

Retrieval Strategies: Vector Space Model and Boolean

(CS429)
Nazli Goharian
nazli@ir.iit.edu

Slides are *mostly* based on Information Retrieval Algorithms and Heuristics, Grossman, Frieder

© Goharian, Grossman, Frieder 2002, 2009

Retrieval Strategy

- An IR *strategy* is a technique by which a relevance measure is obtained between a query and a document.

© Goharian, Grossman, Frieder 2002, 2009

Retrieval Strategies

- Manual Systems
 - Boolean, Fuzzy Set
- Automatic Systems
 - Vector Space Model
 - Language Models
 - Latent Semantic Indexing
- Adaptive
 - Probabilistic, Genetic Algorithms , Neural Networks, Inference Networks

© Goharian, Grossman, Frieder 2002, 2009

Vector Space Model

- Most commonly used strategy is the vector space model (proposed by Salton in 1975)
- Idea: Meaning of a document is conveyed by the words used in that document.
- Documents and queries are mapped into term vector space.
- Each dimension represents tf-idf for one term.
- Documents are ranked by closeness to the query. Closeness is determined by a similarity score calculation.

© Goharian, Grossman, Frieder 2002, 2009

Document and query presentation in VSM (Example)

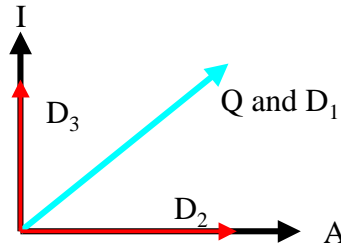
- Consider a two term vocabulary, A and I

Query: A I

D_1 - A I

D_2 - A

D_3 - I



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

© Goharian, Grossman, Frieder 2002, 2009

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.

$(idf_i) = \log_{10}(n / df_i)$, where n is the number of document

© Goharian, Grossman, Frieder 2002, 2009

Weights for Term Components

- Classic thing to do is use $tf \times idf$
- Incorporate idf in the query and the document, one or the other or neither.
- Scale the idf with a log or even a double log.
- Scale the tf ($\log tf+1$) or ($tf/\text{sum } tf$ of all terms in that document)
- Augment the weight with some constant (e.g.; $w = (w)(0.5)$)

© Goharian, Grossman, Frieder 2002, 2009

Weights for Term Components

- Many variations of term weight exist as the result of improving on basic $tf-idf$
- A good one:

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

- Some efforts suggest using different weighting for document terms and query terms. (Example: *Inc.ltc – see book if interested!*)

© Goharian, Grossman, Frieder 2002, 2009

Similarity Measures

- Similarity Coefficient (SC) identifies the Similarity between query Q and document D_i
 - Inner Product (dot Product)
 - Cosine
 - Dice
 - Jaccard

© Goharian, Grossman, Frieder 2002, 2009

Similarity Measures: (Inner Product)

- Inner Product (dot product)

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

- Problem: Longer documents will score very high because they have more chances to match query words.

© Goharian, Grossman, Frieder 2002, 2009

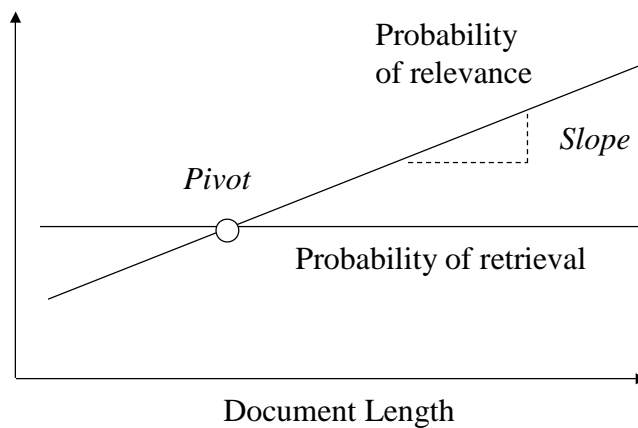
Similarity Measures: (Cosine)

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

- Assumption: document length has no impact on the relevance.
- Normalizes the weight by considering document length.
- Problem: Longer documents are somewhat penalized because indeed they might have more components that are indeed relevant [Singhal, 1997- Trec]

