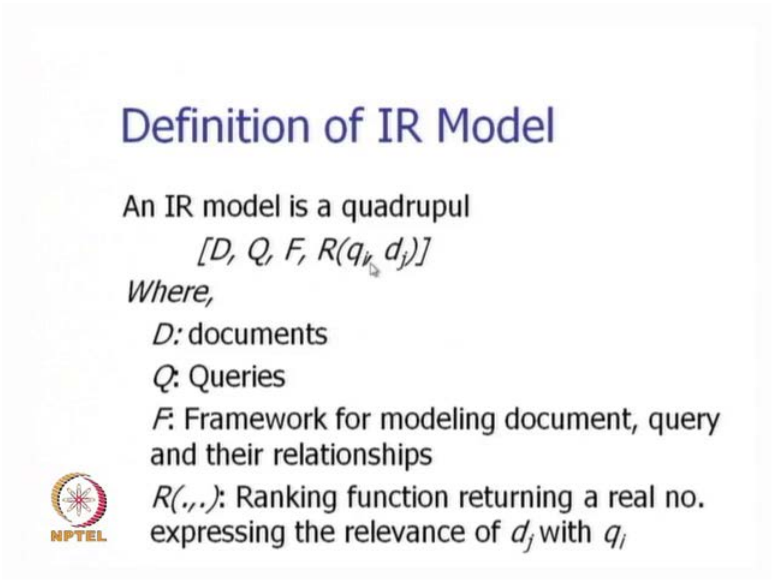
Natural Language Processing
Prof. Pushpak Bhattacharya
Department of Computer Science& Engineering,
Indian institution Technology, Bombay

Lecture - 25
IR Models: NLP and IR Relationship

Today, we will finish the discussion on I R models which captures the way the information need is satisfied for the user by the web or the information repository. We will also explore the relationship between N L P and I R, after all we are discussing information rakehell in the context of natural language processing. Exploring the question of how N L P and I R interact with each other? How the two fields can benefit from each other's strengths? So, we continue first of all with the question of I R model where, the main question is how the information of the information repository; internet being special case of this repository is represented to the user to satisfied his or her information need.

(Refer Slide Time: 01:21)



Definition of IR Model

An IR model is a quadrupul

$[D, Q, F, R(q_i, d_j)]$

Where,

D: documents

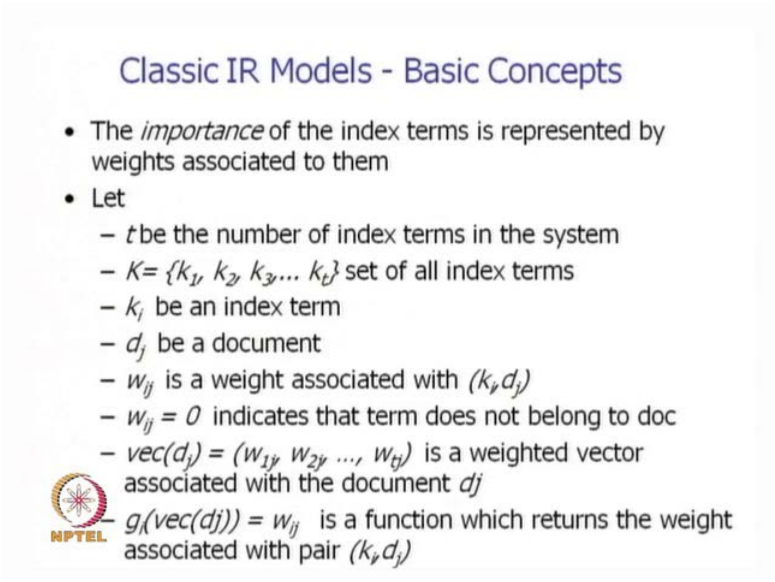
Q: Queries

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, we continue with presentation we have seen this before that an information retrieve model is define as a quadrupul where, the 4 components are D Q F and R. R being a function of q i and d j where, q i is the i-th query, d j is the j-th document. Now, D the 1st component of the trample is the set of documents, Q is the set of queries, F is the frame for modeling the document query and the relationship. So for example, ah we will the document and query be represented as you know vectors of numbers, so that question comes under the preview of F. R is the ranking function which returns a real number expressing the relevance of the document d j; with the query q i.

(Refer Slide Time: 02:32)



Classic IR Models - Basic Concepts

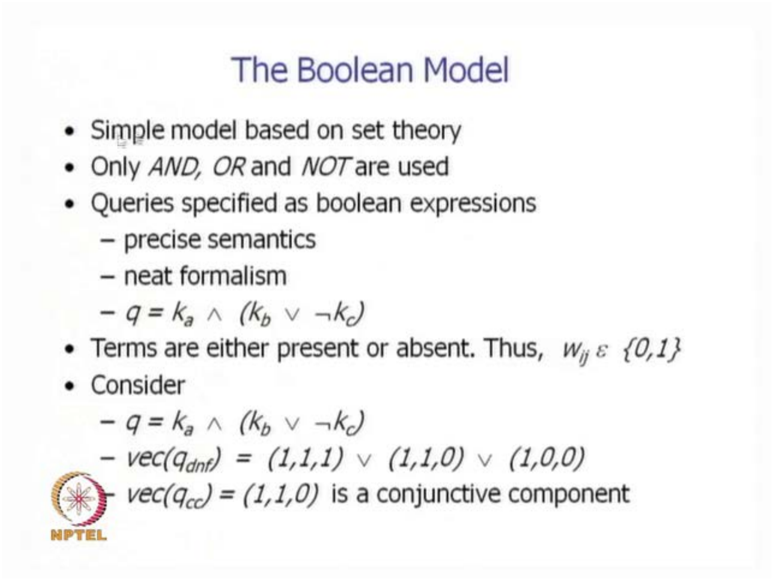
- The *importance* of the index terms is represented by weights associated to them
- Let
  - t be the number of index terms in the system
  - $K = \{k_1, k_2, k_3, \dots k_t\}$ set of all index terms
  - k_i be an index term
  - d_j be a document
  - w_{ij} is a weight associated with (k_i, d_j)
  - $w_{ij} = 0$ 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 dj
  - $g_i(vec(dj)) = w_{ij}$ is a function which returns the weight associated with pair (k_i, d_j)

So, this was discuss before and we also then talked about a how the importance of the index terms is represented by weights associated with them. So suppose, t is the number of index terms in the system which, comes from the vocabulary of the corpuses being searched. So, suppose there are you know d documents in the corpse, we collect all the important terms in these d documents and suppose those number of terms be t. Then k the set of index terms k 1, k 2, k 3 up to k t is providing the index terms; k i is a particular index term d j is a particular document d j, w i j is a weight associated with k i and d j.

So, k i, d j this tupple is a combination which shows the importance of the q what k i in the document d j, w i j equal to 0 indicates that the term does not belong to the document. So, in particular if we stick to a bullion situation where, the weight value is 1 or 0 expressing presents or absence of q word in the document, then, this a particular way of assigning weights to the terms. This is the 1st starting point of presenting the weight of terms in the information retrieval scenario. W i j equal to 0 indicates that the terms does not belong to the document vec d j is nothing but, a vector of weight values of the keywords k 1 to k t with respect to the document d j.

So, w 1 j, w 2 j up to w t j are the weights of the keywords k 1 up to k t in the document d j, that why this double lower suffix 1 j, 2 j and so on. g i vec d j is equal to w i j, this is a function which returns the weight associated with a pair k i d j. So this is the function which extracts the weight of the keyword i with respect to the document j, so this is a precise formulation of the classical information retrieval model. Where, the world consist of the keywords, the document and the weightages of this keywords with respect to the document.

