University of Palestine

Topics In CIS - ITBS 3202

Ms. Eman Alajrami

2nd Semester 2008-2009

Chapter 2– Part2

Information Retrieval Models

Vector Model

Basic Concept

- Each document is described by a set of representative keywords called index term.
- Index term: is simply a word whose semantic help in remembering the documents main themes.
- Consider a collection of one hundred documents, a word which appears in each of the one hundred document is completely **useless** as an index term, because it does not tell us anything about which documents are the user interested in.
- But the word which appear in just 5 document is quite useful, because it narrow down considerably the space of documents which might be of interest to the user.

Retrieval Strategy

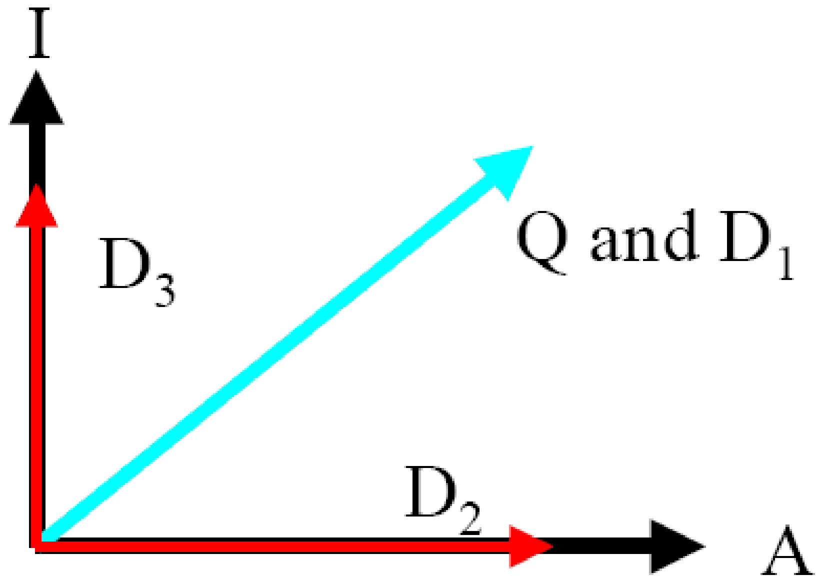
- An IR ***strategy*** is a technique by which a relevance measure is obtained between a query and a document.
- Manual Systems
  - Boolean, Fuzzy Set
- Automatic Systems
  - ***Vector Space Model***
  - Language Models
  - Latent Semantic Indexing
- Adaptive
  - Probabilistic, Genetic Algorithms , Neural Networks, Inference Networks

Vector Space Model

- Most commonly used strategy is the vector space model (proposed by Salton in 1975)
- It recognize that the use of binary weights is too limiting and propose a framework in which ***partial matching*** is possible.
- Partial matching is accomplished by assigning non-binary weights to index term in queries and documents.
- Term weights are used to compute the degree of similarity between each document stored in the system and user query.
- Documents and Queries are mapped into term vector space.
- Documents are ranked by closeness to the query.

Document and Query Presentation in VSM (Example)

- Consider a two term vocabulary, A and I
- D1-A I
- D2-A
- D3–I
- Query: A I



- Idea: a document and a query are similar as their vectors point to the same general direction.

Weights for Term Components

- Using Term Weight to rank the relevance.
- Parameters in calculating a weight for a document term or query term:
  - ***Term Frequency*** (*tf*): Term Frequency is the number of times a term *i* appears in document *j* (tf_{ij})
  - ***Document Frequency*** (*df*): Number of documents a term *i* appears in, (df_i).
  - ***Inverse Document Frequency*** (*idf*): A discriminating measure for a term i in collection, i.e., how discriminating term i is. (the term which appear in many documents are not very useful for distinguishing a relevant document from irrelevant one)

$(idf_i) = \log_{10} (N/ ni)$, ***where***

N= number of documents in the collection

n_i = number of document that contain the term *i*

Computing weights $w_{i,j}$ (*tf-idf*)

- How can we compute the values of the weights $w_{i,j}$?
  - One of the most popular methods is based on combining two factors:
    - The importance of each index term in the document (tf)
    - The importance of the index term in the collection of documents (idf)
  - Combining these two factors we can obtain the weight of an index term *i* as :

$$\mathbf{w_{ij} = tf_{ij} * idf_i = tf_{ij} * \log_{10} (N/ ni)}$$

Where:

N= number of documents in the collection

n_i = number of document that contain the term *i*

- Also called the ***tf-idf*** weighting scheme

Computing weights $w_{i,j}$ (tf-idf) An Example

- We have created our sample collection which consists of tow documents D0, D1, and a query Q as the following

Document ID	**Document text**
D0	Faculty of Information Technology and Computer Science
D1	Information retrieval course is in Computer Information systems

