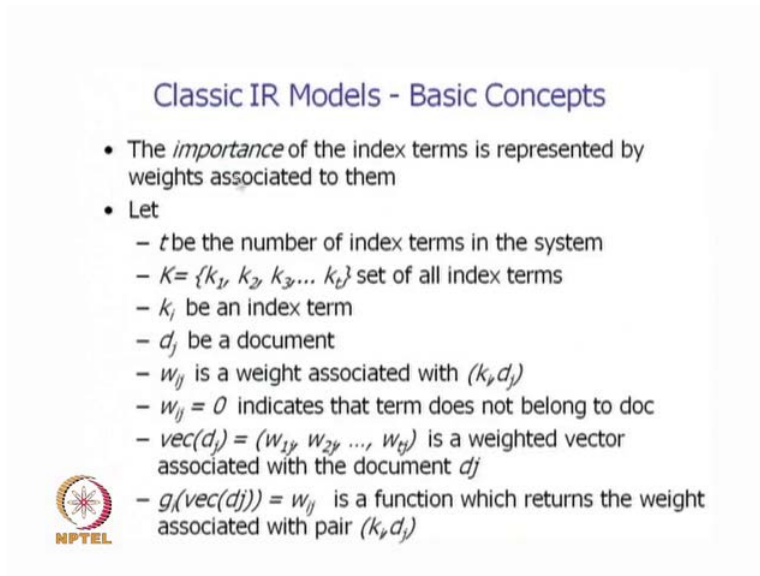
(Refer Slide Time: 23:09)



So, here is a semantic of what is being said here, near the documents. The index terms represent the documents, the query which is an expression of information need of the


user is matched with these documents and a ranking is produced depending on the relevance degree.

(Refer Slide Time: 23:34)



Classic IR Models - Basic Concepts

- The *importance* of the index terms is represented by weights associated to them
- Let
  - $t$  be the number of index terms in the system
  - $K = \{k_1, k_2, k_3, \dots, k_t\}$  set of all index terms
  - $k_i$  be an index term
  - $d_j$  be a document
  - $w_{ij}$  is a weight associated with  $(k_i, d_j)$
  - $w_{ij} = 0$  indicates that term does not belong to doc
  - $vec(d_j) = (w_{1j}, w_{2j}, \dots, w_{tj})$  is a weighted vector associated with the document  $d_j$
  - $g_i(vec(d_j)) = w_{ij}$  is a function which returns the weight associated with pair  $(k_i, d_j)$




So, there are a number of information retrieval models and we discussed two of them here, the 3rd which is the probabilistic information retrieval model will be discussed after this. So, the importance of the index terms is represented by weights associated with them, let us suppose  $t$  is the number of index terms in the system, in the example for Delhi and Bombay discussed a while back  $t$  value was 11, there were 11 terms which were the keyword in the system. So,  $t$  is the number of index terms in the system  $K$  equal to  $k_1, k_2, k_3, \dots, k_t$  is the set of all index terms,  $k_i$  is a particular index term, the  $i$ -th index term,  $d_j$  is a document,  $w_{ij}$  is a weight associated with  $k_i, d_j$  ok.

So, that means what is the weight  $w_{ij}$  of  $k_i$  index term with respect to this document  $d_j$ .  $w_{ij}$  is equal to 0 and this indicates that term does not belong to a document,  $vec(d_j)$  is a vector of weights and they are fixed this way  $w_{1j}$  is the weight of the 1st keyword with respect to the  $j$  document,  $w_{2j}$  is the weight of the 2nd keyword with respect to the  $j$  document,  $w_{tj}$  is a weight of the  $t$ -th keywords which respect to this document  $d_j$ , it is a weighted vector associated with document  $d_j$ ,  $g_i(vec(d_j))$  so, when the  $vec(d_j)$  is given then  $g_i$  is a function which can extract the weight associated with pair  $k_i, d_j$  ok. So, it returns the weight that is indicative of the importance of the keyword  $k_i$  for the document  $d_j$ .