(Refer Slide Time: 05:48)



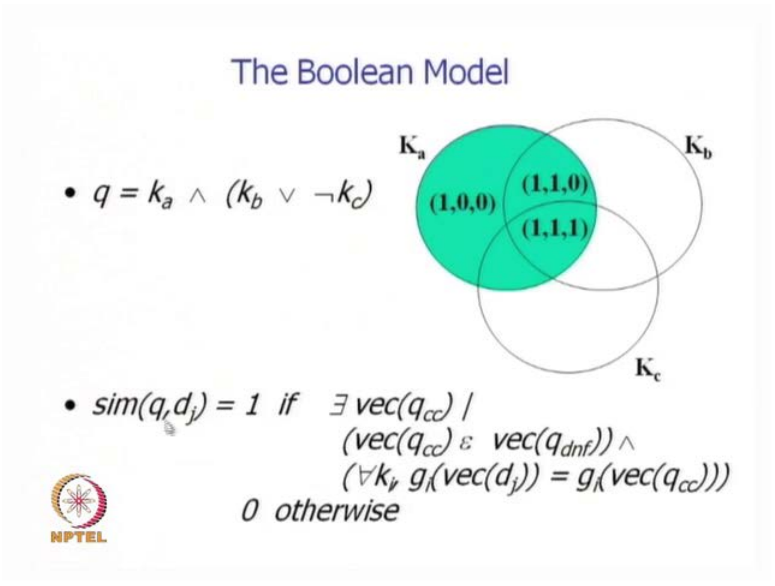
The Boolean Model

- 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, $w_{ij} \varepsilon \{0,1\}$
- 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 now go into different models of information retrieval. The 1st model is the Boolean model, we touched upon this last time; let is explain this more clearly. This is the simple model based on set theory and it allows only and or a not operations, so queries are specified as Boolean expressions: they have a very precious semantic and the formalism is need. So, what do you mean by this? The semantic is precise means that, the meaning of the query is well understood in terms of presence or absence. So, if we say the query is t 1 and t 2 and not of t 3 suppose, we say this and in that case the meaning is very clear.

The meaning is that give me a document which contains the keyword k 1, contains the keyword k 2 and does not contain the word k 3 alright. So, this is the meaning of the statement here that, the Boolean model gives rise to precise semantics. The formulism is need to we simply have the AND, OR and NOT operators and the proposition so to say or the presence or absence of the keywords. Thus, W i j is either 0 or 1 now, if I take this query q which is k a and k b or not k c, so this was a discussed last time we can convert this query into a this junk it normal form. So, this hole expression equivalent k a and k b and k c or k and k b and not k c or k and not k b and not k c; so, this can be very easily see from the query expression k a and k b or not k c. So 1 1 0 is a conjunctive component in the d n f expression for the query.

(Refer Slide Time: 08:31)



The Boolean Model

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

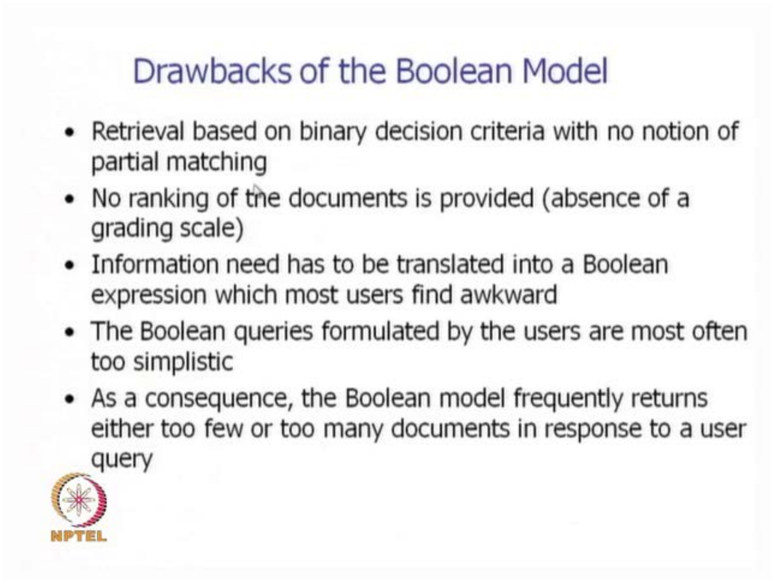
- $sim(q,d_j) = 1$ if $\exists\, vec(q_{cc})\ |$
  $(vec(q_{cc}) \varepsilon\ vec(q_{dnf})) \wedge$
  $(\forall k_i,\ g_i(vec(d_j)) = g_i(vec(q_{cc})))$
  0 otherwise

Now, we can picturise to the query as follows: so it is as if k a the 1st key words has reason of influence shown, as the blue circle k b has a region of influence and k c again has a reason of influence. So, this area where this reason is influence intersect indicates the presence of all k a, k b and k c; all the 3 are presents, so that is why this conjunctive component 1 1 1 belongs to this intersection area. This is the area which contains k a and k b but, not k c similarly, this is the area which contains k a but, not k b and k c. So, this set venn gram kind of picture expresses the conjunctive components very well.

Now we say that the similarity, of a query q and d j is 1 if a particular query q has similarity of 1 with d j, if there exist a conjunctive component of the query such that this conjunctive component belongs to the d n f expression for the query and, we weight component of the vector for the document matches the weight component of the query vector. That means, the each component of the query vector each component of a conjunctive component of query vector matches the components of the document vector; this is the meaning.

So, simply put this means that the presence or absence of keywords in the query should be matching the presence of absence of the keywords in the document d j then we say that the similarities is 1. Or more simply put this only means that the keywords which are present or absent in the query are also present or absent in the document. Then, that particular document has similarity value of 1 with respect to this particular query.

(Refer Slide Time: 11:16)



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

Now the draw backs of the Boolean model are the following: retrieval is based on binary decision criteria with no notion of partial matching, so all keywords which are present or absent in the query also have to present or absent in the document. This is a black and white situation users information need is never this way, user information need is often fuzzy and the user is satisfied with fuzzy information. No ranking of the documents is provided because, the score is simply as similarity value which is either 0 or 1, all documents which match the query term presents or absents have value either 1 or 0.

There is no relative ranking between the documents, so this is also not a realistic situation because, we know documents supply the information need of the user in different degree. Some documents are more relevant other are less relevant so, by its very nature the problem is fuzzy; it is not a black and white situation, it is not a 0 1 situation. Information needs has to translated, into a Boolean expression which most users find awkward. So, one can immediately see that in the Boolean model of information retrieval; Boolean modal information retrieval there is an artificiality ok. What the system is doing is that, it is simply matching the keyword presence or absents in the query, with the key words presence or absence in the document.

