Retrieval modeling

Chapter 4 (4.5)

Chapter 7 (7.1,7.5) and Chapter 8

This subject and the textbook

In the textbook the subscripts in f_{ij} get swapped

that is what we denote f_{ij} in this subject,

the textbook might call it f_{ji}

Since in the textbook i runs through documents and j through terms

which is the reverse of what we do here

the effect might be the same

The textbook also uses q_j for w_{iq}

Introduction

Information retrieval systems assume that the semantics of the document and of the user query can be modeled precisely by a set of index-terms. This approach may be problematic for the following reasons.

(1.a) **Index-terms vs search terms (i.e. words).** A user request might get prepared using an imprecise space of search terms (e.g. all the words in a document) rather than a clearly/precisely defined set of index terms,

(1.b) **Query language.** The list of index terms may not capture the essence or the semantics of the query (Is the search for `A better than B` the same as `B better than A` when performing a keyword search?),

(1.c) **User query training.** Users are not trained to properly form their queries. (Explore semantics/meaning of `TO be OR NOT to be` and its variations.)

Thus it becomes unclear which documents of the output are relevant, and which are not. (The documents of the output would include all documents that contain the index terms of the query.) A **ranking** might take place to further refine and order the output.

- Ranking operates on premises relevant to the notion of document relevance.
- Distinct sets of premises yield distinct information retrieval (IR) models.
- The IR model one adopts determines what is relevant and what is not.

There are three classical information retrieval models: boolean, vector, and probabilistic.

- **2.1. Boolean Model.** In the **boolean model**, documents and queries are represented as sets of index terms and the model is *set-theoretic*. (The incidence matrix of Subject 7 is an example.)
- **2.2. Vector Model.** In the **vector model**, documents and queries are represented as vectors in t-dimensional space and the model is **algebraic**. Dimension t is the number of index-terms and the elements of the vector take not Boolean (i.e. 0-1) but arbitrary values.
- **2.3. Probabilistic Model.** In the **probabilistic model**, the modeling framework is based on probabilities and thus it is probabilistic.

Definitions

For the remainder of this subject we will use some standard notation introduced earlier in Subject 7 (page 3). In this section a query will be considered itself a document consisting of words that are the terms forming the query. Thus a query q will become by itself a document of the collection and a comparison between a query and a document of the collections is to be a comparison between documents.

- **3.1. Modeling Framework. F** is the **modeling framework** that we will be using at any time.
- **3.2. Queries. Q** is a set of query requests. Since we rank results of individual queries, q is one element of **Q**; the latter is thus becoming a single-element set.
- **3.3. Corpus. D** is a collection of **n** documents. The j-th document of the corpus is $\mathbf{d_j}$.
- **3.4. Index terms.** The i-th index term is $\mathbf{k_i}$, $1 \leq i \leq t$, and **t** is the **number of index-terms**.
- **3.5. Document frequency of k_i.** The term $\mathbf{n_i}$ is the number of documents in which k_i appears i.e. the **document frequency** of term k_i.
- **3.6. Term frequency or tf-factor f_{ij}.** $\mathbf{f_{ij}}$ is the raw frequency of k_i in d_j. This is the **term frequency** or **tf**-factor of k_i in d_j.
- **3.7. Normalized frequency.** $\mathbf{F_{ij}}$ is the normalized frequency of k_i in d_j i.e. $F_{ij} = \frac{f_{ij}}{\max\limits_m f_{mj}}$.
- **3.8. Inverse document frequency.** The term $\mathbf{idf_i}$ is the **inverse document frequency** for k_i i.e. $\mathbf{idf_i} = \log\left(1 + \frac{n}{n_i}\right)$. It gives high values for rare words, and low values for frequent ones. (Sometimes one can use $\mathbf{idf_i} = \log \frac{n}{n_i}$ instead.)

4.1 Note. In Subject 7 for specific examples we dropped subscripts and gave longer intuitive names. We instead of using n_i we used `Ndocs` several times, and instead of using f_{ij} we preferred `Nhits` for a given term k_i.

Two similarity (or dissimilarity) measures can then be defined.