(Refer Slide Time: 25:53)

### The Boolean Model

- Simple model based on set theory
- Only *AND*, *OR* and *NOT* are used
- Queries specified as boolean expressions
  - precise semantics
  - neat formalism
  - $q = k_a \wedge (k_b \vee \neg k_c)$
- Terms are either present or absent. Thus,  $w_j \in \{0,1\}$
- Consider
  - $q = k_a \wedge (k_b \vee \neg k_c)$
  - $vec(q_{dnf}) = (1,1,1) \vee (1,1,0) \vee (1,0,0)$
  - $vec(q_{cc}) = (1,1,0)$  is a conjunctive component



Proceeding further, we 1st take up the Boolean model, which is the simplest model and is let us say the basis for a subsequent model, it is a simple model based on set theory, only and or and not Boolean operations are used. The queries are specified as Boolean expression they have a precise semantics in terms absences or presence in the document and the formalism is neat. So, suppose we have this query k a and k b or not k c so, the semantics of these is that give me a document which contains the word contains the keyword k a and either k b is present, either keyword k b is present or the document does not contain the keyword k c.

So, it is a document which contains k and also satisfied the requirement that either k b is present, or k c is not present. So, one could think of a query a based on this particular expression so, give me any document which contains the word Delhi and it contains the word Kutubminar and either contains the word Kutubminar, or does not contain a word let say parliament ok. So, what kind of a query, would that be can a user be expected to form is query this way so that is the main criticism against the Boolean model. The semantic is clear but, it is not natural enough for a user to form this kind of query.

So, term are either present or absent, thus  $W_{ij}$  belongs to the set 0,1 so,  $W_{ij}$  value can be 0 or 1. Now, from the query k a or k and k b or not k c, we can form a document vector in disjunctive normal form, we call it vector of q d n f now, k a, k b, k c they occupy 3 position of a tople so, the 1st position is for k the 1st keyword, the 2nd

position for k b the 2nd keyword 3rd position for k c the 3rd keyword. So, one could expand this query and put it in a disjunctive form so, this would be k a and k b or k a and not k c.

(Refer Slide Time: 29:13)

$$Q = k_a \wedge (k_b \vee \sim k_c)$$

$$\begin{pmatrix} \uparrow & \uparrow & \uparrow \\ k_a & k_b & k_c \end{pmatrix}$$

$$Q = (k_a \wedge k_b) \vee (k_a \wedge \sim k_c)$$

$$\begin{matrix} \swarrow & \searrow & \swarrow & \searrow \\ (1, 1, 1) & (1, 1, 0) & (1, 0, 0) & \end{matrix}$$

So, we write it down to see how vectors are produced so Q is k a and k b or not k c so, we have a tople with 3 position so, is the position of k a is the position of k b, is the position of k c now, q can be written as q can be written as k a and k b or k a and not k c so, this gives rise to a tople 1, 1, 1 or 1, 1, 0 since, k c is not absent here, we can say this is the do not care element and that can be either 1 or 0 so, this is equivalent to k a and k b and k c or k a and k b and not k c so, k and k b and k c k a and k b and not k c and similarly, this will be either k a k b naught k c which is expressed by these, or it is k a not k b naught k c so, these 3 vectors are obtained from the query k a and k b or not k c.

So, now we (Refer Slide Time: 25:53) look at this slide again and we have the q d n f vector form, in. This way 1, 1, 1 or 1, 1, 0 or 1, 0, 0 and a conjunctive component of Vec q d n f is one of the elements, which is linked by or with other elements so, 1, 1, 0 is a conjunctive component.



(Refer Slide Time: 31:06)

**The Boolean Model**

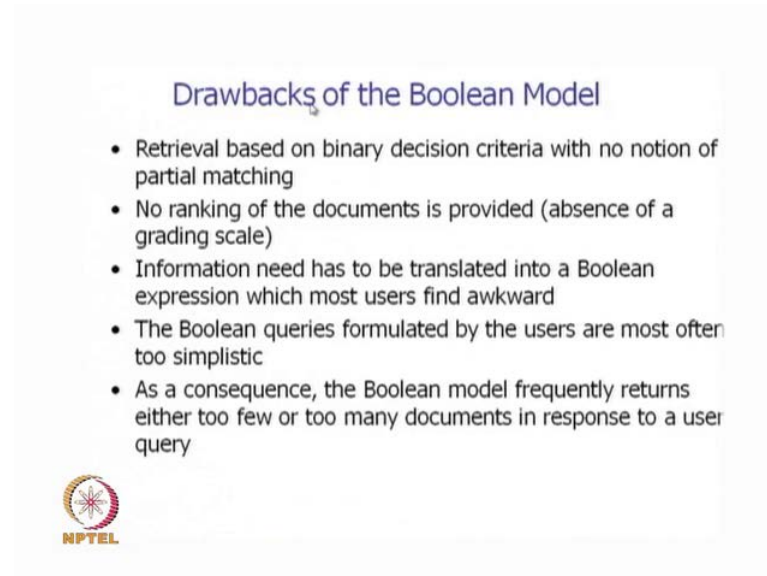
- $q = k_a \wedge (k_b \vee \neg k_c)$

- $sim(q, d_j) = 1$  if  $\exists vec(q_{cc}) \mid$   
 $(vec(q_{cc}) \in vec(q_{dnf})) \wedge$   
 $(\forall k_i, g_k(vec(d_j)) = g_k(vec(q_{cc})))$   
 $0$  otherwise