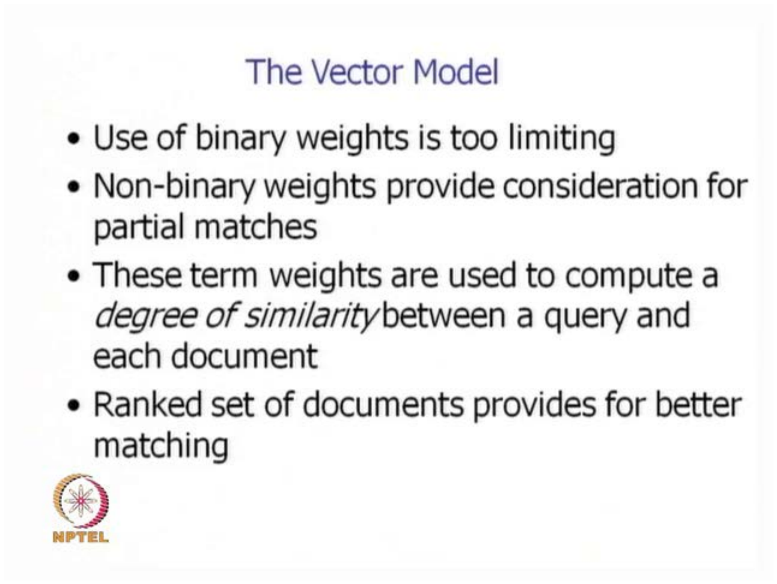
Now, users have information need when the information need arises in the mind it is never in the form of presence or absence of keywords in a document. It is hardly the case, that the user is a trying to express his information in terms of presence or absence. The user is not saying that give me a document which contains the term Delhi but, not the term Qutuabminar ok. This is never the case, it is very artificial the user might say, give me all those information about Delhi which does not mention a Qutubminar or which is independent of qutubminer. Give me all those facts about delhi but, do not tell me anything about qutubminer.

So, users information need is expressed that way so, user is not thinking of a particular document user he is thinking about information. So this information has to come from the information repository. So therefore, we are saying that the Boolean expression formulation is found to be quite awkward by user. The Boolean (Refer Time 14:34) queries are formulated by the users and many times they are too simplistic; this is because we have said that users information needs is much more complex than simple term presence or absence. Many times it is very fuzzy, many times it is imprecise full of noise, so Boolean model being a 0 1 world, cannot capture this situation.

So, as a consequence the Boolean model frequently returns either too few or too many documents in response to a user query. So, this is the fact that, the Boolean model because of its insistence on strong 0 1 matching would either written too few or too many documents. So, all those disadvantage of course, exist for the Boolean model but, it has its own advantage also. First of all there is doubt about the fact that, information need begins or starts getting satisfied from the notion of presence or absence of keywords. So presents or absents keywords is very important.

It is true, that if a term is present in a document, the document does may not necessarily a part in to that particular term. Similarly, if a term is absent from the document it may not necessarily be the fact that, the document does not say anything about that term ok, both the possibility exist. But, still Boolean model of information retrieval provides a basic platform for information retrieval; it provides a sanity check for the search engine which is being the developed. The searching engine should at least capture the situations where the terms are presents or absent in the document. It should be able to; it should have at least that capability, so it is like a sanity check.

(Refer Slide Time: 16:47)



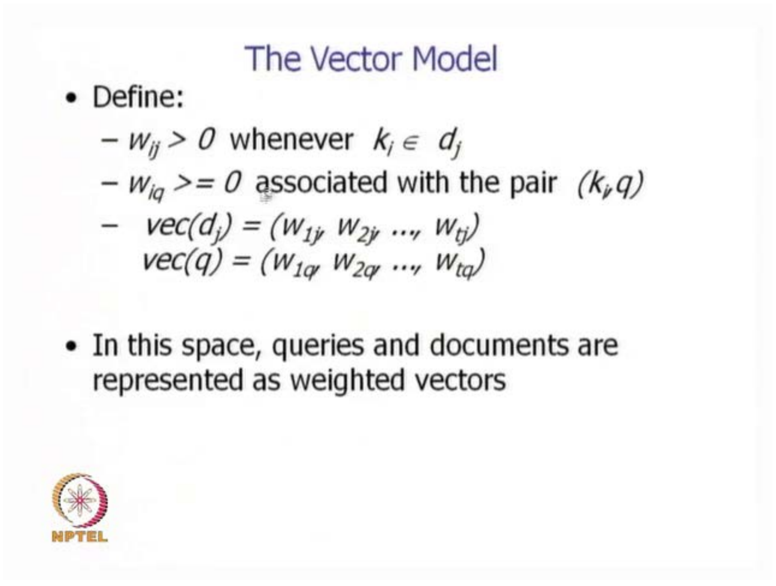
The Vector Model

- 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 move to the next model of information retrieval which is, probably the most popular model and a practical search engines do use this particular model known as the vector model. So, use of binary weights is too limiting non binary weights provide consideration for partial matches and these term weights are used to compute a degree of similarity between a query and documents. So, this is very important; degree of similarity is important because we have a human being making queries to the web, it is the information need of a human being which is being satisfied. And humans being by nature are fuzzy, noisy, imprecise.

Ranked set of documents provides for better matching, so here comes the notion of ranking which is fundamental to information retrieval. And, it is this particular fact the need to have very precious ranking whereby, relevant documents are captured that leads to the expression of can language insides can natural language processing be integrated with information retrieval. This is the genesis of studying, N L P is influence on I Rand I R influence on in N L P.

(Refer Slide Time: 18:21)

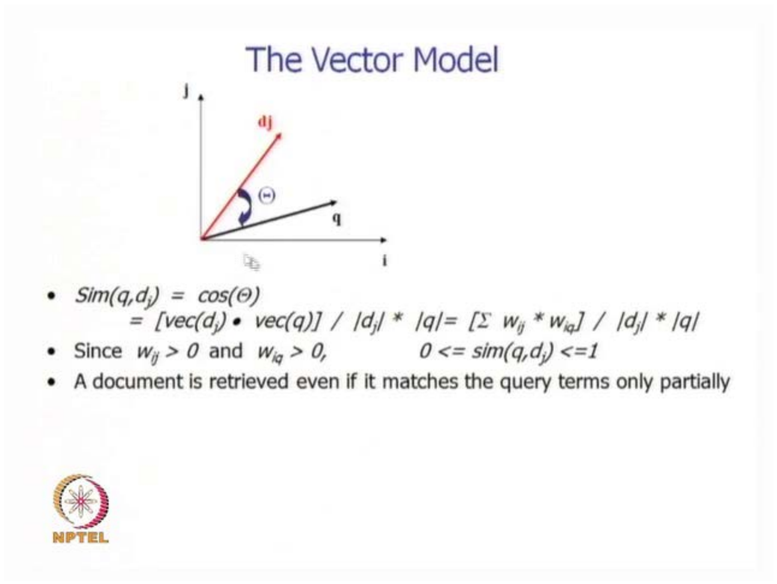


The Vector Model

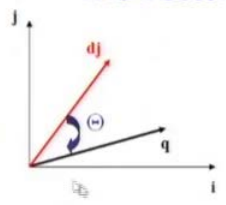
- Define:
  - $w_{ij} > 0$ whenever $k_i \in d_j$
  - $w_{iq} >= 0$ associated with the pair (k_i,q)
  - $vec(d_j) = (w_{1j}, w_{2j}, ..., w_{tj})$
    $vec(q) = (w_{1q}, w_{2q}, ..., w_{tq})$
- In this space, queries and documents are represented as weighted vectors

The vector model is as follows: we have already define the document query, the frame work and a relevance function gives us the information retrieval model. Now, the weights W i j is greater than 0 whenever, k i belongs to d j, W i q is greater than equal to 0 which is associated with the pair k i and q. Now, the vector representation of the document d j is w 1 j, w 2 j or up to w t j, so this is with respect to the keywords 1 to t for the document j. And similarly, the vector of q is W 1 q, W 2 q up to W t q, which is nothing but, the weightages of the keyword in the query. So, in this space, query and documents are represented as weighted vectors.