4.2 Similarity measure: Intra-cluster similarity. Determine the features that best describe the objects of a set. A measure of similarity is the **tf-factor** f_{ij}.

4.3 Dissimilarity measure: Inter-cluster dissimilarity. Determine the features that best distinguish the objects of a set from those of the collection. A measure of dissimilarity is $\mathbf{idf_i}$.

Weights

Several times instead of using raw frequencies or numbers we prefer to use weighed frequencies. Although a normalized frequency is an alternative to term frequency, other variations might be possible.

- **5.1. Relative term frequency.** Variable $\mathbf{r_{ij}}$ is used to denote the **relative term frequency** of index term k_i in document d_j.
  (A choice for r_{ij} is $r_{ij} = F_{ij}$ but more often the following is used: $r_{ij} = (1 + \log f_{ij})$.)
- **5.2. Relative query frequency.** Variable $\mathbf{r_{iq}}$ is the **relative query frequency** of index term k_i in query q.
  (A choice for r_{iq} is $r_{iq} = 1$.)
- **5.3. Term weight.** Variable $\mathbf{w_i}$ is the **term weight** of index term k_i in the corpus and is defined as the inverse document frequency.
  (Thus $w_i = \mathbf{idf_i} = \log\left(1 + \frac{n}{n_i}\right)$.)
- **5.4. Document-term weight.** Variable $\mathbf{w_{ij}}$ is the **document-term weight** of index term k_i in document d_j.
  (One choice for w_{ij} is r_{ij}, i.e. $w_{ij} = r_{ij} = (1 + \log f_{ij})$.)
- **5.5. Query-term weight.** Variable $\mathbf{w_{iq}}$ is the **query-term weight** of index term k_i in query q.
  (One choice for w_{iq} is $w_{iq} = r_{iq} w_i = \log\left(1 + \frac{n}{n_i}\right)$).)

6. Default values.

$$\mathbf{idf_i} = \log\left(1 + \frac{n}{n_i}\right) \qquad \mathbf{r_{ij}} = (1 + \log f_{ij}) \quad , \quad \mathbf{r_{iq}} = 1 \tag{1}$$
$$\mathbf{w_{ij}} = r_{ij} \quad , \quad \mathbf{w_{iq}} = r_{iq}\, \mathbf{idf_i} = \mathbf{idf_i}$$

7. Similarity Measure $\mathbf{s(q, d_j)}$. This way, for a query q and document d_j the similarity $s(q, d_j)$ can be expressed by a formula of the form shown below. (The denominator $\sum_i f_{ij}$ is the number of terms in d_j.)

$$s(q, d_j) = \frac{\sum_i w_{ij} w_{iq}}{\sum_i f_{ij}} = \frac{1}{\sum_i f_{ij}} \cdot \sum_i (1 + \log f_{ij}) \log\left(1 + n/n_i\right) \tag{2}$$

7.1 When we discuss the vector model on pages 8-9, more variations of Eq. 2 will be given.

Weights . . . more options

8. Alternative choices for w_{ij} and w_{iq}. Although Equation (1) provides some choices for w_{ij} and w_{iq} these are not the only possible. Some more are shown below.

8.1 Document-term weight: Option A1. The value of the document-term weight can be expressed sometimes by the **term-weight frequency** i.e.

$$w_{ij} = F_{ij} \times idf_i$$

8.2. Document-term weight: Option A2. A normalized term is given by

$$w_{ij} = \frac{f_{ij}\mathbf{idf_i}}{\sqrt{\sum_i f_{ij}^2 \mathbf{idf_i^2}}}$$

8.3 Salton and Buckley query-term weight: Option B1. A variant is defined (Salton and Buckley) as

$$w_{iq} = \left(0.5 + \frac{0.5 * F_{iq}}{\max_l f_{lj}}\right) \times \mathbf{idf_i}$$

Definitions

In order to establish the relevance of the documents of a corpus to a given query q, a similarity measure of each document d_j of the corpus relative to query q is to be established. This process involves a number of steps as follows.