Proceeding further now, how is the relevance found out of a document so, q is k and k b or not k c so, here we have these 3 spears with respect to Ka K b and K c. Now, this particular area belongs to both k a belongs to all of K a K b and K c so, 1, 1, 1, is the representation of these intersection area, 1, 1, 0 this particular area and that is the intersection of k a and k b and k c being absent, this area belongs to only k a and this is 1, 0, 0. So, similarity of a query with document d j is asserted to be 1, if their exit conjunctive component from the vector of q such that, the conjunctive components belongs to the d n f represent of the query vector and for every component of a the conjunctive vector, we have a match with the document vector representation.


So, that this particular term essentially means that, the component of the conjunctive component of the query vector should each match the component of the document vector, that means whatever, is present in the query should be present in the document whatever, absent in the query should also be absent in the document so, these the essential meaning of the similarity between query and the document. So, this is quite easy to compute.

(Refer Slide Time: 32:58)



**Drawbacks of the Boolean Model**

- Retrieval based on binary decision criteria with no notion of partial matching
- No ranking of the documents is provided (absence of a grading scale)
- Information need has to be translated into a Boolean expression which most users find awkward
- The Boolean queries formulated by the users are most often too simplistic
- As a consequence, the Boolean model frequently returns either too few or too many documents in response to a user query

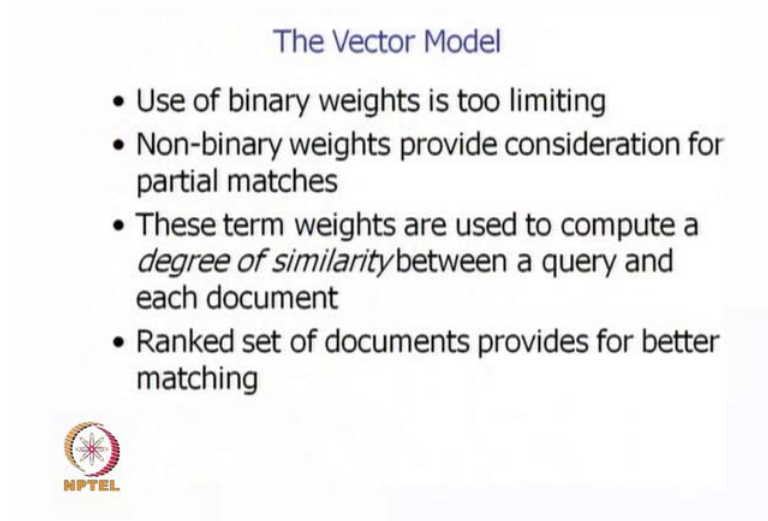


However, there are a number of drawbacks of the Boolean model. So, we have said already retrieval is based on binary decision criteria with no notion of partial matching, the Boolean IR model, the Boolean model matching an relevance insist on for complete matching each term must be either present or absent, no ranking of the document is provided so, there is no graded scale with respect to the ranking that means ranking of the documents are either similar or the similar to the query. Because whatever, present in the query must be present in the document whatever, is absent must be absent in the document.

So, this is another drawback, information need has to be translated into a Boolean expression, which it is very difficult to do for most users so, how can you say give me a document which has Delhi and either contains the word Kutubminar or it does not have the word parliament so, these kind of queries are never ask by the user, the user will simply put a query Kutubminar in Delhi user to not even specify notion. So, this kind of query representation in term of Boolean is unnatural, the Boolean query formulated by the user are most often too simplistic so, this is in term of present or absent, the query and the thinking also there by becomes constraint and simplistic and as a consequence the Boolean model frequently returns either too few or too many document in response to a user query.


So, these are the drawbacks and we would like to mention that Boolean I R model is important but, the I R model does not really prove to effective model. However, it capture this important notions of keywords being present or absent and there by deciding the important of the document, which is a good idea of this a foundational notion.

