Bag of words

↳ collection of terms with frequencies

↳ can contain duplicates

↳ doesnt consider order

Bag of Words

- Example
  - D1: {John likes to watch movies. Mary likes movies too. }
  - D2: {John also likes to watch football games.}
  - D3: { Mary does not like football games }
- Bow Representation
  - D1: {John, like,watch,movie,Mary,like,movie}
  - D2: {John,like,watch,football,game}
  - D3: { Mary,like,football,game }
- Query Term
  - Q: {like football game}

Bag of Words

- Scores
  - Common Terms in document and query over total terms
  - (D1,Q) = (D1 ∩ Q) / (D1 U Q) = 1/8
  - (D2,Q)= 3/5
  - (D3,Q)= 3/4
- Issues
  - Document size affects the overall similarity.
  - Frequency of common terms are ignored.
  - Similarity is only based on common terms, which are treated as independent of each other.
  - Vector Space Model is the answer to these problems.

Vector space model

↳ an algebric model

↳ represents text docs as vectors

→ USES → first use was SMART IR System

↳ info filtering

↳ info retrival

↳ indexing

↳ relavancy ranking

Vector Representation of Docs

↳ build a dictionary

↳ size of vocab → V

↳ V-dim vector —(represents)→ docs in limmited vocab space

↳ dictionary is in alphabetical order of vocabulary

d: <0,1,0,1,.... 0>

Vector Representation of Query

↳ represented similar to doc

↳ V-dim vector —(represents)→ query

↳ only present term in query are non-zero vector

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

column vector → tells which terms appear in that doc

Each document is represented by a binary vector $\in \{0,1\}^{|V|}$ (|V| → size of vocab)

Term Freauency tf

↳ freq of terms in a doc

Sub linear tf scalling

↳ tf can be scaled in diff ways

- Log Linear $1+\log(\mathrm{tf}_{t,d})$ → relavence doesnt increase as much with tf
- Augmented $0.5+\frac{0.5\times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$
- Maximum Frequency normalization

Document Freqency df

↳ freq of docs of a term → tells rarer terms

more informative term = low df
less informative term = high df

- Scaling the df $\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t}$
- Augmented $\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$

inverse df

$$\mathrm{idf}_t = \log_{10} \frac{N}{\mathrm{df}_t}$$

total no of docs

term

Term-document count matrices

- Consider the number of occurrences of a term in a document:
  - Each document is a count vector in $\mathbb{N}^{|V|}$: a column below

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

better to sort

idf example, suppose N = 1 million

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1,000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

$$\mathrm{idf}_t = \log_{10}(N/\mathrm{df}_t)$$

There is one idf value for each term t in a collection.

tf-idf weighting

↳ most popular and effective weighting scheme

$$\text{tf-idf}_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$
$$= 1 + \log(\mathrm{tf}) \times \log_{10}(N/\mathrm{df})$$

↳ assign term t a weight in doc d

↳ t occurs many times with small no. of docs → High W

↳ t occurs few/too many times in docs
↳ t is in all docs
→ low weight

Binary → count → weight matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Each document is now represented by a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$

Different Variants

Term frequency		Document frequency		Normalization	
n (natural)	$tf_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1+\log(tf_{t,d})$	t (idf)	$\log \frac{N}{df_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5+\frac{0.5\times tf_{t,d}}{\max_t(tf_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-df_t}{df_t}\}$	u (pivoted unique)	$1/u$ (Section 6.4.4)
b (boolean)	$\begin{cases}1 & \text{if } tf_{t,d}>0\\ 0 & \text{otherwise}\end{cases}$			b (byte size)	$1/CharLength^{\alpha}, \alpha<1$
L (log ave)	$\frac{1+\log(tf_{t,d})}{1+\log(ave_{t\in d}(tf_{t,d}))}$				

▶ Figure 6.15 SMART notation for tf-idf variants. Here *CharLength* is the number of characters in the document.

PROS

↳ simple model based on Linear Algebra

↳ Term weights aren't binary

↳ can compute continous degree of similarity b/w d and q

↳ docs ranked according to relevance

↳ allows partial matching

CONS

↳ order of terms appearing in docs is lost

↳ assumes terms are statistically independent

↳ semantically sensitive

- Search keywords must precisely match document terms; word substrings might result in a "false positive match"
- Semantic sensitivity; documents with similar context but different term vocabulary won't be associated, resulting in a "false negative match".

Cosine similarity

↳ measures similarity b/w q and d in VSM

$$\cos\theta = \frac{d\ q}{|d|\ |q|}$$

$$\cos\theta : \frac{(V_1 \cdot Y_1)+(V_2 \cdot Y_2)+(V_3 \cdot Y_3)}{\sqrt{(V_1^2+V_2^2+V_3^2)(Y_1^2+Y_2^2+Y_3^2)}}$$

Example : The Vector Space Model

As an example, consider two documents D1 = (0.5, 0.8, 0.3) and D2 = (0.9, 0.4, 0.2) indexed by three terms, where the numbers represent term **weights**.

