5. Vector Space Model

Outline

Overview

- Most frequently used model
- Documents and query represented by t-dimensional vectors
- t is number of terms of the indexing vocabulary
- i-th vector component is term weight (non-negative real number)
- High weight $\Rightarrow$ corresponding term represents document/query well

5.1 The Basic Idea

- Let N be number of documents
- Let n_k be the number of documents containing term k
- Let tf_{dk} be the frequency of occurrence of term k in document D
- Document D is represented by vector $V_D = (w_{d1}; ...; w_{dt})$
- Analog: query Q by $V_Q = (w_{q1}; ...; w_{qt})$
- w_{qk} and w_{dk} are relevance of document D and query Q for term k, respectively
- Similarity calculation using scalar product: $\langle V_Q, V_D \rangle$

$$similarity(V_Q, V_D) = \langle V_Q, V_D \rangle = \sum_{k=1}^{t} w_{qk} w_{dk}$$

Basic assumption

- Vector dimensions are independent of each other
- Cluster hypothesis
  - documents and query lie compactly together in a high-dimensional vector space (similar documents show similar directions)

Example

- Selected index terms: 'Houses', 'Italy', 'Gardens', 'France'
- First approach: $w_{dk} = tf_{dk}$
  (no consideration of document lengths)
- D_1 = 'Houses in Italy' $\Rightarrow V_{D1} = (1; 1; 0; 0)$
- D_2 = 'Houses in Italy and around Italy' $\Rightarrow V_{D2} = (1; 2; 0; 0)$
- D_3 = 'Gardens and Houses in Italy' $\Rightarrow V_{D3} = (1; 1; 1; 0)$
- D_4 = 'Gardens in Italy' $\Rightarrow V_{D4} = (0; 1; 1; 0)$
- D_5 = 'Gardens and Houses in France' $\Rightarrow V_{D5} = (1; 0; 1; 1)$
- Q = 'Houses in Italy' $\Rightarrow V_Q = (1; 1; 0; 0)$
- Similarity(V_Q, V_{D1})=2, Similarity(V_Q, V_{D2})=3,
  Similarity(V_Q, V_{D3})=2, Similarity(V_Q, V_{D4})=1,
  Similarity(V_Q, V_{D5})=1

5.2 The Basic Formulas of the Vector Space Model

Calculation of term weights w_{dk}

- Many very similar formulas available - here the most common formula
- 2 factors
  1. frequency of occurrence of a term in the document
  2. discriminatory power of a term related to document collection

Frequency of occurrence of a term in a document

- The higher the frequency of occurrence tf_{dk}, the better the term describes the document
- Problem: absolute frequency of occurrence tends to be higher for long documents
- Consequence: long documents are preferred during search
- Example: $V_{D1} = (50; 5; 0; 0)$, $V_{D2} = (0; 2; 2; 0)$, $V_Q = (0; 1; 1; 0)$
  $\Rightarrow V_{D1}$ is unfairly preferred:
  $\langle V_Q, V_{D1} \rangle = 5$ and $\langle V_Q, V_{D2} \rangle = 4$

Frequency of occurrence of a term in a document

Independence from document length by normalization of document vectors:

$$w_{dk} = \frac{tf_{dk}}{\sqrt{\sum_{i=1}^{t} tf_{di}^2}}$$

Example

- Absolute frequency:
  $V_{D1} = (50; 5; 0; 0)$, $V_{D2} = (0; 2; 2; 0)$, $V_Q = (0; 1; 1; 0)$
- Relative frequency by vector normalization:
  $V_{D1} = (0.995; 0.0995; 0; 0)$, $V_{D2} = (0; 0.707; 0.707; 0)$
- Query vector unchanged: $V_Q = (0; 1; 1; 0)$
  $\Rightarrow \langle V_Q, V_{D1} \rangle = 0.0995$ and $\langle V_Q, V_{D2} \rangle = 1.41$
- Before: $\langle V_Q, V_{D1} \rangle = 5$ and $\langle V_Q, V_{D2} \rangle = 4$