9.1. Step 1: Choose similarity measure $\mathbf{s(q, d_j)}$. For query q and document d_j the similarity $s(q, d_j)$ may be established by way of Equation (2), where w_{ij}, w_{iq} are from Equation (1). Thus

$$s(q, d_j) = \frac{\sum_i w_{ij} w_{iq}}{\sum_i f_{ij}}$$

The similarity measure is an information retrieval measure of the relevance of d_j to query q. A ranking of d_j relative to query q can be established then. The **ranking function $\mathbf{R(q, d_j)}$** can use similarity-based measure(s) exclusively or incorporate other factors/measures as well such as the importance of document d_j in the corpus.

9.2 Step 2: Determine rank $\mathbf{R(q, d_j)}$.

9.2.1. Rank based on information-retrieval terms only (i.e. similarity-based rank).

$$R(q, d_j) = s(q, d_j).$$

9.2.2. Rank based not entirely on information-retrieval terms.

$$R(q, d_j) = s(q, d_j) + \textbf{Other factors}.$$

10. The Boolean Model. It is a very simple model based on set theory. According to the model a term either appears or does not appear in a document. Thus in order to determine the weight of term k_i in document d_j i.e. w_{ij} we set $w_{ij} = 1$ if k_i appears in d_j and we set $w_{ij} = 0$ otherwise. The frequency f_{ij} does not matter at all. One can thus arrange an **incidence matrix** with the documents becoming the rows and the index-terms the columns (as we did in Subject 7). A column becomes the incidence vector of a given term; its ones indicate the documents containing the term (once or more than once times). Similarly, a given row is a descriptor of a given document. Its ones show the index-term appearing in the document. In the Boolean model, a document becomes a binary (boolean) row vector of terms e.g. $d_i = (k_1, \ldots, k_i, \ldots)$.

10.1 Queries in the boolean model. Queries are specified in the form of a boolean expression using boolean operators such as `AND, OR, NOT` also denoted by $\wedge$, $\vee$ and $\neg$ respectively. Queries are usually written in DNF (Disjunctive Normal Form) i.e. as a disjunction of conjunctions. (Note that humans prefer to express their queries in Google in conjunctive normal form instead.)

10.1.1 Example query Q_1 in conjunctive form. A query Q_1 for example can be written as $Q_1 = q_1 = k_1 \wedge (k_2 \vee k_3)$ to determine all documents that contain term k_1 and one or both of k_2, k_3.

10.1.2 Example query Q_1 in DNF form. Query Q_1 in DNF should read as $Q_1 = q_1' = (k_1 \wedge k_2) \vee (k_1 \wedge k_3)$.

10.2 Similarity $s(q, d_j)$ between query q and document d_j. Let query q in DNF form be expressed by q'. Let r be any of the conjunctive terms of q' (such as $k_1 \wedge k_2$) , $k_1 \wedge k_3$). The similarity $s(q, d_j)$ of q and d_j is 1 if and only if there is a conjunctive term in that is true, and 0 otherwise (i.e. all conjunctive terms are 0).

10.3 Advantages of the Boolean model: Simplicity, clean formalism.

10.4 Disadvantages of the Boolean model: Exact matching only supported (too few or too many hits). No partial matching. No ranking. Users need to translate queries into DNF.

10.5 Index term weighing *i.e. the use of w_{ij} that is neither 1 nor 0 but reflects the values of f_{ij} is a better approach, and thus the boolean model is never better than the next model, which is the* **vector model**.

Introduction

11. The Vector Model. In the **vector model**, the documents are represented as vectors. This is the approach of the boolean model. The difference between the vector and boolean models is that the document vectors were boolean (vectors of zeroes and ones) in the boolean model whereas in the vector model are not (necessarily) boolean.

11.1 Document representation as a vector. Thus document d_j is represented by a vector
$d_j = (w_{1j}, \ldots, w_{ij}, \ldots, w_{tj})$
of components w_{ij}, where w_{ij} denotes the weight of term k_i in document d_j.

11.2 Query representation as a vector. Queries are also formulated as vectors and thus query q is represented by a vector
$q = (w_{1q}, \ldots, w_{iq}, \ldots, w_{tq})$
i.e. the query becomes a "short document" itself. (In such a case, the weight w_{iq} of a term that appears in the query is one, i.e. $w_{iq} = 1$.)