(Refer Slide Time: 19:24)



The Vector Model

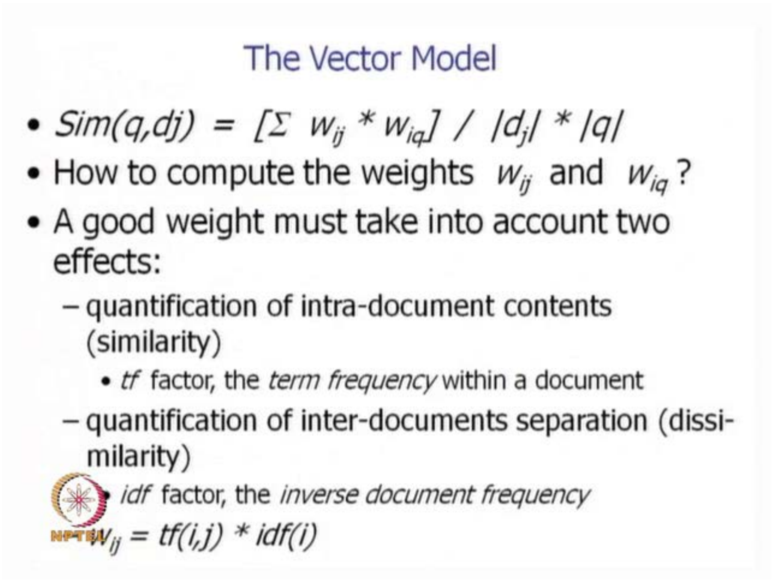


- $Sim(q,d_j) = cos(\Theta)$
  $= [vec(d_j) \bullet vec(q)] / |d_j| * |q| = [\Sigma \ w_{ij} * w_{iq}] / |d_j| * |q|$
- Since $w_{ij} > 0$ and $w_{iq} > 0$, $0 <= sim(q,d_j) <=1$
- A document is retrieved even if it matches the query terms only partially

So, the similarity major is typically the cosine similarity major, the similarity between the query q and a document d j is the cosine of the angle between the vector for q and vector for d j. So, this is computed as vector for d j dot vector for q; the dot product divided by the mod of d j and mod of q. Because, we know that the dot product is nothing but, the product of the magnitude of the vectors and cos of the angle between them. So this particular expression gives us the cos of the angle between the 2 vectors.

Now, the vector the dot product can be found from the some of the product of the weight values ok, with i varying from 1 to t where, t is the number of key words and it is divided by the modules of d j vector and q vector. So, similarity because of this formulation is between 0 and 1, a document is retrieved even it matches the query terms only partial. So, this is the crucial difference from the Boolean model of information retrieval, even if the query terms partially match we would retrieve the document and show it to the user. Of course, the rank could be lower than document which, match the information need exactly.

(Refer Slide Time: 21:06)

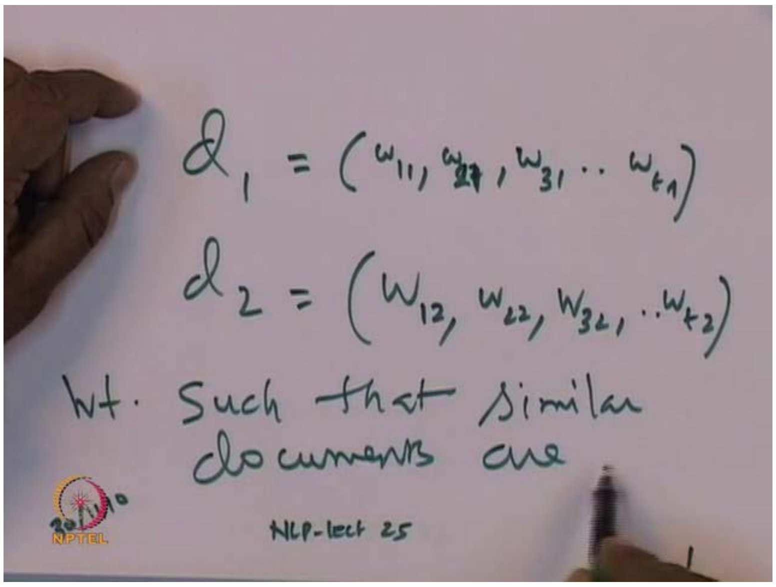


The Vector Model

- *Sim(q,dj) = [Σ 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 (dissi-milarity)
    - *idf* factor, the *inverse document frequency*
- *w_{ij} = tf(i,j) * idf(i)*

So, similarity is expressed as this product of weightage terms corresponding terms divided by modulus of d j and modulus of d q modulus of d j and q this product. Now the questions is how do we compute the weights W i j and W i q? So a good weightage must take into account these 2 effects: quantification of intra document contents, which is the similarity; so this is the t f factor the term frequency within a document and quantification of intra documents separation, this is the dissimilarity. And this is known as the i d f factor the inverse document frequency.

(Refer Slide Time: 22:03)

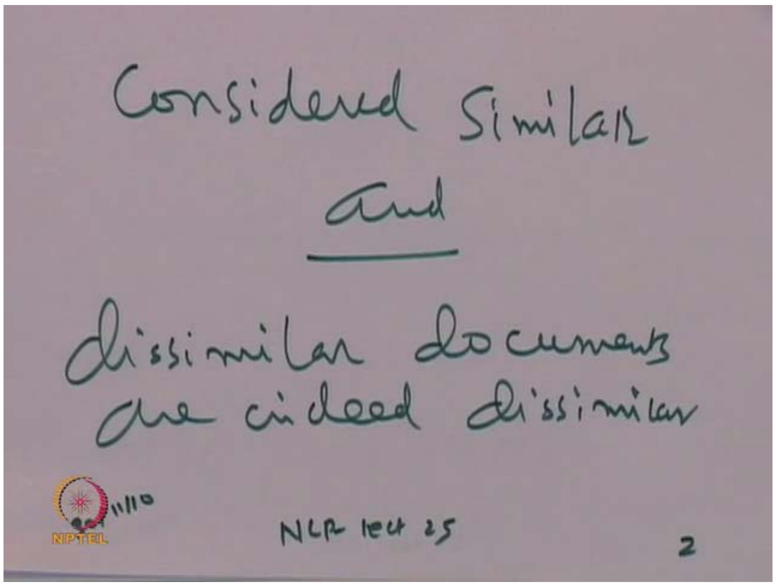


$d_1 = (w_{11}, w_{21}, w_{31} \cdot\cdot\, w_{t1})$

$d_2 = (W_{12}, w_{22}, W_{32}, \cdot\cdot W_{t2})$

Wt. such that similar documents are

NLP-lect 25



Considered similar

and

dissimilar documents are indeed dissimilar

NLP lect 25

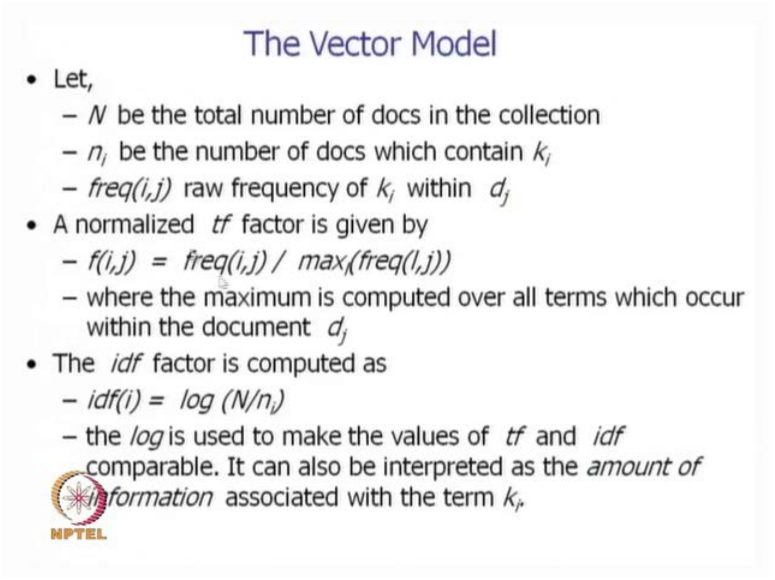
2

So, the what is being said is that, this is what being said, bit of writing will make it clear. So, suppose, we have 2 documents: d 1 and d 2; the vector for d 1 is w 1 1, w 1 2 w sorry w 2 1, the 2nd suffix is the document suffix w 3 1 up to w t 1 ok corresponding to t keywords. And for document 2 it is w 1 2, w 2 2, w 3 2 up to w t 2 ok, we have these 2 documents. Now the weightage should be such that, if the documents are similar to each other then the weigh should capture that similarity. And if the documents are distinct from each other there contents are very very different that also should be captured by this weighting scheme.