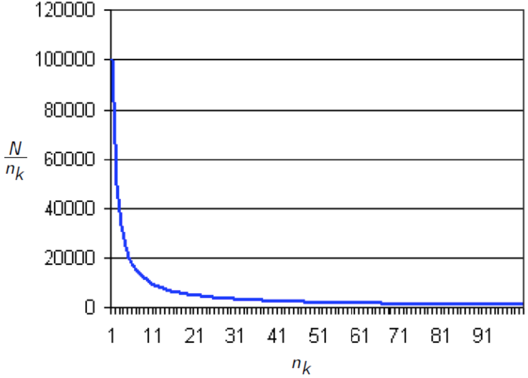
Discriminatory power of a term

- If term occurs in fewer documents, it is more specific
- More specific terms have higher discriminatory power and should be weighted higher
- Approach: discriminatory power as quotient: $\frac{N}{n_i}$

$$w_{dk} = \frac{tf_{dk} \cdot \frac{N}{n_k}}{\sqrt{\sum_{i=1}^{t} (tf_{di} \frac{N}{n_i})^2}}$$

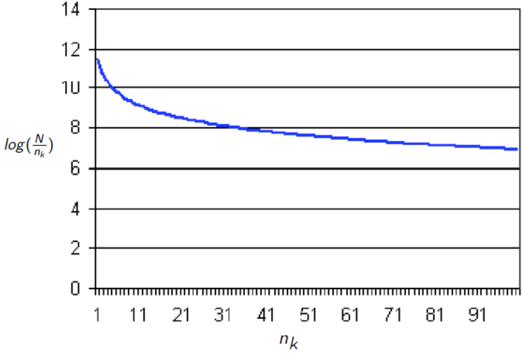
Discriminatory power as a quotient

- Problem with $\frac{N}{n_i}$: very rare terms dominate vectors
- Example: $N = 100000$, $n_i = 5$ and $n_j = 50$



Discriminatory power using logarithm

- Natural logarithm mitigates effect
- $log(\frac{N}{n_k})$ instead of $\frac{N}{n_k}$



The $tf \cdot idf$ formula

$$w_{dk} = \frac{tf_{dk} \cdot log\frac{N}{n_k}}{\sqrt{\sum_{i=1}^{t}(tf_{di} \cdot log\frac{N}{n_i})^2}}$$

- *tf* stands for term frequency
- *idf* stands for inverse document frequency
- $tf \cdot idf$ formula is a heuristic formula

Query representation

- Query Q represented by $V_Q = (w_{q1}; \ldots; w_{qt})$
- Do not determine W_{qk} values using $tf \cdot idf$ formula, since
  - each term in query has its own base weight 0.5
  - normalization not necessary, since factor is the same for all documents
- Proposal by Salton and Buckley:

$$w_{qk} = \begin{cases} \left(0.5 + \frac{0.5 \cdot tf_{qk}}{\max\limits_{1 \le i \le t} tf_{qi}}\right) \cdot \log \frac{N}{n_k} & \text{if } tf_{qk} > 0 \\ 0 & \text{if } tf_{qk} = 0 \end{cases}$$

Query representation

Base weight, as term was explicitly selected by user

How often does the term occur in the query?

Discriminatory power

$$w_{qk} = \begin{cases} \left(0.5 + \dfrac{0.5 \cdot tf_{qk}}{\max\limits_{1 \leq i \leq t} tf_{qi}}\right) \cdot log \dfrac{N}{n_k} & \text{if } tf_{qk} > 0 \\ 0 & \text{if } tf_{qk} = 0 \end{cases}$$

Similarity between documents and query

- Calculation via scalar product:

$$similarity(V_Q, V_D) = \langle V_Q, V_D \rangle = \sum_{k=1}^{t} w_{qk} w_{dk}$$

- Law from linear algebra:

$$\langle V_Q, V_D \rangle = \sqrt{\langle V_Q, V_Q \rangle} \cdot \sqrt{\langle V_D, V_D \rangle} \cdot cos\alpha$$

- Cosine measure is cosine of the angle enclosed by V_Q and V_D:

$$sim_{cos}(V_Q, V_D) = \frac{\sum_{k=1}^{t} w_{qk} \cdot w_{dk}}{\sqrt{\sum_{i=1}^{t} w_{qi}^2} \cdot \sqrt{\sum_{j=1}^{t} w_{dj}^2}}$$