11.3 Query-document similarity $s(q, d_j)$. The similarity between query q and document d_j is thus defined as a similarity measure between two "documents", and this similarity measure is then defined as the similarity between two vectors.

11.4 Expressing $s(q, d_j)$ in the vector model. The similarity $s(q, d_j)$ can be defined through traditional similarity measures $S(q, d_j)$ between vectors. Such measures $S(q, d_j)$ include the following.

Similarity Measures

- **11.5 Euclidean Distance**. $S(q,d_j) = \sqrt{\sum_i (w_{ij} - w_{iq})^2}$. This is a **dissimilarity** measure, since a non-zero value is indicative of the distance between the vectors, or its closeness to zero indicates the similarity between the document and the query. (The query that is usually short discriminates against long documents.) Then $s(q,d_j) = S(q,d_j)$.
- **11.6. Dot-Product or Inner-product Distance**. $S(q,d_j) = \sum_i w_{ij} * w_{iq}$. In this case the query discriminates in favor of long documents posing the opposite problem of euclidean distance. What should thus be of interest is the difference in directions ie the angle between the vectors. Then $s(q,d_j) = S(q,d_j)$.
- **11.7. Cosine** $S(q,d_j) = \dfrac{q \cdot d_j}{|d_j| \times |q|} = \dfrac{\sum_i w_{ij} * w_{iq}}{\sqrt{\sum_i w_{ij}^2}\sqrt{\sum_i w_{iq}^2}} = \dfrac{\sum_i w_{ij} * w_{iq}}{W_j W_q}$. This similarity measure captures the difference in direction of the vectors of the query q and document d_j. It also justifies the normalization proposed in the bottom of page 5 for $s(q,d_j)$ there. If the angle is zero between q, d_j then the cosine is 1; if the two are orthogonal to (different from) each other then cosine is 0. (One could set $s(q,d_j) = S(q,d_j)$.)
- **11.8. Manhattan Distance**. $S(q,d_j) = \sum_i |w_{ij} - w_{iq}|$. Then $s(q,d_j) = S(q,d_j)$.

We may use the cosine definition 11.7 and incorporate it into Equation (1) to derive a similarity measure close to but not the same as Equation (2).

- **11.9 Updated similarity measure $s(q,d_j)$ over Equation (2).**

$$s(q,d_j) = \frac{1}{W_j W_q} \times \sum_{i \in q \cap d_j} (1 + \log f_{ij}) \cdot \log(1 + n/n_i). \tag{3}$$

$$W_j = \sqrt{\sum_i w_{ij}^2} \qquad w_q = \sqrt{\sum_i w_{iq}^2} \qquad w_{ij} = (1 + \log f_{ij}) \qquad w_{iq} = \mathbf{idf_i}$$

Similarity and Ranking

11.10 Comparison of measures: 11.5, 11.6 and 11.8. For measures 11.5, 11.6, and 11.8, the closer the distance, the smaller the $S(q, d_j)$ is and the better the similarity will be. That is, **lower** $S(q, d_j)$ **values imply close similarity**.

11.11 Cosine measure 11.7. For the cosine measure 11.7, the closer the angle, the smaller the angle is and the closer $S(q, d_j)$ is to 1. That is, higher $S(q, d_j)$ values imply close similarity. Cosine is a normalized similarity metric.

11.12 Partial matches and ranking. The existence of non-binary weights (attributes of the vectors) provide for partial matches. **A degree of similarity is possible and ranking is feasible.**

11.13 Answering a query q **for a corpus** D **using similarity measures only.** The process of **answering a query** q thus becomes obvious.

- For all documents d_j determine similarity $s(q, d_j)$.
- Rank documents based on similarity measure (exclusively).

The vector model
Reliability of similarity measures

11.14 Reliability issues: frequent words. Keyword frequencies used in similarity measures favor frequent words which might not convey enough information about the document.

11.14.1 Problems using f_{ij} for w_{ij}. The tf-factor f_{ij} therefore should not be overemphasized. A better measure becomes the normalized term-frequency factor F_{ij} but again by not much.