Q	**"computer information retrieval system"**

Document ID	***tf***
D_0	
Faculty	1
Information	1
Technology	1
Computer	1
Science	1
D_1	
Information	2
Retrieval	1
Course	1
Computer	1
Systems	1

Term ID	**Term**	***Df (n)***	**idf**
t1	Faculty	1	$\log_{10} 2/1=0.301$
t2	Information	2	$\log_{10} 2/2=0$
t3	Technology	1	$\log_{10} 2/1=0.301$
t4	Computer	2	0
t5	Science	1	0.301
t6	Retrieval	1	0.301
t7	Course	1	0.301
t8	System	1	0.301

Computing weights $w_{i,j}$ (tf-idf) An Example Cont…

Now we calculating the weight for each term by

Wij = tf X **idf**

	Term ID							
	T1	T2	T3	T4	T5	T6	T7	t8
D0	1*.301 = .301	1*0=0	1*.301 = .301	1*0=0	1*.301 = .301	0*.301 =0	0*.301 =0	0*.301 =0
D1	0*.301 =0	2*0=0	0*.301 =0	1*0=0	0*.301 =0	1*.301 = .301	1*.301 = .301	1*.301 = .301
Q	0	1*0=0	0	1*0=0	0	1*0.301 =0.301	0	1*0.301 =0.301

Finally we can use the result to compute the similarities between the Documents i and specified query by using one of the Similarities methods.

Similarity Measure

- A similarity measure is a function that computes the degree of similarity between two vectors(Documents and Query)
- Using a similarity measure between the query and each document:
  - It is possible to rank the retrieved documents in the order of relevance.
  - It is possible to enforce a certain threshold so that the size of the retrieved set can be controlled.

Similarity Methods

- Inner Product (dot Product)
- Cosine
- Dice
- Jaccard

Inner Product (dot Product)

- Similarity between vectors for the document ***d****$_i$* and query ***q*** can be computed as the vector inner product:

$$\sum_{k=1}^{t}(d_{ik} \bullet q_k)$$

where d_{ik} is the weight of term *i* in document *k* and q_k is the weight of term *i* in the query

Cosine Similarity

- Very commonly used
- Measures cosine of angle between document-query (or document-document) vector

$$\frac{\sum_{k=1}^{t}(d_{ik} \bullet q_k)}{\sqrt{\sum_{k=1}^{t} d_{ik}^{2} \bullet \sum_{k=1}^{t} q_k^{2}}}$$

where *dik* is the weight of term *i* in document *k* and *qk* is the weight of term *i* in the query

Jaccard Similarity

$$\frac{\sum_{k=1}^{t}(d_{ik} \bullet q_k)}{\sum_{k=1}^{t} d_{ik}^{2} + \sum_{k=1}^{t} q_k^{2} - \sum_{k=1}^{t}(d_{ik} \bullet q_k)}$$

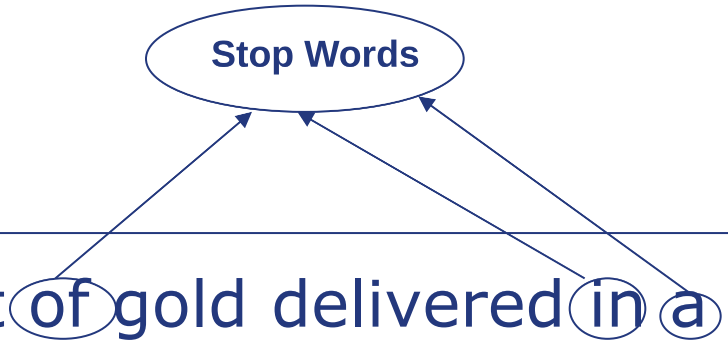
where *dik* is the weight of term *i* in document *k* and *qk* is the weight of term *i* in the query

Dice Similarity

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

Vector Model Example

- An example of building index structure is shown with a sample collection A with documents D0,D1,D2,D3,D4, and query Q.



D1: “Shipment of gold delivered in a fire”

D2: “Delivery of silver arrived in a silver truck”

D3: “Shipment of gold arrived in a truck”

Q: “gold silver truck”

Vector Model Example Cont…

1. In the First step, we identify the number of how many times the term has appeared in each document which is referred as the ***term frequency*** (**t*f***).

D1: "Shipment of gold delivered in a fire"
D2: "Delivery of silver arrived in a silver truck"
D3: "Shipment of gold arrived in a truck"
Q: "gold silver truck"

Document ID	***tf***
D1	
Shipment	1
gold	1
delivered	1
Fire	1
D2	
Delivery	1
Silver	2
arrived	
truck	2
D3	
Shipment	1
gold	1
arrived	1
truck	1

Vector Model Example Cont…

2. After that we need to determine the number of documents a term appears in, which is known as ***document frequency*** **(d*f*)**, also we need to compute the ***inverse document frequency*** **(id*f*)** which is denoted by the following equation:

   id*f* = log10 (N / ni), where **N** is the total number of documents and **ni** is the number of documents in which the term i appears in.