- Generated document order for cosine measure and scalar product is the same

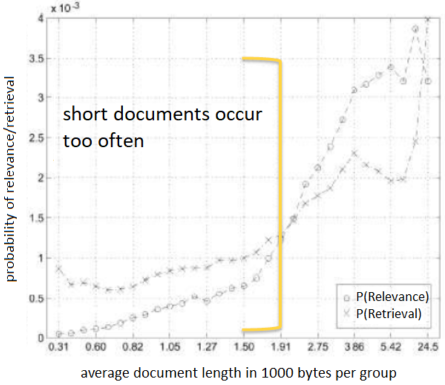
5.3 A Variant of the *tf* · *idf* Formula

- Many improvements of the *tf* · *idf* formula are proposed
- The following is one example of many variants

- Normalization of vectors should avoid disadvantage of short documents
- Side effect: short documents have few components with $w_{dk} > 0$
- Normalization of a few components generates relatively high values
- Consequence for short queries: preference for short documents
- Normalization is therefore “too much of a good thing”
  $\Rightarrow$ compensation

Test on concrete data set

CDs 1 and 2 of the TREC collection with 741856 documents and Topics 151 to 200



Normalization adjustment

Normalization factor

$$\sqrt{\sum_{i=1}^{t}(tf_{di} \cdot log\frac{N}{n_i})^2} = \sqrt{\sum_{i=1}^{t} w_{di}^2}$$

- Long documents should become stronger
- Short documents should become weaker

- Use $kf_d \cdot w_{dk}$ instead of w_{dk} when normalizing:

$$kf_d = (1 - slope) + slope \cdot \frac{old\_normalization}{average\_old\_normalization}$$

Example

- $slope = 0.75$ and $average\_old\_normalization = 40$

Document	Old normalization	kf_d	New normalization	Vector length
D_1	20	0,625	32	0,625
D_2	40	1,0	40	1,0
D_3	80	1,75	45,71	1,75

Determination of the slope value

- Must be determined empirically
- Example CDs 1 and 2 of TREC:
  $average\_old\_normalization = 13.36$
- Optimal is $slope = 0.75$

	$tf \cdot idf$	modified approach				
slope	-	0.60	0.65	0.70	0.75	0.80
found	6526	6342	6458	6574	6629	6671
precision	0.284	0.302	0.310	0.314	0.317	0.316
improvement	-	+6.5%	+9.0%	+10.7%	+11.7%	+11.3%

Summary: variant of *tf* · *idf* formula

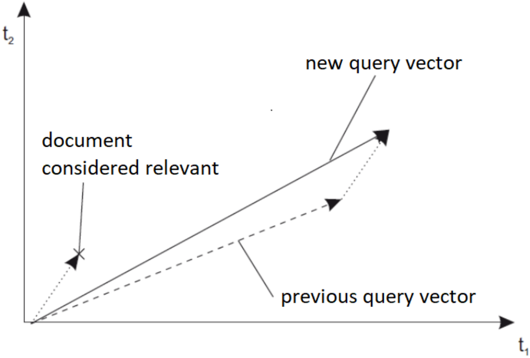
- Modified normalization: relevance and retrieval probability get closer $\Rightarrow$ higher precision
- Attention: $w_{df} \leq 1$ does no longer hold $\Rightarrow$ adaptation of algorithms that rely on normalization
- Slope value must be determined empirically

5.4 Relevance Feedback

- Vector space model expects information need from user as query
- Goal: user interactions as additional sources of information for clarification
- User evaluates individual results after first run (relevant, not relevant)
- Relevance feedback uses evaluation to reformulate new query
- Evaluations lead to shift of the query vector

Shifting the query vector

- 2 terms
- Changing the ‘search angle’ in the direction of the document evaluated as relevant



Query shifting methods

- Starting point is decomposition of the feedback set into
  - F^+ (documents considered relevant) and
  - F^- (documents considered irrelevant)
  - Possibly also $F^?$ (neither considered relevant nor irrelevant)
- Methods
  - Ide (dec hi)
  - Ide (regular)
  - Rocchio

Ide (dec hi)

- Documents from F^{+} are added
- D^{-rel} from F^{-} with highest rank in result is subtracted

$$V_Q^{new} = V_Q^{old} + (\sum_{D \in F^{+}} D) - D^{-rel}$$

Ide (regular)