So you see, this is a very important point weight such that, similar documents are I will continue to the 2nd page; similar documents are considered similar and dissimilar documents are indeed dissimilar. So, this is the responsibility on the weightage scheme the weight values would be such that documents which are similar are close together documents which are dissimilar are far from each other. (Refer Slide Time: 21:06) So, the slide is express this factor and says that there is something called t f factor which, captures intra document contents similarity and there is this i d f factor which achieves intra documents separation.

And the weighted is equal to term frequency of i j into inverse document frequency of i. So, here t f i j means, the term frequency of keyword i in the j-th document. How many times the keywords i appear in the j-th document and i d f of the keyword i? This will be defined shortly.

(Refer Slide Time: 24:41)



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
  - $f(i,j) = freq(i,j) / max_l(freq(l,j))$
  - 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, here we explain these terms suppose n is the total number of documents in the collection and n i is the number of documents which contain k i ok. The particular keyword k i is contained in n i documents, freq i j is the raw frequency of q i within d j. So this is the term frequency of the keyword k i within d j it is equal to the number of times the keyword k i appears in the document d j. So a normalized t f factor is given as f i j equal to frequency of i j divided by max over l of frequency l j.

What does it mean? It means that take that term whose frequency is maximum in the j-th document take that frequency. Let that divide the frequency of the i-th keyword in the j-th document; so that would give a normalized t f factor of the i-th keyword in the j-th document. And this maximum is computed over all terms which occurs within the document d j. The i d f factor is computed as i d f i equal to log of N by n i so, n i we remember is the number of documents which contains the keyword k i and N is the total number of documents in the collection. So, N by n i is the inverse of the proportion of the documents which contain the keyword k i.

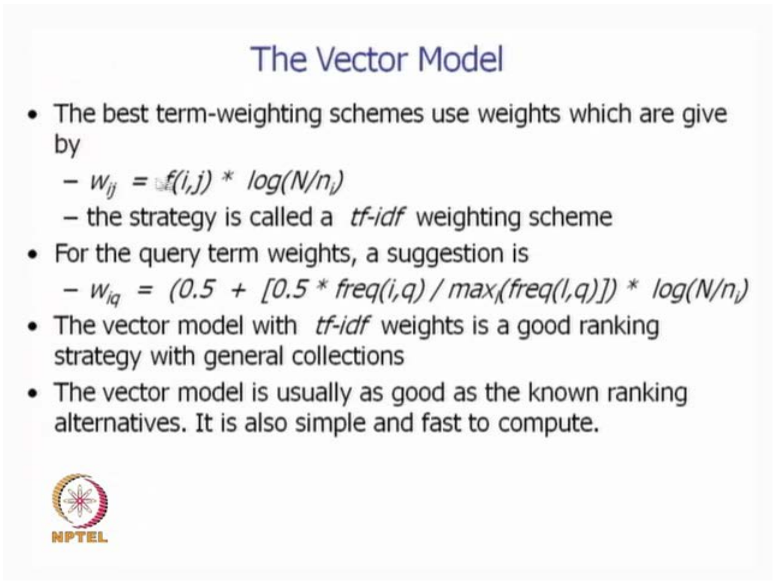
So log is the use to make the values of t f and i d f comparable, it also can interrupted as the amount of information associated with the term k i. So n i by N would be something like a probability so, this term is like a negative log of probability which is something like the entropy. And entropy is related to the information content of the document, so this is i d f fact. Now, you should verify that i d f is indeed a factor which this distinguishes one document from another because of a particular key word k i.

In particular look at the boundary conditions if n i equal to N that means, if a particular word or a particular keyword appears in all documents then this, by virtue of this keyword we will not be able to distinguish between documents. So, this particular word is not specific to particular document and therefore, log of 1 will be equal to 0; so i d f of this i-th key word is equal to 0.

And, since the weight value contains i d f as product component, the weight values become equal to 0. So, the meaning of this is that the weight value of a term of a keyword which appears in all documents is equal to 0. If we take the other extreme that is the keyword appear in a very small number of documents then, this tends to be large so log of this number would be smaller but, even then it will be large value. That means keywords which appear in sparse numbers documents; small numbers of documents have high distinguish power. So once you give that particular term in the query, if a particular document or 2 documents at most would come out.

And therefore, this have very high information content ok and therefore, there weightages also will be high. So this is the way the weightage scheme is decided for document vector and it makes lot of sense.

(Refer Slide Time: 29:15)



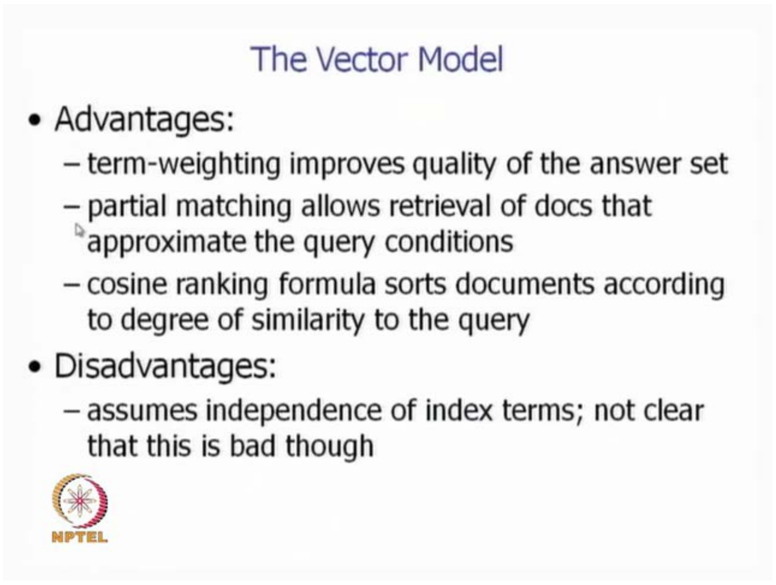
The Vector Model

- The best term-weighting schemes use weights which are give 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
  - $w_{iq} = (0.5 + [0.5 * freq(i,q) / \max_l(freq(l,q)]) * \log(N/n_i)$
- 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.

So, the best term weighting schemes use weights which are given by product of the term frequency and the inverse documents frequency this particular strategy is known as t f i d f weighting scheme. For the query terms weights a suggestion, is that we take the idea factor of course, we also take the term frequency. But, we have a multiplying factor of 0.5; this is the normalize terms frequency multiply by 0.5. We also add 0.5, so all these are actually empirically decided it is found to be a good formulation of the weight age of the term in the query.