Given the query Q = (1.5, 1.0, 0) indexed by the same terms, the cosine measures for the two documents are:

$$Cosine(D_1,Q) = \frac{(0.5\times 1.5)+(0.8\times 1.0)}{\sqrt{(0.5^2+0.8^2+0.3^2)(1.5^2+1.0^2)}} = \frac{1.55}{\sqrt{(0.98\times 3.25)}} = 0.87$$

$$Cosine(D_2,Q) = \frac{(0.9\times 1.5)+(0.4\times 1.0)}{\sqrt{(0.9^2+0.4^2+0.2^2)(1.5^2+1.0^2)}} = \frac{1.75}{\sqrt{(1.01\times 3.25)}} = 0.97$$

SMART System

- The SMART system did a rigor research on different schemes, for document and query they have used mnemonic ddd.qqq
- The first letter in each triplet specifies the term frequency component of the weighting, the second the document frequency component, and the third the form of normalization used.
- For example, a very standard weighting scheme is lnc.ltc, where the document vector has log-weighted term frequency, no idf (for both effectiveness and efficiency reasons), and cosine normalization, while the query vector uses log-weighted term frequency, idf weighting, and cosine normalization.

Consider the following given document-collection and a query.

d1: w1 w2 w4 w6
d2: w1 w2 w7 w3
d3: w8 w5 w4 w5 w6
q: w2 w5 w6

Using the Vector Space Model (VSM) for IR, compute, document vectors using tf*idf weighting scheme, where tf is the term frequency in a document, and idf is log(N/df) {N is total number of documents and df is the document frequency of term t} with * as multiplication of the two factors. Rank the given document w.r.t. query q using cosine function as a distance. Show all intermediates steps of calculations.

N=3

	idf	tf-D1	tf-D2	tf-D3	tf-q	tf*idf D1	tf*idf D2	tf*idf D3	tf*idf q	df
w1	0.18	1	1	0	0	0.18	0.18	0	0	2
w2	0.18	1	1	0	1	0.18	0.18	0	0.18	2
w3	0.48	0	1	0	0	0	0.48	0	0	1
w4	0.18	1	0	1	0	0.18	0	0.18	0	2
w5	0.48	0	0	2	1	0	0	0.96	0.48	1
w6	0.18	1	0	1	1	0.18	0	0.18	0.18	2
w7	0.48	0	1	0	0	0	0.48	0	0	1
w8	0.48	0	0	1	0	0	0	0.48	0	1

Vector d1 =<0.18,0.18,0,0.18,0,0.18,0,0> and |d1| = 0.36

Vector d2 =<0.18,0.18,0.48,0,0,0,0.48,0> and |d2| = 0.73

Vector d3 =<0,0,0,0.18,0.96,0.18,0,0.48> and |d3| = 1.10

Vector q = <0,0.18,0,0,0.48,0.18,0,0> and |q| = 0.54

Sim(d1,q) = cos(d1,q) = (d1 X q) / (|d1| x |q|) = 0.73

Sim(d2,q) = cos(d2,q) = (d2 X q) / (|d2| x |q|) = 0.08

$$\cos(d_1, q) = \frac{(0.18 \times 0.18) + (0.18 \times 0.18)}{\sqrt{0.18^2 + 0.18^2 + 0.18^2 + 0.18^2}\ \sqrt{0.18^2 + 0.48^2 + 0.18^2}} = 0.33123$$

In Section 6.3 we will develop a more rigorous form of Equation (6.9).

Exercise 6.8

Why is the idf of a term always finite? cuz it is calculated based on an log ratio of: total no of docs / no of docs containing the term. since the no. of doc in a corpus if finite, IDF will also be finite

Exercise 6.9

What is the idf of a term that occurs in every document? Compare this with the use of stop word lists.

idf=0

	Doc1	Doc2	Doc3
car	27	4	24
auto	3	33	0
insurance	0	33	29
best	14	0	17

► **Figure 6.9** Table of tf values for Exercise 6.10.

term	df_t	idf_t
car	18,165	1.65
auto	6723	2.08
insurance	19,241	1.62
best	25,235	1.5

► **Figure 6.8** Example of idf values. Here we give the idf's of terms with various frequencies in the Reuters collection of 806,791 documents.