© Goharian, Grossman, Frieder 2002, 2009

Pivoted Cosine Normalization



© Goharian, Grossman, Frieder 2002, 2009

Pivoted Cosine Normalization

- Comparing likelihood of retrieval and relevance in a collection to identify *pivot* and thus, identify the new *correction factor*.

$$SC(Q, D_i) = \frac{\sum_{j=1}^t w_{qj} d_{ij}}{(1.0 - s) + (s) \frac{\sqrt{\sum_{j=1}^t (d_{ij})^2}}{avgn}}$$

Avgn: average document normalization factor over entire collection prior to any correction

© Goharian, Grossman, Frieder 2002, 2009

Pivoted Unique Normalization

- *Pivoted Cosine Normalization* worked well for short and moderately long documents.
- Problem: Extremely long documents are favored.
- *Pivoted Unique Normalization* (10% improvement in Trec queries).

© Goharian, Grossman, Frieder 2002, 2009

Pivoted Unique Normalization

$$SC(Q, D_i) = \frac{\sum_{j=1}^t w_{qj} d_{ij}}{(1.0 - s)p + (s)(|d_i|)}$$

$d_{ij} = (1 + \log(tf))idf / (1 + \log(atf))$ where, atf is average tf
 $|d_i|$: number of unique terms in a document.

p : average number of unique terms in a document for a collection

s : can be obtained via training (experimental determined)

© Goharian, Grossman, Frieder 2002, 2009

Similarity Measures (Cont'd)

- Dice

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

- Jaccard

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

© Goharian, Grossman, Frieder 2002, 2009

VSM Example

- Q: “gold silver truck”
- 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”

Id	Term	df	idf
1	a	3	0
2	arrived	2	0.176
3	damaged	1	0.477
4	delivery	1	0.477
5	fire	1	0.477
6	gold	2	0.176
7	in	3	0
8	of	3	0
9	silver	1	0.477
10	shipment	2	0.176
11	truck	2	0.176

© Goharian, Grossman, Frieder 2002, 2009

VSM Example

doc	t ₁	t ₂	t ₃	t ₄	t ₅	t ₆	t ₇	t ₈	t ₉	t ₁₀	t ₁₁
D ₁	0	0	.477	0	.477	.176	0	0	0	.176	0
D ₂	0	.176	0	.477	0	0	0	0	.954	0	.176
D ₃	0	.176	0	0	0	.176	0	0	0	.176	.176
Q	0	0	0	0	0	.176	0	0	.477	0	.176

- Computing SC using inner product:
- $SC(Q, D_1) = (0)(0) + (0)(0) + (0)(0.477) + (0)(0) + (0)(0.477) + (0.176)(0.176) + (0)(0) + (0)(0)$

© Goharian, Grossman, Frieder 2002, 2009

Algorithm for Vector Space (dot product)

- Assume: $t.\text{idf}$ gives the idf of any term t
- $q.\text{tf}$ gives the tf of any query term

Begin

Score[] \leftarrow 0

For each term t in Query Q

 Obtain posting list l

 For each entry p in l

 Score[p.docid] = Score[p.docid] + (p.tf * t.idf)(q.tf * t.idf)

- Now we have a SCORE array that is unsorted.
- Sort the score array and display top x results.

© Goharian, Grossman, Frieder 2002, 2009

Summary: Vector Space Model

- Pros
 - Fairly cheap to compute
 - Yields decent effectiveness
 - Very popular
- Cons
 - No theoretical foundation
 - Weights in the vectors are arbitrary
 - Assumes term independence

© Goharian, Grossman, Frieder 2002, 2009

Boolean Retrieval

- For many years, most commercial systems were only Boolean.
- Most old library systems and Lexis/Nexis have a long history of Boolean retrieval.
- Users who are experts at a complex query language can find what they are looking for.
(t1 AND t2) OR (t3 AND t7) WITHIN 2 Sentences
(t4 AND t5) NOT (t9 OR t10)
- Considers each document as bag of words

© Goharian, Grossman, Frieder 2002, 2009

Boolean Retrieval

- *Expression*:=
 - term
 - (*expr*)
 - NOT *expr* (*not recommended*)
 - *expr* AND *expr*
 - *expr* OR *expr*
- (cost OR price) AND paper AND NOT article