(Refer Slide Time: 35:25)



The Vector Model

- Use of binary weights is too limiting
- Non-binary weights provide consideration for partial matches
- These term weights are used to compute a *degree of similarity* between a query and each document
- Ranked set of documents provides for better matching




So, we proceed and go to the vector model, which is the pre dominate field in I R today. And it is also based on a pretty a simple idea, what it says is that the use of binary weights is too limiting so, if you just say 1 or 0, then this hardly is a realistic representation for the document, non binary weights provide consideration for partial matches these term weights are used to compute a degree of similarity between a query and each document. So, these term are used to compute a degree similarity between a query and each document. And ranked set of documents provides for better matching. So, there is a notion of ranking when we retrieve document not all document satisfy the information need of the user equally ok. And therefore, it is important to know the degree of the relevance of the documents.

(Refer Slide Time: 36:38)

**The Vector Model**

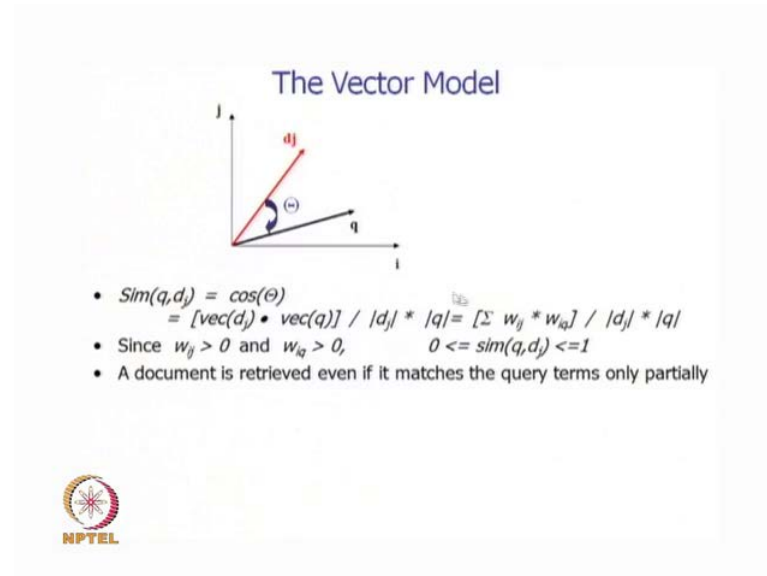
- Define:
  - $w_{ij} > 0$  whenever  $k_i \in d_j$
  - $w_{iq} \geq 0$  associated with the pair  $(k_i, q)$
  - $vec(d_j) = (w_{1j} \ w_{2j} \ \dots \ w_{tj})$
  - $vec(q) = (w_{1q} \ w_{2q} \ \dots \ w_{tq})$
- In this space, queries and documents are represented as weighted vectors



So, we now proceed to define, the vector model and in this we 1st say that this way  $w_{ij}$  is greater than 0 whenever, a key term in the query is present in the document.  $w_{iq}$  greater than or equal to 0 is associated with the pair  $i$ -th keyword and the document  $q$ . The vector of  $d_j$  is formed by the weights, weight values,  $w_{1j}$   $w_{2j}$  up to  $w_{tj}$ , corresponding to the  $t$  keywords  $w_1, w_2, w_t$  ok. And these are the weight values of this  $t$  keywords with respect to the document  $d_j$ , they indicates the important of the 1st 2nd 3rd up to the  $t$ -th keyword for the document.

Similarly, vector  $q$  is obtained from the weight values of this  $t$  keywords and seeing their weights with respect to the query  $q$ . Now, queries are typically very small, documents are larger in length so, this weight values have to be obtained in a judicial way, query is small the document is large. So, in this space, in the vector model framework queries and documents are represented as weighted vectors.

(Refer Slide Time: 38:22)




Now, if we have the vector model and we got the query vector and the document vector, then the similarity, between these 2 vectors  $Sim(q, d_j)$  is obtained by a COS similarity measure. So, this COS similarity measure is nothing but, the COS of the angle between these two vectors, which is obtained as the dot product of these 2 vectors, divided by the product of the magnitude of these 2 vectors. And these come out to be equal to the multiplication at the corresponding position of the weights values of these 2 vectors and divided by their modules.

Now, COS of any angle is between -1 and plus 1 and in this particular case we never get any negative values because, the weight values are always greater than 0. So, the similarity value will come out to be between 0 and 1, these are non-negative, a document is retrieved even if it matches the query terms only partially ok. So, retrieval is done anywhere by the search engine and it is a rank calculation on the relevance calculation, where COS similarity measure is used.