And that comes from the fact that the query is typically a very small, it is a document code uncode document with a small number of terms and therefore, we have to see how to weight the terms in the query? Also it is not clear how to get the i d f value of query term with respect to the queries. So that is why this kind of empirical formulation is used. The vector model with t f i d f weights is the good ranking strategy general collection, the vector model is usually as good as the known ranking alternatives; it is also simple and fast to compute. So, over the years in a through many implementation through the engineering experiences, it has been found that the vector model really relievers the goods it brings up; up in the list the relevant documents. And hence, the systems have continued with the vector model, we exclude this model further.

(Refer Slide Time: 31:05)

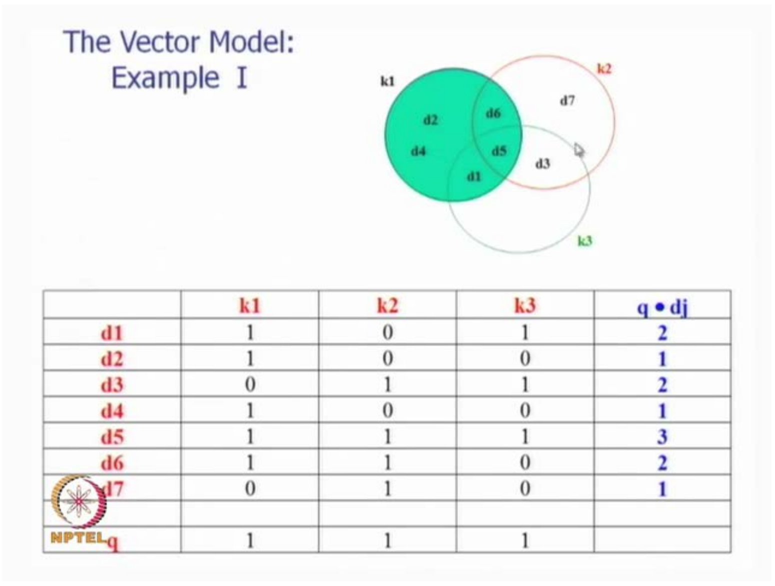


The Vector Model

- 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 advantage of the vector model are: term weighting improves quality of the answer set, partial matching allows retrieval of documents that are approximate the query conditions, cosine ranking formula sorts according to the degree of similar to the query. And the disadvantages that it assumes independence of index terms; so this is a fact that the terms are given weighted independent of their context. So, the weight of the term is simply obtained from its frequency in the document and it is i d f value. This is how the weight is calculated but, it is not clear that it is a bad decision; it is delivering the goods.

(Refer Slide Time: 31:47)



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	

And now, we take an example to understand the vector model, the effect of the weightage. So here is a venn diagram like picture there are 3 keywords: k 1, k 2 and k 3. The blue circles shows that in the circle of k 1 there are d 2 ,d 6, d 5, d 1 and d 4; this means that this documents d 1, d 2, d 4, d 5 and d 6 contain the keyword k 1. Similarly, this red circles shows that documents: d 3, d 5 and d 7 contain the keyword k 2 and this a light blue circle shows that, the keyword k 3 is contain in d 3, d 1, and d 5. A table represents the this fact document d 1 is represented as 1 0 and 1, we are giving a Boolean representation for the documents.

So k 1 is present, k 2 is absent, k 3 is present in document d 1, this is the case d 1 you see is the intersection of k 1 and k 3. Similarly, d 2 with we can see from here, d 2 is only in the region of k 1; it contains neither k 2 nor k 3 so, d two is 1 0 0. D 3 is shown as 0 1 1 because, d 3 is in the intersection of k 2 and k 3, k 1 does not appear there, so 0 11. Similarly, we have d 4 which is 1 0 0, d 5 is 1 1 1 is d 5 1 1 1 because, d 5 these are intersection: of k 1, k 2 and k 3. D 6 is 1 1 0, d 7 is 0 1 0 and query contains let me say all the keywords k 1, k 2 and k 3. Now, if we take a dot product of q and d j and the dividing factor which is the weightage, which is the modulus of q and modulus of d j that is same for all of them; so, the that will not change the score of the documents.

So here, we are taking the dot product of q and d j 111, d 1 score will be 2 d 2 score will be 1 1 0 0 and 111 and dot product is 1, d 3 score will be 2, d 4 score is 1, d 5 score is 3, d 6 score is 2 and d 7 score is 1. Now, you can see that, this particular scoring for this kind of vector creation is intuitive d 5 should indeed have a high score, because all the keywords are presenting this and d 6 and d 1 should have similar score because each of them contains a 2 out of 3 keywords. So let see d 1 score is 2 d 6 score also 2; d 2 and d 4 again should a same score of 1 because, only 1 q keyword k 1 is present in this and that is the case.

So, you see if you simply go by term presence or absence then d 5 must be the winning document, which is the case. Now, we change the weightages in the query ok the documents vector are changed, so they simply express term presence or absence however, the term weightages are changed in the query; we say the k 3 has a weightage of 3, k 2 has as weightage of 2, k 1 has a weighted of 1. So, k 1 may be present but, its weightag is smallest. So, when that happens, what we find is that: the score for d 1 is 4, score for d 2 is 1, score for d 3 is 5, score for d four is 1, score for d 5 is 6, score for d 6 is 3, score for d 7 is 2.

So, d 5 is again the winner because all keywords are present in this and their weight age gets added. However, we see a rearrangement of documents of other documents so, d 1 and d 6 had equal score in the last weighting scheme of 1 0. Here, we see that d 1 and d 6 have different score d 1's score has become higher because it contains a high weightage term k 3. Moving further, if we now have weightages reflected in the documents also, so in document d 1, k 1 weightage is 2, in document d 4 k 1 weightage is d 4, in d 5 its weightage is only 1 it is just present, without any weight modification. Similarly, k 3 has weightage of 1 in d 1, 3 in d 3, 4 in d 5 and so on, again we see d 5 is the wining document but, d 1 and d 6 score have now become same.

D seven which was nowhere before has a got high score now and its wearing a d 2 and d 6. So, this is the effect that the terms have when they are weighted; when terms are weighted the pictures changes very dramatically. Thus, term presence or absence gives one kinds of score for the documents ok however, when the weightages are changed then they may show a completely different picture. Documents which are high up in the list can go down and the other documents can rise up and this should be the case because, different keywords have different weight ages in the documents and this should be reflected.

And this also captures the facts that, the documents and queries should reflect with weighted of the terms leading to different relevance values. Now, the key question that arises is that, if the weighted so important and it changes the ranking of the documents quite dramatically. How do the weightages come? Where they come from? So we take information of the documents information of the whole corpus, the knowledge of the world, the knowledge of the domain from the documents come and then the weightages are formulated.

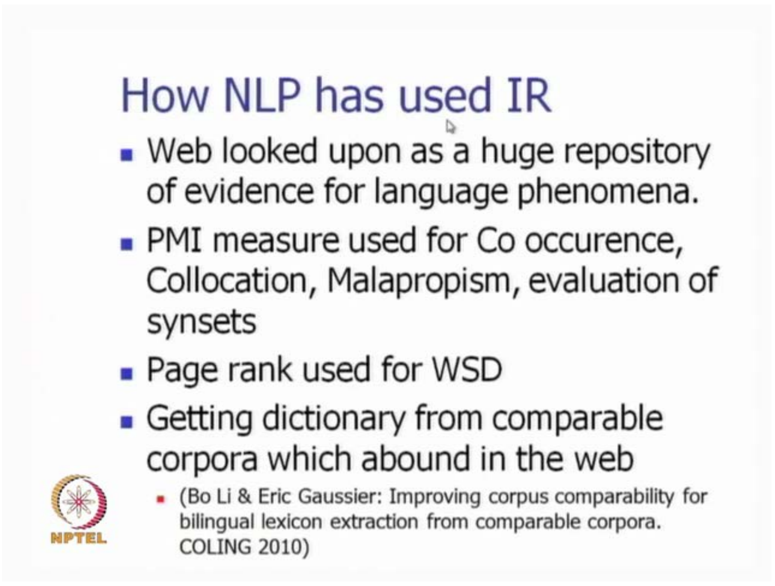
So, this is a this is an engineering problem; this is problem which requires understanding the problems very well. So, now we go back to our discussion of N L P and I R, so once we know that, I R is looking for solutions to the problem of term weighting ok that is the crucial point term weighting. The better the term weighting the better is the retrieval and we will see there is the another factor query refinement, query expansion this also has a role in the information retrieval scenario. Now, weightages of the term can come from natural language processing considerations. So, we go back to our discussion of the relation between natural language processing and information retrieval.