Exercise 6.10

Consider the table of term frequencies for 3 documents denoted Doc1, Doc2, Doc3 in Figure 6.9. Compute the tf-idf weights for the terms car, auto, insurance, best, for each document, using the idf values from Figure 6.8.

tf-idf = tf × idf

	Doc 1	Doc2	Doc3
car	27(1.65) = 44.55	4(1.65) = 6.6	24(1.65) = 39.6
auto	3(2.08) = 6.24	68.64	0
insurance	0	53.46	46.98
best	21	0	22.5

Exercise 6.11

Can the tf-idf weight of a term in a document exceed 1?

yes, if tf is very high than idf

Exercise 6.15

Recall the tf-idf weights computed in Exercise 6.10. Compute the Euclidean normalized document vectors for each of the documents, where each vector has four components, one for each of the four terms.

	Doc 1	Doc 2	Doc 3
car	27(1.65) = 44.55	4(1.65) = 6.6	24(1.65) = 39.6
auto	3(2.08) = 6.24	68.64	0
insurance	0	53.46	46.98
best	21	0	22.5

$Doc1 = \sqrt{44.55^2 + 6.24^2 + 0 + 21^2} = 49.64$

$Doc2 = \sqrt{6.6^2 + 68.64^2 + 53.46^2} = 87.25$

$Doc3 = \sqrt{39.6^2 + 46.98^2 + 22.5^2} = 65.43$

car: $44.55/51.40 = 0.891$

auto: $6.24/51.40 = 0.125$

insurance = 0

best: $21/51.40 = 0.423$

Exercise 6.17

With term weights as computed in Exercise 6.15, rank the three documents by computed score for the query car insurance, for each of the following cases of term weighting in the query:

1. The weight of a term is 1 if present in the query, 0 otherwise.
2. Euclidean normalized idf.

Q: car insurance

	Doc 1	Doc 2	Doc 3	w
car	0.897	0.076	0.595	1
auto	0.125	0.786	0	0
insurance	0	0.613	0.706	1
best	0.423	0	0.383	0

q[1,0,1,0]

1) score

D1 = 0.897 + 0 = 0.897

D2 = 0.076 + 0.613 = 0.689

D3 = 0.595 + 0.706 = 1.301

Ranking: D3, D1, D2

q[0.4478, 0.6024, 0.4692, 0.4344]

2) score

?

Ranking: D2, D3, D1

0.7975 0.7825 0.6883

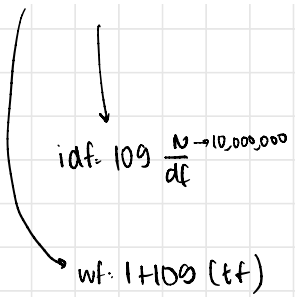
that produced by cosine similarities.

Exercise 6.19

Compute the vector space similarity between the query "digital cameras" and the document "digital cameras and video cameras" by filling out the empty columns in Table 6.1. Assume $N = 10{,}000{,}000$, logarithmic term weighting (wf columns) for query and document, idf weighting for the query only and cosine normalization for the document only. Treat and as a stop word. Enter term counts in the tf columns. What is the final similarity score?

N = 10,000,000

	query					document			
word	tf	wf	df	idf	q_i = wf-idf	tf	wf	d_i = normalized wf	$q_i \cdot d_i$
digital	1	1	10,000	3	3	1	1	0.52	1.56
video	0	0	100,000	2	0	1	1	0.52	0
cameras	1	1	50,000	2.3	2.3	2	1.3	0.68	1.56



Similarity score = 1.56 + 1.56
= 3.12

Exercise 6.20

Show that for the query **affection**, the relative ordering of the scores of the three documents in Figure 6.13 is the reverse of the ordering of the scores for the query **jealous gossip**.

term	SaS	PaP	WH
affection	0.996	0.993	0.847
jealous	0.087	0.120	0.466
gossip	0.017	0	0.254

► **Figure 6.13** Term vectors for the three novels of Figure 6.12. These are based on raw term frequency only and are normalized as if these were the only terms in the collection. (Since affection and jealous occur in all three documents, their tf-idf weight would be 0 in most formulations.)

Q: affection

Score

SaS = 0.996

PaP = 0.993

WH = 0.847

O = SAS, PaP, WH

Q) Jealous gossip

Score

SaS = 0.996 + 0.087 = 1.083

PaP = 0.993 + 0.12 = 1.113

WH = 0.847 + 0.466 = 1.313

O = WH, PaP, SaS

	Doc1	Doc2	Doc3
car	27	4	24
auto	3	33	0
insurance	0	33	29
best	14	0	17

► **Figure 6.9** Table of tf values for Exercise 6.10.

term	df_t	idf_t
car	18,165	1.65
auto	6723	2.08
insurance	19,241	1.62
best	25,235	1.5

► **Figure 6.8** Example of idf values. Here we give the idf's of terms with various frequencies in the Reuters collection of 806,791 documents.