Vector Model Example Cont…

D1: "Shipment of gold damaged in a fire"
D2: "Delivery of silver arrived in a silver truck"
D3: "Shipment of gold arrived in a truck"
Q: "gold silver truck"

Term ID	**Term**	***Df (ni)***	**Idf= log10 (N / ni)**
t1	arrived	2	$\log_{10} 3/2=0.176$
t2	damaged	1	0.477
t3	Delivery	1	0.477
t4	Fire	1	0.477
t5	gold	2	0.176
t6	silver	1	0.477
t7	Shipment	2	0.176
t8	truck	2	0.176

Vector Model Example Cont…

3. Now we generate the sparse matrix which represents the collection and the weights of each term in each document which can computed by multiplying the ***term frequency*** by the ***inverted document frequency*** as in the following equation:

tfi X **idf***.* This operation is implemented for both, the documents and the query.

Vector Model Example Cont…

The Weights for each term listed in the following spars matrix :

	Term ID							
	T1	T2	T3	T4	T5	T6	T7	T8
D1	0	1*0.477 =0.477	0	1*0.477 =0.477	1*0.176 =0.176	0	1*0.176=0.176	0
D2	0.176	0	0.477	0	0	2*0.477 =0.954	0	0.176
D3	0.176	0	0	0	0.176	0	0.176	0.176
Q	0	0	0	0	0.176	0.477	0	0.176

Vector Model Example Cont…

4. Now we need to find similarity measures by using All previous Methods .

- Inner Product (Dot Product)

$$\sum_{k=1}^{t}(d_{ik} \bullet q_k)$$

- SC (Q,D1)=(0)(0) + (0)(0) + (0)(0.477) + (0)(0) + (0)(0.477) + (0.176)(0.176) + (0)(0) + (0)(0)=0.0309
- SC (Q,D2)=0.338
- SC (Q,D3)=0.061

Vector Model Example Cont…

Document ID	Similarity score
D_1	0.0309
D_2	0.338
D_3	0.061

5. Now, we rank the documents in a descending order according to their similarity score, as the following:

Document ID	Similarity score
D_2	0.338 (most relevant)
D_3	0.061
D_1	0.0309

We can use threshold to retrieve documents above the value of that threshold .

Vector Model Example Cont…

- Cosine

$$\frac{\sum_{k=1}^{t}(d_{ik} \bullet q_k)}{\sqrt{\sum_{k=1}^{t} d_{ik}^{2} \bullet \sum_{k=1}^{t} q_k^{2}}}$$

Where , $\sum_{k=1}^{t}(d_{ik} \bullet q_k)$ the inner product and $\sqrt{\sum_{k=1}^{t} d_{ik}^{2} \bullet \sum_{k=1}^{t} q_k^{2}}$ is the length for documents and query.

Note : When using the cosine similarity measure, computing the *tf-idf* values for the query terms we divide the frequency by the maximum frequency (2) and multiply with the *idf* values. So the query will be:

	T1	T2	T3	T4	T5	T6	T7	t8
Q	0	0	0	0	½*0.176= **0.088**	2/2*0.477= **0.477**	0	½*0.176= **0.088**

Vector Model Example Cont…

- For cosine method we must calculate the length of each documents and the length of query as the following:
  - Length of D1 = sqrt(0.477^2+0.477^2+0.176^2+0.176^2)= ***0.7195***
  - Length of D2 = sqrt(0.176^2+0.477^2+0.954^2+0.176^2)= ***1.095***
  - Length of D3 = sqrt(0.176^2+0.176^2+0.176^2+0.176^2)= ***0.352***
  - Length of Q = sqrt(0.088^2+0.477^2+0.088^2)= ***0.538***
- Inner product for each document is:
  - D1= 0.0309
  - D2=0.338
  - D3=0.061

Vector Model Example Cont…

- Then the similarity values are:
  - cosSim(D1,Q) = 0.0309 / 0.719 * 0.538 = **0.0799**
  - cosSim(D2,Q) = 0.338 / 1.095 * 0.538 = **0.574**
  - cosSim(D3,Q) = 0.061 / 0.352 * 0.538 = **0.322**
- Now, we rank the documents in a descending order according to their similarity score, as the following

Document ID	**Similarity score**
D_2	**0.574 (most relevant)**
D_3	**0.322**
D_1	**0.0799**