(Refer Slide Time: 39:55)

**The Vector Model**

- $Sim(q, d_j) = [\sum w_{ij} * w_{iq}] / |d_j| * |q|$
- How to compute the weights  $w_{ij}$  and  $w_{iq}$ ?
- A good weight must take into account two effects:
  - quantification of intra-document contents (similarity)
    - *tf* factor, the *term frequency* within a document
  - quantification of inter-documents separation (dissimilarity)
    - *idf* factor, the *inverse document frequency*

  $w_{ij} = tf(i, j) * idf(i)$


So, proceeding with more details of the vector model, we get similarity of  $q$  and  $d_j$  as the dot product of the 2 vectors  $q$  and  $d_j$  divided by the modules of these 2 magnitude of these 2 vectors and the product. Now, the key question that comes to our mind is how to compute? The weights  $w_{ij}$  and  $w_{iq}$  how do we do it? A good weight must take into account 2 effects, quantification of intra document contents the similarity, which is also called the *tf* factors, the term frequency within a document. And quantification of intra documents separation, this is the dissimilarity and this is captured by what is called the *idf* factor the inverse document frequency.

So, a good weights must take into account this 2 effects quantification of intra document contents, quantification of intra documents separation, the *tf* factor and the *idf* factor. The weight of  $w_{ij}$  of a query term of a keyword  $i$  with respect to a document  $d_j$  is given as  $tf_{ij}$  into  $idf_i$ . So, the inverse document frequency of the  $i$ -th keyword multiplied by the term frequency of  $i$  in the  $j$ -th document so, these are very simple measure very fundamental measure in information retrieval and this is used or some variation of this idea is used for ranking.

(Refer Slide Time: 41:40)

**The Vector Model**

- Let,
  - $N$  be the total number of docs in the collection
  - $n_i$  be the number of docs which contain  $k_i$
  - $freq(i,j)$  raw frequency of  $k_i$  within  $d_j$
- A normalized *tf* factor is given by
  - $f(i,j) = freq(i,j) / \max(freq(i,j))$
  - where the maximum is computed over all terms which occur within the document  $d_j$
- The *idf* factor is computed as
  - $idf(i) = \log(N/n_i)$
  - the *log* is used to make the values of *tf* and *idf* comparable. It can also be interpreted as the *amount of information* associated with the term  $k_i$ .



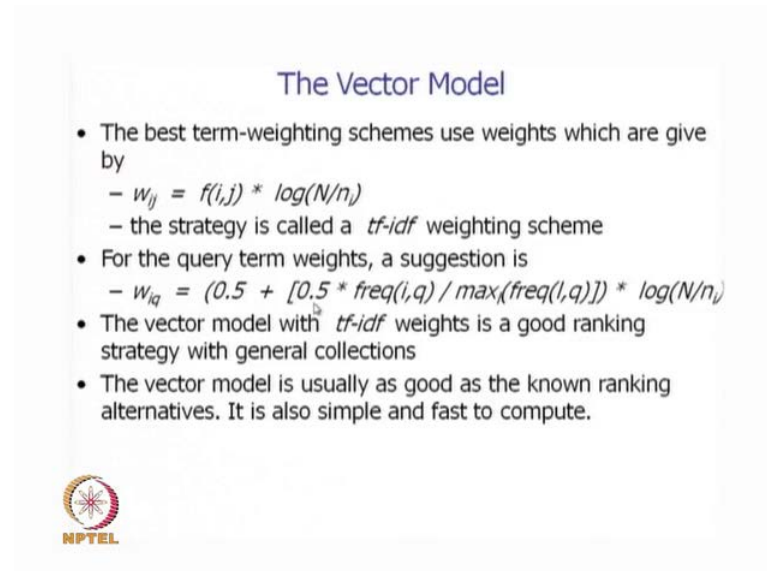
So, formulating this further, suppose  $N$  is a total number of document in the collection,  $n_i$  is the number of document which contain  $k_i$  and frequency  $i, j$  is the raw frequency of  $k_i$  within  $d_j$  so, these the term frequency of  $k_i$  in the document  $d_j$  in a normalized *tf* factor is given as  $f(i, j)$  which is nothing but, frequency of  $i$  in  $j$ , that is the term frequency of the  $i$ -th keyword divided by maximum over all words frequency present in the document. So, that means we take every words from this particular document and measure the frequency, pick up the maximum frequency amongst all this use these to divide the term frequency of the  $i$ -th keyword ok.

