CS 533 – Information Retrieval Systems

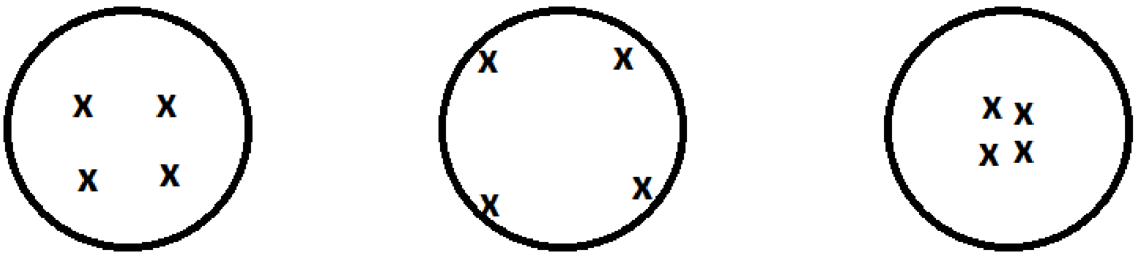
Class Notes (07.04.2014 – 09.04.2014)

prepared by Mustafa Can Çavdar

Term Discrimination Value (TDV)

Assign more weights to terms that make documents more distinguishable from each other.

Example:



additional term that makes docs. more distinguishable (Good discriminator)

additional term that makes docs. less distinguishable (Bad discriminator)

How to calculate TDVs?

1. Using similarity values [1]

$$S = \begin{bmatrix} - & S12 & S13 & S14 \\ - & - & S23 & S24 \\ - & - & - & S34 \\ - & - & - & - \end{bmatrix}$$

Find average similarity among documents => Space density Q

$$Q = \frac{1}{m(m-1)/_2} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} Sij$$

Q_j: average similarity without term t_j

Term Type	Q: Q_j	TDVs
Good Discrimination	<	>0
Bad Discrimination	>	<0
Indifferent Discrimination	=	= 0

$$TDV_j = Q_j - Q \qquad \text{OR} \qquad TDV_j = \frac{Q_j}{Q}$$

Approximate calculation of TDVs:

One possibility

Find the centroid of the collections.

$$D = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \qquad \text{Centroid C} = [2/3 \quad 1/3 \quad 2/3 \quad 2/3]$$

$$Q = \frac{1}{m} \sum_{i=1}^{m} Sim(di, c)$$

During the calculation of Q_j, ignore t_j.

Advantage: more efficient   Disadvantage: approximation

$$TDV_j = Q_j - Q \qquad \text{OR} \qquad TDV_j = \frac{Q_j}{Q}$$

2. Using cover coefficient concept [2]

Advantage: no approximation TDV_j

Space density concept is replaced by n_c.

More distinguishable docs => more no. of clusters.

Less distinguishable docs => lesser no. of clusters.

n_c: number of clusters with all terms
n_{cl}: number of clusters without term t_l.
If t_l is good discriminator => its assignment increases no. of clusters. ($n_c > n_{cl}$)
If t_l is bad => $n_c < n_{cl}$

TDV = n_c - n_{cl} OR TDV = n_c / n_{cl} => possibilities

$$n_c = \sum_{i=1}^{m} C_{ii} = \sum_{i=1}^{m} \delta_i = \sum_{i=1}^{m} \alpha_i * \left(d_{i1}^2 * \beta_1 + d_{i2}^2 * \beta_2 + \cdots + d_{in}^2 * \beta_n\right)$$

$$n_{cl} = \sum_{i=1}^{m} \delta_i^l = \sum_{i=1}^{m} \alpha_i^l * \left(d_{i1}^2 * \beta_1 + \cdots + d_{i(l-1)}^2 * \beta_{l-1} + d_{i(l+1)}^2 * \beta_{l+1} + \cdots + d_{in}^2 * \beta_n\right)$$

$$\alpha_i^l = \left(\alpha_i^{-1} - d_{il}\right)^{-1}$$

$$n_{cl} = \sum_{i=1}^{m} \alpha_i^l * \left(\frac{\delta_i}{\alpha_i} - d_{il}^2 * \beta_l\right)$$