© Goharian, Grossman, Frieder 2002, 2009

Boolean Example

<i>doc</i>	<i>t</i> ₁	<i>t</i> ₂	<i>t</i> ₃	<i>t</i> ₄	<i>t</i> ₅	<i>t</i> ₆	<i>t</i> ₇	<i>t</i> ₈	<i>t</i> ₉	<i>t</i> ₁₀	<i>t</i> ₁₁
<i>D</i> ₁	0	0	1	0	1	1	0	0	0	1	0
<i>D</i> ₂	1	1	0	1	0	0	0	0	1	0	1
<i>D</i> ₃	1	1	0	0	0	1	0	0	0	1	1
<i>D</i> ₄	0	0	0	0	0	1	0	0	1	0	1

Q: t1 AND t2 AND NOT t4

0110 AND 0110 AND 1011 = 0010 That is D3

© Goharian, Grossman, Frieder 2002, 2009

Processing Boolean Queries

- Doc-term matrix is too sparse, thus, using inverted index
- Query optimization in Boolean retrieval:
The order in which posting lists are accessed!

© Goharian, Grossman, Frieder 2002, 2009

Processing Boolean Query

t_1 AND t_2

- Algorithm:

Find t_1 in index (lexicon)

Retrieve its posting list

Find t_2 in index (lexicon)

Retrieve its posting list

Intersect (merge) the posting lists

The matching DocIDs are added to the result list

© Goharian, Grossman, Frieder 2002, 2009

Processing Boolean Query

t_1 AND t_2 AND t_3

- What is the best order to process this?
- Process in the order of increasing document frequency, i.e, smaller Posting Lists first!
- Thus, if t_1 , t_2 have smaller PL than t_3 , then process as:

$(t_1 \text{ AND } t_2) \text{ AND } t_3$

© Goharian, Grossman, Frieder 2002, 2009

Intersection of Posting Lists

Algorithm

Sort query terms based on document frequency

Merge the smallest posting list with the next smallest posting list and create the result set

Merge the next smaller posting list with the result set, update the result set

Continue till no more terms left

© Goharian, Grossman, Frieder 2002, 2009

Processing Boolean Query

$(t1 \text{ OR } t2) \text{ AND } (t3 \text{ OR } t4) \text{ AND } (t5 \text{ OR } t6)$

- Using document frequency estimate the size of disjuncts
- Order the conjuncts in order of smaller disjuncts

© Goharian, Grossman, Frieder 2002, 2009

Extended (Weighted) Boolean Retrieval

- Boolean OR (t1 OR t2)

$$SC(Q_{q_1 \vee q_2}, D_i) = \frac{\sqrt{w_{q_1}^2 d_1^2 + w_{q_2}^2 d_2^2}}{\sqrt{w_{q_1}^2 + w_{q_2}^2}}$$

- Boolean AND (t1 AND t2)

$$SC(Q_{q_1 \wedge q_2}, D_i) = 1 - \frac{\sqrt{w_{q_1}^2 (1-d_1)^2 + w_{q_2}^2 (1-d_2)^2}}{\sqrt{w_{q_1}^2 + w_{q_2}^2}}$$

© Goharian, Grossman, Frieder 2002, 2009

Extended (Weighted) Boolean Retrieval

- Ranking by term frequency (Sony Search Engine)

x AND y: $tf_x \times tf_y$

x OR y: $tf_x + tf_y$

NOT x: 0 if $tf_x > 0$, 1 if $tf_x = 0$

- User may assign term weights

cost and +paper

© Goharian, Grossman, Frieder 2002, 2009

Fuzzy Sets

- x OR y: $\max(w_x, w_y)$
- x AND y: $\min(w_x, w_y)$
- NOT x: $1 - w_x$
- Example:
 - $D_1 = \{(\text{cost}, 0.2), (\text{paper}, 0.3)\}$
 - “cost AND paper” scores D_1 at 0.2

© Goharian, Grossman, Frieder 2002, 2009

Summary of Boolean Retrieval

- Pro
 - Can use very restrictive search
 - Makes experienced users happy
- Con
 - Simple queries do not work well.
 - Complex query language, confusing to end users

© Goharian, Grossman, Frieder 2002, 2009