(Refer Slide Time: 39:25)

NLP and IR Relationship

N L P and I R relationships.

(Refer Slide Time: 39:27)



How NLP has used IR

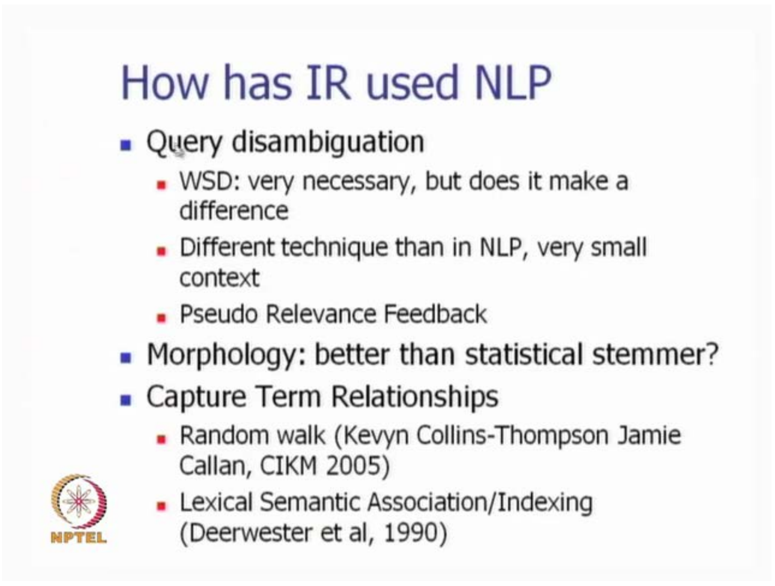
- Web looked upon as a huge repository of evidence for language phenomena.
- PMI measure used for Co occurence, Collocation, Malapropism, evaluation of synsets
- Page rank used for WSD
- Getting dictionary from comparable corpora which abound in the web
  - (Bo Li & Eric Gaussier: Improving corpus comparability for bilingual lexicon extraction from comparable corpora. COLING 2010)

1st question is how natural language processing has used information retrieval. The N L P has used I R in the following way: web is looked upon as huge repository of evidence for language phenomena. Where, this full of text, lots of corpora, attention of peoples, writings of writers, writing writings of common man, writings of people from: different languages, different backgrounds. So, web is a very is. repository of language phenomena.

Now, there is a very effective major called point wise mutual information which, has been used for many very fundamental N L P issues like: co occurrence, collocation etcetera. And also application like, malapropism detection, evaluation of synsets; so, this P M I measure is obtain against the corpse on the web. A very famous idea namely: page rank has been used for ((Refer Time 40:23)) page ranking from information retrieval literature; it has been made famous by information retrieval. And page rank like ideas have found application at many places in particular, disambiguation algorithm in un supervise setting has made use of page ranking.

Off late there has been lot of interest in generating probablistic dictionary from comparable corpora which abound in the web and this is recent work. So, again natural language processing systems have benefited from web and I R in the sense of getting probabilistic dictionary this research is on of course.

(Refer Slide Time: 41:07)



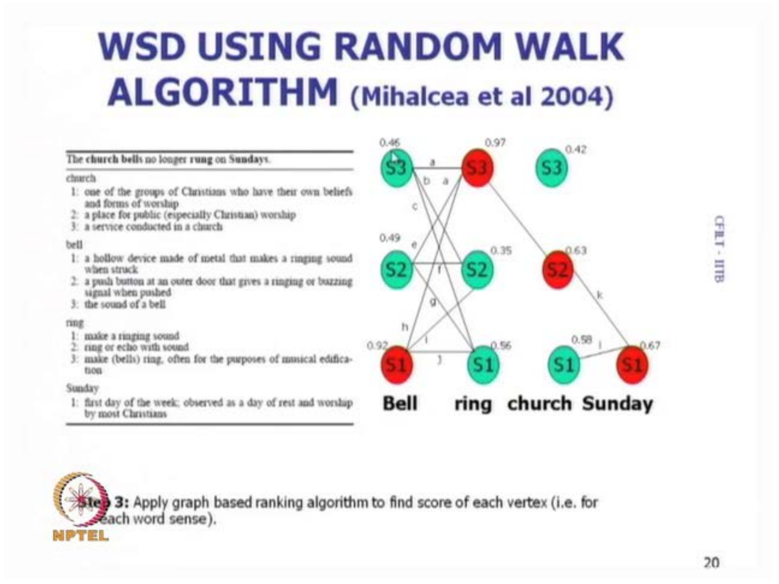
How has IR used NLP

- Query disambiguation
  - WSD: very necessary, but does it make a difference
  - Different technique than in NLP, very small context
  - Pseudo Relevance Feedback
- Morphology: better than statistical stemmer?
- Capture Term Relationships
  - Random walk (Kevyn Collins-Thompson Jamie Callan, CIKM 2005)
  - Lexical Semantic Association/Indexing (Deerwester et al, 1990)

And in term how has information retrieval used natural language processing. So, it has; I R has always been concerned with query disambiguation, query words are typically polysemy’s. when we do only a term base matching and ignore synonyms then, we have load recall if you are not careful about polysemy then its lead and tropic drift and lower precaution. So, disambiguation is important in information retrieval however, the techniques are very different from natural language processing because, the context is very small in query disambiguation. So, the field has used pseudo relevance feedback we will have to discuss this issue later.

Another important basic question which has exist in information retrieval is whether morphology should be or should not be done, is it better then statistical stemmer? Now, languages which are morphologically strong reach, it seems they need you know strong morphological processing. So morphology processing is very N L P concern and the statistical stemmer has been the invention of information retrieval. So, do they come together can we do without elaborate morphology processing is a question to be answered. But, yes information retrieval has benefited from the morphology analysis which, linguistic and N L P has contributed to the field. The other parties is something we mention few minutes back, term relationship have to capture term weightages have to be capture; this is a central concern in I R. And once is 2 kinds of work in this area: random work and lexical semantic association or indexing so, both of this would be discussed.