Notice that not all documents contain t_l. For such documents;

$$\delta_i = \delta_i^l$$

Let's define the set of documents that contain t_l.

$$D_l = \{d_i |\, d_i \in D \text{ and } d_{il} \neq 0\}$$

$$f_l = |D_l|$$

$$n_{cl} = n_c + \sum_{i=1}^{f_l} \left[\alpha_i^2 * \left(\frac{\delta_i}{\alpha_i} - d_{il}^2 * \beta_l\right) - \delta_i\right]$$

$$TDV = n_c - n_{cl} = \sum_{i=1}^{f_l} \left[\delta_i - \alpha_i^l * \left(\frac{\delta_i}{\alpha_i} - d_{il}^2 * \beta_l\right)\right]$$

For a binary D it can be shown that;

$$TDV_l = \sum_{i=1}^{f_l} \alpha_i^l * (d_{il} * \beta_l - \delta_i)$$

More efficient calculation of TDVs using cc concept

$$n_c = \frac{m * n}{t}$$

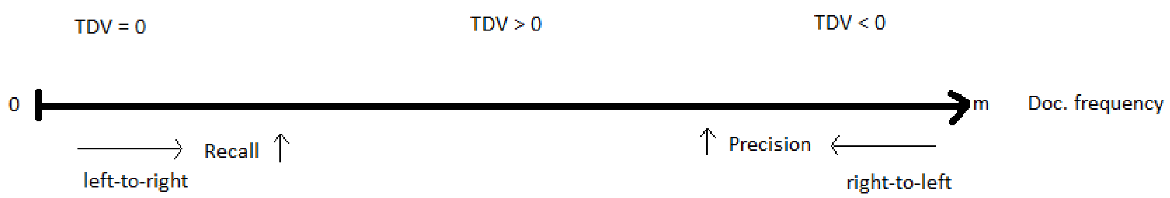
$$n_{cl} = \frac{m * (n - 1)}{(t - \#\, docs\ contains\ t_l)}$$

Consistency between cc-based and similarity based TDV calculations
tg: term generality

Term characteristics	Sim-based	Cc-based
Terms with high tg	TDV < 0	TDV < 0
Terms with medium tg	TDV > 0	TDV = 0
Terms with low tg	TDV = 0	TDV > 0

<u>How to use TDVs?</u>

1. To obtain a D matrix with less no. of terms. In cc based TDV values the terms with TDV = 0 do not change the no. of clusters. => Probably they also don't change the composition of the clusters.
2. To obtain better vocabularies;



<u>Left-to-right transformation:</u> (word grouping)

Consider a database related to education.