So, this maximum is computed over all terms which occur in the document  $d_j$ . The *idf* factor is computed by a little more complicated formula  $idf(i)$  is  $\log(N/n_i)$  so,  $N/n_i$  is a number, which is total number of document in the system divided by the number of document which contain the keyword  $k_i$ . So,  $N/n_i$  is greater than or equal to 1 so, when this fellow is one that means if a keywords is present in all documents, then taking the log off this number  $\log 1$  will be equal to 0 so, the *idf* of a keyword which is of a word which is present in all document is 0, that means this particular word has no distinct characteristic to give weight age to a particular document.

So, this log is use to make the value of *tf* and *idf* comparable, it can also be interpreted as the amount of information associated with the term  $k_i$  so,  $\log N/n_i$ ,  $n_i$  by  $N$  is like something like a probability measure for the keyword for the present of the keyword and


$N/n_i$  is inverse of that probability so, negative log of the probability will be something like an information content associated with the term  $k_i$ .

(Refer Slide Time: 44:20)



The Vector Model

- The best term-weighting schemes use weights which are given by
  - $w_{ij} = f(i,j) * \log(N/n_i)$
  - the strategy is called a *tf-idf* weighting scheme
- For the query term weights, a suggestion is
  - $w_{iq} = (0.5 + [0.5 * \text{freq}(i,q) / \max(\text{freq}(l,q))]) * \log(N/n_i)$
- The vector model with *tf-idf* weights is a good ranking strategy with general collections
- The vector model is usually as good as the known ranking alternatives. It is also simple and fast to compute.



The best term weighting schemes use weights, which are given by  $w_{ij}$  equal to term frequency into inverses document frequency  $f_{ij}$  into log of  $N/n_i$ , this strategy is called *tf-idf* weighting schemes. For the query term weights a suggestion is that  $w_{iq}$  is computed in this way. So, why the difference here the differences because, the query typically contains a very small number of terms and therefore, the weights have to be found out in a more tricky way so, this formula used  $\log(N/n_i)$  is nothing but, the *idf* term, this multiplies a particular term which is 0.5 plus 0.5 into the term frequency.


So, 0.5 addition of 0.5 all this are done to compensate for the fact that query is short in size, a document is much longer and this kind of addition multiplication etcetera, facilitate coming up with good weight values. So, this formula difficult to explain it was arrived at by Trier and error, which is like an experimental understanding of the information retrieval this searcher or engineers. The vector model with *tf-idf* weight is a good ranking strategy with general collections, the vector model is usually as good as the known ranking alternatives it is also simple and fast to compute.



(Refer Slide Time: 46:11)

### The Vector Model

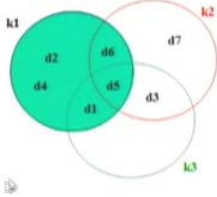
- Advantages:
  - term-weighting improves quality of the answer set
  - partial matching allows retrieval of docs that approximate the query conditions
  - cosine ranking formula sorts documents according to degree of similarity to the query
- Disadvantages:
  - assumes independence of index terms; not clear that this is bad though




The advantages of the vector model are term weighting improves quality of the answer set, partial matching is allowed and that leads to retrieval of document that approximate the query conditions, cosine ranking formula sorts the documents according to the degree of similarity to the query. The disadvantages are that it assume independence of the index terms so, this kind of independence it is not clear that it is bad, the independence is often an effect a idea.

(Refer Slide Time: 46:52)

### The Vector Model: Example I



	k1	k2	k3	q • dj
d1	1	0	1	2
d2	1	0	0	1
d3	0	1	1	2
d4	1	0	0	1
d5	1	1	1	3
d6	1	1	0	2
d7	0	1	0	1
q	1	1	1	



So, here is an example, which we shown in terms of the keywords  $k_1$ ,  $k_2$  and  $k_3$  and the document which contain them for example, this blue circle shows that, the term  $k_1$  is present in  $d_2$ ,  $d_4$ ,  $d_6$ ,  $d_5$  and  $d_1$ . If I look at the whole situation from the point of document, then we a can represent the document in term of their keywords. So,  $d_1$  has 1, 0, 0 that means  $d_1$  contains only 1, 0, 1 so,  $d_1$  is in the intersection of  $k_1$  and  $k_3$  so, it contains  $k_1$ , 1 here  $k_3$ , 1 here it does not contain  $k_2$  and therefore,  $k_2$  position is 0. Similarly, all this document is represent in terms of the keywords, will discuss this example in the next lecture. And also move on to other ranking models.