Vector Model Example Cont…

- Jaccard

$$\frac{\sum_{k=1}^{t}(d_{ik} \bullet q_k)}{\sum_{k=1}^{t} d_{ik}^{2} + \sum_{k=1}^{t} q_k^{2} - \sum_{k=1}^{t}(d_{ik} \bullet q_k)}$$

Where $\sum_{k=1}^{t}(d_{ik} \bullet q_k)$ is the inner product and $\sum_{k=1}^{t} d_{ik}^{2}$ is the square of summation of the weight of each term in each document and $\sum_{k=1}^{t} q_k^{2}$ which is the summation of the weight of each term in the query

Vector Model Example Cont…

- Inner product for each document is:
  - D1= 0.0309
  - D2=0.338
  - D3=0.061
- The square of the summation of the weight of each term in each document is:
  - $\sum_{k=1}^{8} D_{1,k}^{\;2}$ = (0.477^2+0.477^2+0.176^2+0.176^2)= **0.517**
  - $\sum_{k=1}^{8} D_{2,k}^{\;2}$ = (0.176^2+0.477^2+0.954^2+0.176^2)= **1.987**
  - $\sum_{k=1}^{8} D_{3,k}^{\;2}$ = (0.176^2+0.176^2+0.176^2+0.176^2)= **0.124**
- square of the summation of the weight of each term in the query is :

$\sum_{k=1}^{8} q_k^{\;2}$ = (0.176^2+0.477^2+0.176^2)= **0.289**

Vector Model Example Cont…

- Then the similarity values are:
  - JacSim(D1,Q) = 0.0309 / 0.517 + 0.289 - 0.0309 = **0.039**
  - JacSim(D2,Q) = 0.338 / 1.095 + 0.289 - 0.338 = **0.323**
  - JacSim(D3,Q) = 0.061 / 0.352 + 0.289 - 0.061 = **0.105**
  - Now, we rank the documents in a descending order according to their similarity score, as the following

Document ID	**Similarity score**
D_2	**0.323 (most relevant)**
D_3	**0.105**
D_1	**0.039**

Calculating the Jaccard Similarity Measure using Normalized Frequency (*fi,j*)

- Calculating the Jaccard similarity using normalized frequency will follow the previous step except that the maximum frequency will be determined and the weight equation will be different for both documents and query, it becomes:

$$w_{i,j} = f_{i,j} \times \log_{10} \frac{N}{n_i}$$

Where the ***Normalized Frequency*** (***fi,j***) is denoted by the following equation:

$$f_{i,j} = \frac{freq_{i,j}}{Max.\, freq_{i,j}}$$

Where ***freqi,j*** is the number of times the term ki is mentioned in the document, and **Max. *freqi,j*** is computed over all terms which are mentioned in the document.

Vector Model Example Cont…

- First we must determined the maximum frequency for each document as follow :

Document ID	**Max. Frequency**
D_1	1
D_2	2
D_3	1

Vector Model Example Cont…

Now we are going to compute the *Normalized Frequency* (*fi,j*) and the term weights (wi,j), based on the equations specified early as the following:

Term ID	**Document ID**	***Tf freg*** $_{i,}$	$f_{i,j} = \frac{freq_{i,j}}{Max.\ freq_{i,j}}$	$Idf_i = \log(N/n_i)$	$w_{i,j} = f_{i,j} \times \log \frac{N}{n_i}$
t1	D_2	1	½= 0.5	log10 3/2=0.176	0.088
t1	D_3	1	1/1= 1	log10 3/2=0.176	0.176
t2	D_1	1	1/1= 1	0.477	0.477
t3	D_2	1	½= 0.5	0.477	0.2385
t4	D1	1	1/1= 1	0.477	0.477
t5	D_1	1	1/1= 1	0.176	0.176
t5	D_3	1	1/1= 1	0.176	0.176
t6	D_2	2	2/2=1	0.477	0.477
t7	D_1	1	1/1= 1	0.176	0.176
t7	D_3	1	1/1= 1	0.176	0.176
t8	D_2	1	½= 0.5	0.176	0.088
t8	D_3	1	1/1= 1	0.176	0.176

Vector Model Example Cont…

- Now we can compute the sparse matrix after mapping the term weights.
- In order to compute the term weight of the ***query*** we going to use the Salton equation, denoted by:

$$w_{i,j} = \left(0.5 \times \frac{0.5\, freq_{i,j}}{\max_l freq_{i,j}}\right) \times \log \frac{N}{n_i}$$

- Finally ,we can compute the Jaccard similarity as we do in previous steps

Advantages & Disadvantages

- Advantages
  - Its term weighting scheme can improve retrieval performance
  - Allows partial matching
  - Retrieved documents are sorted according to their degree of similarity
- Disadvantages
  - Terms are assumed to be mutually independent. In some cases this might hurt performance.
  - Need whole doc set to determine weights
  - Extra computation