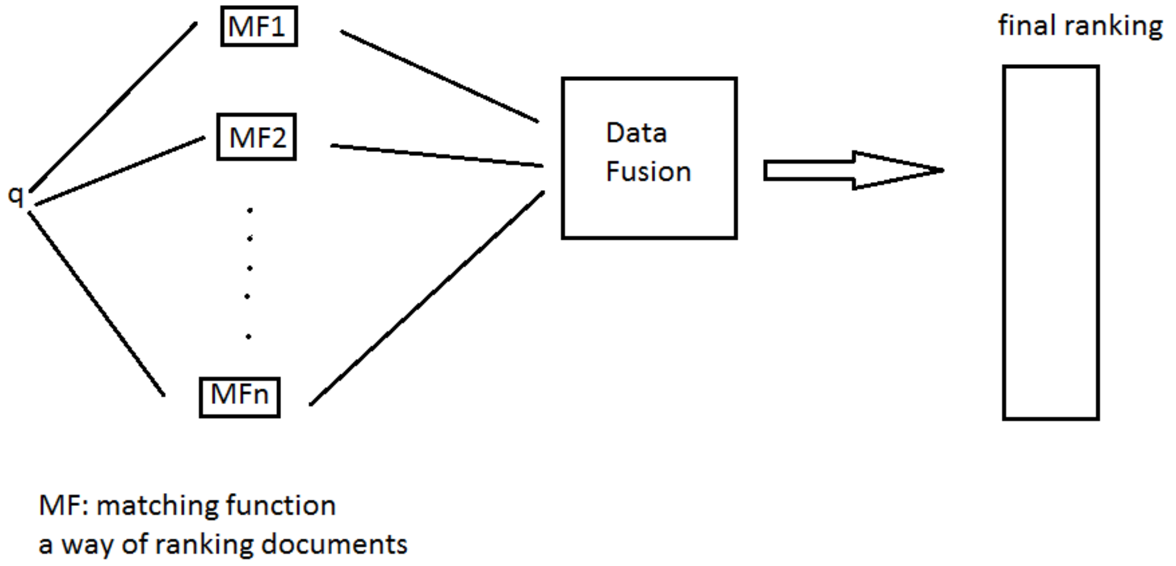
$$D = \begin{array}{c} \begin{array}{ccc} C & Pascal & Fortran \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$

C, Pascal, Fortran => programming language

$$D = \begin{array}{c} \text{Programming language} \\ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \end{array}$$

In a user query; C, Pascal and Fortran must be wrapped to Programming language.

<u>Data Fusion</u> [3]



1. Rank position(reciprocal rank) method

$$r(d_i) = \frac{1}{\sum(1/position(d_{ij}))}$$

i: document j: system

Example:

A, B, C, D: search engines
a, b, c, d, e, f, g: documents

A = (a, b, c, d)
B = (a, d, b, e)
C = (c, a, f, e)
D = (b, g, e, f)

$$r(a) = \frac{1}{\frac{1}{1} + \frac{1}{1} + \frac{1}{2}} = 0.4$$

$$r(b) = \frac{1}{\frac{1}{2} + \frac{1}{3} + 1} = 0.52$$

$$r(c) = \frac{1}{\frac{1}{3} + 1} = 0.75$$

a > b > c > d > e > f > g

2. Borda Count Method

Jean Charles de Borda (1733-1799)

The highest rank individual (in an n-way vote) gets n votes and each subsequent gets one vote less (so #2 get n-1 …).

If there are candidates left unmarked by the voter, the remaining points are divided evenly among the unranked candidates.

Example:

A = (a, c, b, d)
B = (b, c, a, e)
C = (c, a, b, e)

$$BC(a) = BC_A(a) + BC_B(a) + BC_C(a) = 5 + 3 + 4 = 12$$

$$BC(b) = BC_A(b) + BC_B(b) + BC_C(b) = 3 + 5 + 3 = 11$$

$$BC(c) = BC_A(c) + BC_B(c) + BC_C(c) = 4 + 4 + 5 = 13$$

c > a > b

Problem: deletion of a document may change the order drastically.

3. Condorcet Method

Majoritarian Method

The winner is the document which beats each of the other documents in a pairwise comparison

Example:

3 candidate documents: a, b, c

5 systems: A, B, C, D, E

A: a>b>c - B:a>c>b - C:a>b=c - D:b>a - E:c>a

We build a pairwise comparison matrix that shows how many times a document beats other one, is beated by the other one and they are tied. For example; in below comparison matrix;

PairwiseComp(a,b) = (4,1,0) show that doc a gets higher rank than doc b in 4 systems and doc b gets higher rank than doc a in 1 system.

At the end document having with the most number of wins gets the first rank and the other ones are sorted.

Pairwise comparison

	a	b	c
a	-	4, 1, 0	4, 1, 0
b	1, 4, 0	-	2, 2, 1
c	1, 4, 0	2, 2, 1	-

Pairwise winners

	Win	Lose	Tie
a	2	0	0
b	0	1	1
c	0	1	1

Final ranking of documents: a > b = c

Data Streams

Document data stream processing

Data Stream Mining

Online: dynamic. We have document in temporal order.