(Refer Slide Time: 43:02)



WSD USING RANDOM WALK ALGORITHM (Mihalcea et al 2004)

The **church bells** no longer **rung** on **Sundays**.

church
1: one of the groups of Christians who have their own beliefs and forms of worship
2: a place for public (especially Christian) worship
3: a service conducted in a church

bell
1: a hollow device made of metal that makes a ringing sound when struck
2: a push button at an outer door that gives a ringing or buzzing signal when pushed
3: the sound of a bell

ring
1: make a ringing sound
2: ring or echo with sound
3: make (bells) ring, often for the purposes of musical edification

Sunday
1: first day of the week; observed as a day of rest and worship by most Christians

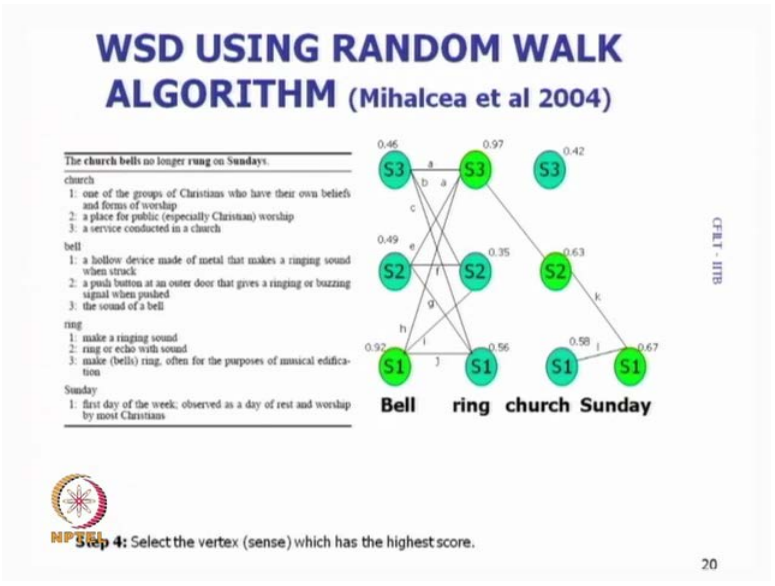
Step 3: Apply graph based ranking algorithm to find score of each vertex (i.e. for each word sense).

NPTEL

20

If we look at this problem of words and disambiguation using random work then, the problem is as follows: we have the words in a sentence and what happens is that on each word we column of senses. So, bell has 3 senses: 1, 2, 3; a hollow of device made of metals that makes a ringing sound when destruct, a push button at an outer that gives ringing or causing signal when pushed the sound of bell. So, this senses are reacted on the word bell. Similarly, ring church on Sunday all of them have senses shown, so then after that this senses are linked by arks and from each stage to the next stage there is complete connection. The weightages are as per the less similarity measure, find out the over lab of words in the glosses and examples sentences and there by create a graph.

(Refer Slide Time: 44:12)



WSD USING RANDOM WALK ALGORITHM (Mihalcea et al 2004)

The **church bells** no longer **rung** on **Sundays**.

church
1: one of the groups of Christians who have their own beliefs and forms of worship
2: a place for public (especially Christian) worship
3: a service conducted in a church

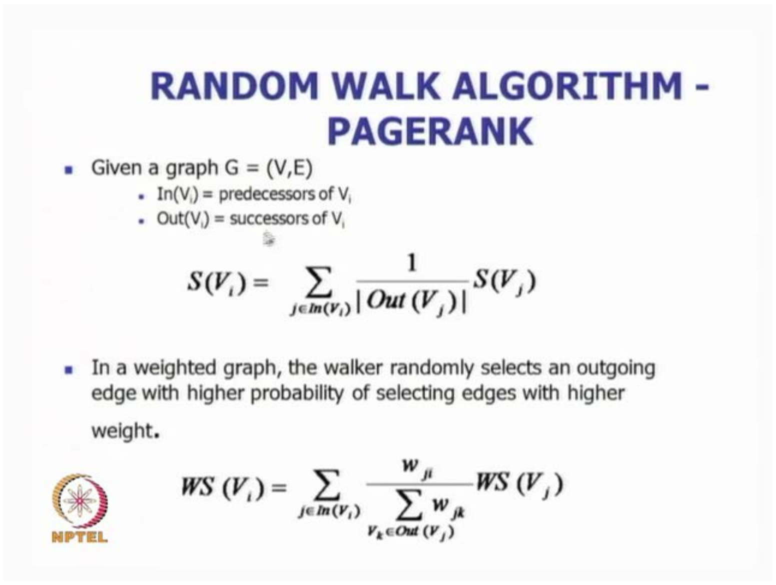
bell
1: a hollow device made of metal that makes a ringing sound when struck
2: a push button at an outer door that gives a ringing or buzzing signal when pushed
3: the sound of a bell

ring
1: make a ringing sound
2: ring or echo with sound
3: make (bells) ring, often for the purposes of musical edification

Sunday
1: first day of the week; observed as a day of rest and worship by most Christians

Step 4: Select the vertex (sense) which has the highest score.

20



RANDOM WALK ALGORITHM - PAGERANK

- Given a graph G = (V,E)
  - $In(V_i)$ = predecessors of V_i
  - $Out(V_i)$ = successors of V_i

$$S(V_i) = \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

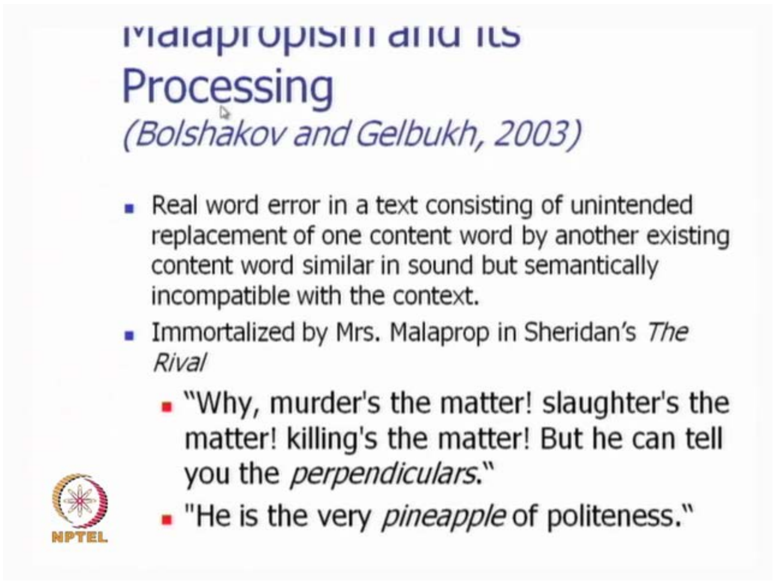
- In a weighted graph, the walker randomly selects an outgoing edge with higher probability of selecting edges with higher weight.

$$WS(V_i) = \sum_{j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

After that one runs this page rank algorithm and the scores get updated at iteration, using this formula which we are not detailing out now. Essentially, the idea is that every note has an importance imparted it by the weightages on the arks which decent on this notes and then from this notes the importance gets transfer to other notes. So, this is the way the notes transfer important from one stage to another and finally, we notes emerge at every column of senses.

So this is the random work algorithm; this also the page rank algorithm and this idea came from information retrieval, words and is disambiguation algorithm of ((Refer Time 45:07)) which works on unsupervised corpora, unmark corpora, unsense corpora or the setting is unsupervised learning with only word net as the this source that produces results of sense disambiguation.

(Refer Slide Time: 45:24)



Malapropism and its Processing

(Bolshakov and Gelbukh, 2003)

- Real word error in a text consisting of unintended replacement of one content word by another existing content word similar in sound but semantically incompatible with the context.
- Immortalized by Mrs. Malaprop in Sheridan's *The Rival*
  - "Why, murder's the matter! slaughter's the matter! killing's the matter! But he can tell you the *perpendiculars*."
  - "He is the very *pineapple* of politeness."

There is the very interesting problem where, N L P has made use of information retrieval and this problem, I describe as malapropism and it is processing. So, this problem has become important in correcting errors in text which are symmetric in nature. So, if they are spelling errors they can be corrected, if the syntactic errors then there are systems which do grammar check and come up with suggestions or and at least show that, something is wrong grammatical with the sentence. But, words which are spelt correctly but, semantically odd do not feed in the semantics of the context they are harder to detect. So, we will discuss this problem in the next lecture.