11.15 Reliability issues: rare words. Inter document dissimilarity that relies on idf, i.e. the inverse document frequency may alternatively be used. This measure gives high values for rare words and thus reduces the importance or effect of frequent words. Thus idf_i can be used instead of F_{ij} or f_{ij}.

11.16 Reliability issues: frequent and rare words and the tf-idf weight. One can combine the two contributions by using f_{ij} or F_{ij} in conjunction with idf_i for w_{ij}. Therefore a weight factor for w_{ij} that can be used in the vector model is $w_{ij} = F_{ij} * idf_i$ which is also known as the **tf-idf** weight. A better approach is to use a normalized logarithmic component for F_{ij} as in $(1 + \log f_{ij})$ multiplied by idf_i. This is the form used in Equation (4).

As a conclusion,

11.17. Advantages of the vector model. Good ranking (term weighing, partial matching, cosine similarity).

11.18 Disadvantages of the vector model. Assumes terms are independent; does not handle synonyms. Query weighing may not reflect relevance. It also favors longer over shorter documents. Similarity is not (may not be) scaled.

Computing $s(q, d_j)$ in practice

11.19 Computation time of answering a query q for corpus D. For each query q, in order to determine the relevant to the query documents, one needs to compute $s(q, d_j)$ for all documents d_j in the collection D.

There are n documents and t terms, thus this computation becomes very expensive at $O(nt)$. This was discussed for incidence matrices in Subject 7.

For a corpus of $n = 1,000,000,000$ documents and $t = 100,000,000$ terms, the computation cost is prohibitively expensive.

The sparsity of the incidence matrix implies that an adjacency-list-based rather than an adjacency-matrix-based approach can be used instead.

With this in mind in the computation of $s(q, d_j)$ only the index terms that appear in document d_j and query q need to contribute a relevant term to $s(q, d_j)$. By using the formula of Equation (4) and weights from Equation (1) we get the pseudocode of Figure 1.

The running time of Figure 1 is proportional to n (the number of documents) and $\sum_i n_i$ where n_i is the number of documents in which k_i appears, rather than $O(nt)$. **We also note that the sum $\sum_i n_i$ is over index i, and index i runs through the index-terms of the query i.e. for all practical purposes i takes few values (for practical queries and consulting the statistics of Subject 2, i is no more than 4 or 5 since the average query rarely has more than 4 or 5 terms)**. Thus $\sum_i n_i$ in the worst case (if every $n_i = n$) would be no more than $O(n)$, a t factor improvement over $O(nt)$.

Note also that in the code with the intent to be thorough we included lines 7-10 that compute W_q and also computed during query time W_j in lines 1-5. The latter can be precomputed at indexing time and stored within the inverted table as a property of a document (or possibly in the doclist that maintains information about the documents of the corpus). The former computation could have been incorporated into lines 11-23. Alternately we can avoid the division by $W_j * W_q$ altogether and the similarities won't be scaled.

Computing $s(q, d_j)$ in practice (continued)

```
   s(q, d[0..n-1]) // Compute similarity for query q for all n documents of the collection
1. for(j=0;j < n ; j++)  {
2.    s(q,d[j])=0.0;                          // Initialize a counter for each document.
3.    for (i=0;i=Length(d(j));i++)            // this depends on the length of d(j) not on t itself!
4.    W(j)=Retrieve(w(i,j)*w(i,j));           // This retrieves rather than computes W(j)
5. }                                          // This can be precomputed...
6. Wq = 0;
7. for(i=0;i < q.length(); i++) {             // Go through each term of the query one by one
8.  ki = q.term[i];                           // Let the i-th term of query q be ki
9.  Wq= Wq+ idf(ki)*idf(ki);
10.}
11.for(i=0;i < q.length(); i++) {             // Go through each term of the query one by one
12. ki = q.term[i];                           // Let the i-th term of query q be ki
13. Wq= Wq+ idf(ki)*idf(ki);
14. Location= LocateInInvertedIndex(ki);      // Locate the entry for ki in the Vocabulary
15. Ndocs = InvertedIndex[Location].Ndocs;    // Locate the Ndocs for term ki
16.  for(k=0;k<Ndocs;k++) {                   // Go through the Ndocs documents that contain ki
17.     DocIndex = LocP+k;                    // The k-th document is in location LogP+k of the
18.     docID=OccurrenceList[DocIndex].docID;// occurrence list
19.     // Compute the contribution to the document/term pair in s(q,docID);
20.     s(q,docID) = ((1+log(OccurrenceList[Docindex].Nhits)) *
21.                  log(1+N/InvertedIndex[Location].Ndocs)))/(W(docID)*W(q));
22.   }//and repeat for all docs of ki.In line 7 formula (3) for modified S3
23. } // and repeat for all query terms ki of q
```