Archived: Google Book Collection

Application Areas

- Financial markets
- Health care
    - Vital signals
    - Patient monitoring
- Traffic monitoring
- Intelligence applications (CIA)

- Computational social sciences

Aim;

- Financial trends
- Life-threating developments
- Traffic congestion
- Terrorist activities
- Future social actors

Processed data;

- News articles
- Intelligence reports
- Scientific papers
- Emails, tweets

Within the context of IR, they are used for;

- Information filtering: sends newly arriving docs to the owners of the matching user profiles.
- Information aggregation: news portals, blog portals. (what is to be selected for the front page)
- Trend detection
- Sentence extraction based summarization

Diversity and Novelty Detection in Document Data Streams

Diversity => cover different aspects as much as possible

Novelty => cover new items of different aspects

TDT: topic detection and tracking

Workshop 1997-2004

First story detection

(First Story Detection is Hard by James Allan et al.)

Topic Tracking: given 2 to 4 sample stories, find other stories about them

✗ Earthquake

✓ Earthquake in 1999 (more specific)

Story link detection: given 2 stories, are they related?

Topic detection (cluster detection)

QUESTIONS

1. Use rank position method to rank documents with following results.
   A = (a, b, c)
   B = (d, c, a)
   C = (c, a, d)

$$r(a) = \frac{1}{\frac{1}{1} + \frac{1}{3} + \frac{1}{2}} = 0.55$$

$$r(b) = \frac{1}{\frac{1}{2}} = 2$$

$$r(c) = \frac{1}{\frac{1}{3} + \frac{1}{2} + 1} = 0.55$$

$$r(d) = \frac{1}{\frac{1}{1} + \frac{1}{3}} = 0.75$$

r (a) = r (c) < r (d) < r (b) => a = c > d > b in rankings

2. Use Borda Count Method to rank documents in the above question.

$$BC(a) = BC_A(a) + BC_B(a) + BC_C(a) = 4 + 2 + 3 = 9$$

$$BC(b) = BC_A(b) + BC_B(b) + BC_C(b) = 2 + 0 + 0 = 2$$

$$BC(c) = BC_A(c) + BC_B(c) + BC_C(c) = 2 + 3 + 4 = 9$$

$$BC(d) = BC_A(d) + BC_B(d) + BC_C(d) = 0 + 4 + 2 = 6$$

Therefore, ranking is a = c > d > b

3. Use Condorcet Method to rank documents in the first question.

Pairwise Comparison:

$-$	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{c}$	$\boldsymbol{d}$
$\boldsymbol{a}$	$-$	$3-0-0$	$2-1-0$	$2-1-0$
$\boldsymbol{b}$	$0-3-0$	$-$	$1-2-0$	$1-2-0$
$\boldsymbol{c}$	$1-2-0$	$2-1-0$	$-$	$2-1-0$
$\boldsymbol{d}$	$1-2-0$	$2-1-0$	$1-2-0$	$-$

Pairwise Winners:

$-$	Win	$Lose$	Tie
a	3	0	0
b	0	3	0
c	2	1	0
d	1	2	0

Final ranking: a>c>d>b

4. In which cases *word grouping* is useful?

If terms are rare ones as in the example above which were *C*, *Pascal* and *FORTRAN*, we must wrap them to a more common term as *programming language* in the example to increase recall.

5. What is a *matching function* and what it is used for?

A matching function basically is a way of ranking documents for a given query. Documents are ranked according to different number of functions and after that the results are combined with data fusion process and we get final ranking of documents.

REFERENCES

1. Salton, G., A Theory of Indexing

2. Can, Ozkarahan, Computation of Term/document Discrimination Values by Use of the Cover Coefficient Concept

3. Nuray, Can, Automatic Ranking of Information Retrieval Systems