- Documents from F^+ are added
- Documents from F^- are subtracted

$$V_Q^{new} = V_Q^{old} + (\sum_{D \in F^+} D) - (\sum_{D \in F^-} D)$$

Rocchio

- Documents from F^+ are averaged and added
- Documents from F^- are averaged and subtracted
- Influence of F^+ and F^- is weighted by α and β ($\alpha + \beta = 1$), respectively
- Typical assignment: $\alpha = 0.25$ and $\beta = 0.75$

$$V_Q^{new} = V_Q^{old} + \beta \cdot \frac{1}{|F^+|} \sum_{D \in F^+} D - \alpha \cdot \frac{1}{|F^-|} \cdot \sum_{D \in F^-} D$$

Experimental results

- Precision values as mean values over recall levels 0.25, 0.5 and 0.75
- Results under *selected* terms: only terms in many relevant documents are considered
  $\rightarrow$ advantage: accelerated query processing
- Significant improvements in precision values are achieved
- Ide (dec hi) performs best
- Reason: heterogeneous set F^- does not give a clear direction

Experimental results

method used		CACM 3204 Doc. 64 queries	CISI 1460 Doc. 112 queries	CRAN 1397 Doc. 225 queries	INSPEC 12684 Doc. 84 queries	MED 1033 Doc. 30 queries	average
initial query							
	Precision	0,1459	0,1184	0,1156	0,1368	0,3346	
Ide (dec hi)							
with all terms	Precision	0,2704	0,1742	0,3011	0,2140	0,6305	
	improvement	+86%	+47%	+160%	+56%	+88%	+87%
selected terms	Precision	0,2479	0,1924	0,2498	0,1976	0,6218	
	improvement	+70%	+63%	+116%	+44%	+86%	+76%
Ide (regular)							
with all terms	Precision	0,2241	0,1550	0,2508	0,1936	0,6228	
	improvement	+66%	+31%	+117%	+42%	+86%	+68%
selected terms	Precision	0,2179	0,1704	0,2217	0,1808	0,5980	
	improvement	+49%	+44%	+92%	+32%	+79%	+59%
Rocchio (standard $\beta = 0,75$; $\alpha = 0,25$)							
with all terms	Precision	0,2552	0,1404	0,2955	0,1821	0,5630	
	improvement	+75%	+19%	+156%	+33%	+68%	+70%
selected terms	Precision	0,2491	0,1623	0,2534	0,1861	0,5297	
	improvement	+71%	+37%	+119%	+36%	+55%	+64%

Pseudo relevance feedback

- First elements of the ranking are used for relevance feedback
  - query expansion improves recall
  - works well only if top n precision is already good

5.5 Implementation of the Vector Space Model

Variants:

- Inverted lists (standard approach)
- Multidimensional access structures (high dimensionality problem)
- Signatures

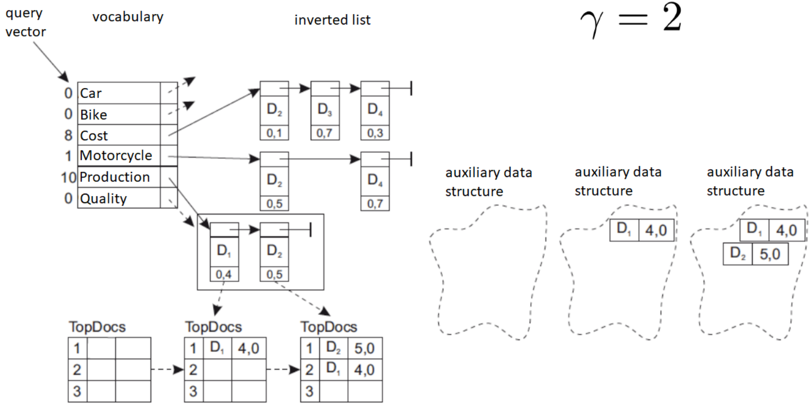
Basic algorithms with inverted lists

- Querier usually wants to see only the γ best results
- Steps:
  1. run through the inverted lists of the query terms in order of falling query term weights w_{qk}
  2. for each pair (D, w_{dk}) in the inverted list just considered, add $w_{dk} \cdot w_{qk}$ to the previous weight of D
  3. after fully considering an inverted list, check if document with rank $\gamma + 1$ can still reach rank γ
     if not, then abort