Figure 1: Similarity computation in the vector model (adjacency-list approach)

12. Probabilistic model. The **Probabilistic** model tries to capture the Information Retrieval problem within a probabilistic framework i.e. it tries to assign probabilities to documents so that a probability signifies the relevance of the document to the given user query. Under this model, given a query, there is an *ideal* answer set that satisfies the query. The framework tries to guess/estimate an initial description of an ideal answer set. and then improve the initial guess/estimate by iteratively refining it. At every iteration of this process, the user's input is used to refine the answer set.

12.1 Disadvantages of the probabilistic model. One disadvantage is the guessing of the initial answer set. Also, it does not use **tf , idf** information at all. Experiments due to Salton and Buckley confirmed that the vector model is superior.

13. Other models. Besides the boolean, vector and the probabilistic models, other models have been proposed. These include an extension to the boolean model, and one after fuzzy set theory.
13.1. Fuzzy Set Model which is an alternative set-theoretic model. A query term defines a fuzzy set and each document has a degree of membership to that set (a real number between 0 and 1). In the boolean model, by contrast, membership is a clearly (binary) defined relationship.
13.2. Extended Boolean Model extends the boolean model with vector model capabilities to rank query results. Consider a simple query consisting of two terms k_1 and k_2 i.e. $q = k_1 \wedge k_2$ or $q = k_1 \vee k_2$. Define the unit square on the plane with sides equal to 1. Then determine x, y values in terms of td-idf information of k_1, k_2 in a document d_j. The similarity of q, d_j is then expressed as the distance of point (x, y) of two-dimensional space from certain vertices of the square.

```
For Document dj
   (0,1)  ---------------- (1,1)  For a  query q= k1 AND k2
          |               |         similarity s(q,dj) is normalized distance from point (1,1)
      y ' ' ' ' ' 'o      |         i.e. 1-sqrt(((1-x)**2 + (1-y)**2)/2)
          |        '      |       For a  query q= k1 OR  k2
          |        '      |         similarity s(q,dj) is normalized distance from point (0,0)
          |        '      |         i.e. sqrt((x**2+y**2)/2)
          |        '      |       Point is at o (x,y) where x,y are defined
          |        '      |         in terms of td-idf of k1 and k2 in dj.
   (0,0)  ---------'------ (1,0)
                   x
```

Figure 2: Extended Boolean model

Computation and Analysis

14.1. Set of all documents in the corpus. Consider a query request q on a corpus of n documents.

14.2. Set A of relevant documents. The set of relevant documents to the query is A and its cardinality is a.

14.3. Set B of retrieved documents. An information retrieval policy retrieved the documents of set B. Its cardinality is b.

14.4. Set T of retrieved documents that are relevant. The set $T = A \cap B$ contains all the retrieved documents that are relevant. Let its cardinality be t.

The top-left element of the table (disregarding the TOTAL rows and columns) shown in Figure (3) shows the cardinality of T the number of retrieved documents that are relevant. Then $b-t$ is the number of retrieved documents that are not relevant (i.e. they irrelevant) to the query. This the top-right element of the table shown in Figure (3).

The number of relevant documents not retrieved is $a-t$ i.e. the difference between the number of all relevant documents in A minus those relevant documents retrieved in B and formed T. This is the bottom-left element of the table shown in Figure (3).

What needs to be confirmed is the number of irrelevant document not retrieved. The collection has n document of which a are relevant. Thus the number of irrelevant documents is $n-a$. Those retrieved are $b-t$, thus those not retrieved are $n-a-b+t$.

The ratio $(b-t)/(n-a)$ is called the **fallout** and gives the fraction of the irrelevant but retrieved documents over all the irrelevant documents.

	RELEVANT	IRRELEVANT	(TOTAL)
RETRIEVED	t	b-t	b
NOT RETRIEVED	a-t	n-a-b+t	n-b
(TOTAL)	a	n-a	

```
PRECISION  P =   t / b       The first-row information is used to derive P.
RECALL     R =   t / a       The first-column information is used to derive R.
A= set of relevant documents    B=set of retrieved documents   T= relevant documents retrieved
```

Figure 3: Retrieval and Recall

Overview

15 Tag information. For web-based documents that are in HTML format, one can formulate weight factors that reflect the position of the index terms in the HTML text. The Webor Method (Cutler et. al.) partitions HTML tags into ordered classes. It then determined approximate percentages of the distribution of term occurrences among a set of six ordered classes. These are listed below.

- **title** (eg. <TITLE>), Distribution: $< 1\%$.
- **anchor** (eg <A>), Distribution: $< 3\%$.
- **header** (eg. <H1> to <H6>), Distribution: $\approx 4\%$.
- **list** (eg. <DL>,<OL>,<UL>), Distribution: $\approx 5\%$.
- **strong** (eg. <STRONG>, <B>, <EM>, <I>, <U>), Distribution: 10%.
- **plain** (eg. none of the above) Distribution: $< 75\%$.

Then, instead of using a single tf-value f_{ij} or F_{ij} for a term to describe w_{ij}, Webor uses a tf-vector (tfv) such as

$$\mathbf{tfv} = (tf_1, tf_2, tf_3, tf_4, tf_5, tf_6)$$

where tf_i is the tf-term for occurrences of the index term in a class i tag in the document. For each element of the vector different **importance values (iv)** are then assigned. Let iv_i be the importance values for the i-th class.

The new term-frequency is then computed by the combination of the tf-values of the classes weighed appropriately.

$$\sum_i tf_i \cdot iv_i.$$

15.1 Some conclusions based on experimentation: Anchors and strong/emphasized text are important, header information is less, title is slightly just above plain text and listings.

Introduction for the next Subject

The relevance of a query to a document (e.g. a web-page) is established by first computing the function $s(q, d_j)$ that determines the similarity between query q and document d_j.

This similarity measure is based solely on information retrieval-based knowledge (e.g. the searchable contents of the document). Search engines (or web-searching methods in general) do not rely on this measure exclusively to rank a web-page (e.g. document d_j).

They also use additional information such as the link structure of d_j. The link structure relates to information such as:

- (a) what documents/pages are pointed by d_j and are these pages important (e.g. authoritative),
- (b) what documents/pages point to d_j and are these pages important (e.g. hubs of information such as a directory page).

All such link information can be made to contribute a separate component to the rank of d_j or a web-page in general.

Thus the **rank** of d_j relative to query q denoted by $R(q, d_j)$ is a function that will represent the rank of a document/web-page. One component of $R(q, d_j)$ is $s(q, d_j)$.

One of the first attempts to use linkage information was the **vector spread activation model** by Yuwono.

Vector spread activation model. The ranking score $R(q, d_j)$ is the sum of the similarity $s(q, d_j)$ of d_j plus a portion of the similarity of each document that points to d_j. This is under the assumption that if relevant documents point to d_j, then d_j should also be relevant. Let $l(i, j)$ be 1 or 0 based on whether document d_i points to d_j or not. Then

$$R(q, d_j) = s(q, d_j) + \beta \sum_i R(q, d_i) l(i, j)$$

where β is a user defined parameter (e.g. $\beta = 0.2$).

References:

1. M. Cutler, Y. Shih, and W. Meng: Using the Structures of HTML Documents to Improve Retrieval. Usenix Symposium on Internet Technologies and Systems (USITS'97) . pp.241-251, Monterey, California, December 1997.

2. M. Cutler, H. Deng, S. Manicaan, and W. Meng: A New Study on Using HTML Structures to Improve Retrieval. Eleventh IEEE Conference on Tools with Artificial Intelligence (ICTAI'99) , Chicago, November 1999.