Example

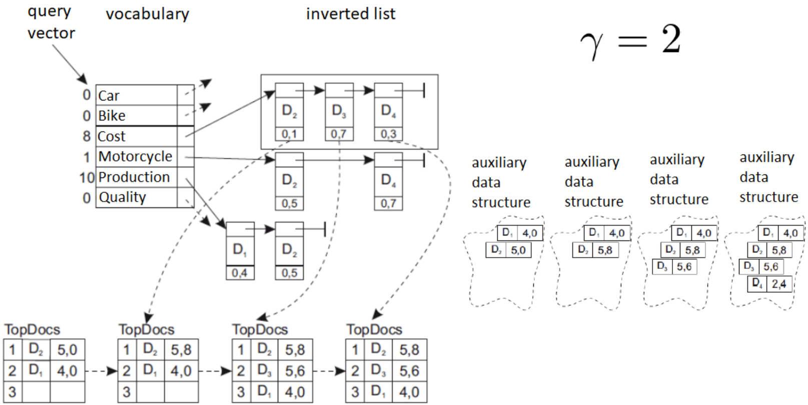
- Vocabulary {Car, Bike, Cost, Motorcycle, Production, Quality}
- Query vector (0; 0; 8; 1; 10; 0)
- $\gamma + 1$ currently best documents are kept in the sorted array *TopDocs*
- Additionally, each read document in auxiliary data structure
  - grows enormously under certain circumstances!

Example II



$$\gamma = 2$$

Example III



$$\gamma = 2$$

Realization of the auxiliary data structure

- Management of pairs (D, w), i.e. document ID and weight
- Operations
  - $IsInDS(D)$ tests whether D is contained
  - $InsertIntoDS(D, w)$ inserts new pair
  - $AddToDSEntry(D, w)$ adds weight w to the document
  - $GetWeightFromDS(D)$ determines weight

Necessary operations on inverted list

Operations on term k:

- *InitList*(k) initializes list for sequential iteration
- *GetNextElementOfList*(k) returns current pair (D, w_{dk})
- *NotAtEndOfList*(k) tests if end is not reached yet

Realization of the *TopDocs* array

- Contains $\gamma + 1$ documents
- Seen document is inserted at correct position (in sorted list)

The *MaxRemainingWeight*() operation

- Determines maximum possible gain of weight
- Assumes $w_{dk} \leq 1$

$$MaxRemainingWeight() = \sum_{\text{all remaining query terms } k'} w_{qk'}$$

Algorithm according to Buckley & Lewit

- Returns γ most relevant documents
  (but not necessarily in correct order)

```
for ( all terms in the query vector with w_qk > 0) do
    Let k' be the previously not considered term with the highest value w_qk';
    InitList(k');
    while (NotAtEndOfList(k')) do
        (D, w_dk') = GetNextElemOfList(k');
        if IsInDS(D) then
            AddToDSEntry(D, w_qk' · w_dk');
        else
            InsertIntoDS(D, w_qk' · w_dk');
        end;
        Sort D in the TopDocs array according to the weight
        GetWeightFromDS(D);
    end;
    if (TopDocs[γ] > TopDocs[γ+1] + MaxRemainingWeight())
        then FINISH;
end;
```

Performance increase

- Idea: let algorithm terminate earlier and accept inaccuracy
- Buckley & Lewit: termination when $n \leq \gamma$ documents have been reliably detected

```
if (TopDocs[n] > TopDocs[γ+1] + MaxRemainingWeight()) then
    FINISH;
```

Experiments with $\gamma = 10$

n	CPU time	# blocks read	Recall
1	61.3	2833	0.1588
2	68.2	3058	0.1566
3	75.9	3255	0.1555
4	83.1	3451	0.1538
5	90.3	3630	0.1548
6	93.6	3721	0.1543
7	99.7	3863	0.1501
8	104.5	3980	0.1505
9	107.0	4033	0.1508
10	109.2	41	0.1508

better result despite early termination

baseline

Evaluation of the experiments

- The smaller *n* the better the recall value
- Possible reason:
  - highly weighted terms (high discriminatory power) are taken into account whereas for long queries the remaining terms 'dilute' the result
- Conclusion: exact calculation does not necessarily lead to better results

Evaluation vector space model

- Advantages:
  - simplicity
  - directly applicable to document collection in contrast to probabilistic model
  - very good empirical retrieval quality
  - efficient implementation as inverted lists
  - relevance feedback improves quality of results
- Disadvantages:
  - assumes independence of dimensions
  - no theoretical foundation of the model
  - adaptation to structured documents only possible with difficulty