Exercise 6.10

Consider the table of term frequencies for 3 documents denoted Doc1, Doc2, Doc3 in Figure 6.9. Compute the tf-idf weights for the terms car, auto, insurance, best, for each document, using the idf values from Figure 6.8.

tf-idf = tf × idf

	Doc 1	Doc2	Doc3
car	27(1.65) = 44.55	4(1.65) = 6.6	24(1.65) = 39.6
auto	3(2.00) = 6.24	68.64	0
insurance	0	53.46	46.98
best	21	0	22.5

Exercise 6.23

Refer to the tf and idf values for four terms and three documents in Exercise 6.10. Compute the two top scoring documents on the query **best car insurance** for each of the following weighing schemes: (i) `nnn.atc`; (ii) `ntc.atc`.

q = best car insurance

Agenda

- Efficient scoring and ranking
- Inexact Top k documents
- Index Elimination
- Champion Lists
- Static Quality Scores
- Impact Ordering
- Cluster Pruning
- Conclusion

Efficient Scoring and Ranking

- Modified Cosine computation as discussed in the last lecture.
  - Only compute the partial scores for each q term and document d (only common dimensions will add values to score).
  - Should we go for all N document- obviously not.
  - We are only interested in k top ranked documents
  - How we can make it efficient?
    - If we know in advance that which documents are high-scoring for a given query?
    - If we can arrange all the documents score in an efficient top retrieval data structures (Max. Heap)
    - Any rough estimates of ranking which can be computed quickly will be helpful.

Fast Cosine Scores

↳ how relevant a doc is

```
FastCosineScore(q)
1  float Scores[N] = 0
2  for each d
3  do Initialize Length[d] to the length of doc d
4  for each query term t
5  do calculate w_{t,q} and fetch postings list for t
6     for each pair(d, tf_{t,d}) in postings list
7     do add wf_{t,d} to Scores[d]
8  Read the array Length[d]
9  for each d
10 do Divide Scores[d] by Length[d]
11 return Top K components of Scores[]
```

Inexact Serach

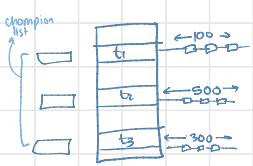
- Inexact Top Key Document Retrieval
  - Find a set A of documents that are contenders, where K < |A| ≪ N.
  - A does not necessarily contain the K top-scoring documents for the query, but is likely to have many documents with scores near those of the top K.
  - Return the K top-scoring documents in A.
- Index Elimination → reduces no of docs considered during the scoring process of a query
- This is based on two key factors:
  - We only consider documents that contain many (and as a special case, all) of the query terms (tf).
  - Only Consider idf query terms with high value(not highest)

Champion Lists

↳ computed at the time of building dictionary

↳ for every dict term t, the champion list contains the top k docs which have the highest weights in the posting list for t

- Call it champion list for term t
- Only compute the scores for k documents.
- This k may not be same for every term t.
- Pick the top (10-20 documents based on scores)



Static Quality Scores

↳ top ranking docs should be

↳ Relevant → Terms/cosine scores

↳ Authoritative → query independent properties of a doc → Page Rank

↳ How to Model Authority of a doc

↳ assign a query independent quality score to each doc

↳ quality score [0,1] → denoted by g(d) → Authority

quality / goodness

↳ Net Score

↳ return top k docs

$$\text{Net score}(q,d) = g(d) + \cos(q,d)$$

(g(d): authority; cos(q,d): relevance; → weighted combination)

↳ Global champion lists

Impact Ordering

↳ computes scores for docs which have high enough $wf_{t,d}$

↳ sort each posting by $wf_{t,d}$

↳ now no common order

↳ How do we compute scores to be picked off top

1. Early termination

- When traversing t's postings, stop early after either
  - a fixed number of r docs
  - $wf_{t,d}$ drops below some threshold
- Take the union of the resulting sets of docs
  - One from the postings of each query term
- Compute only the scores for docs in this union

2. idf-ordered terms

- When considering the postings of query terms
- Look at them in order of decreasing idf → high idf terms likely to contribute most to score
- As we update score contribution from each query term
  - Stop if doc scores relatively unchanged
- Can apply to cosine or some other net scores

Threshold on idf: we ignore documents that contain terms with low inverse document frequency (idf). This saves time because low-idf terms usually have long lists of documents, so skipping them reduces the number of documents we need to check.

Cluster Pruning

Cluster pruning: preprocessing

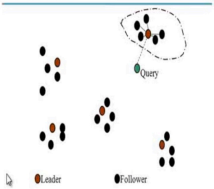
- Pick $\sqrt{N}$ *docs* at random: call these *leaders*
- For every other doc, pre-compute nearest leader
  - Docs attached to a leader: its *followers*;
  - Likely: each leader has ~ $\sqrt{N}$ followers.

Cluster pruning: query processing

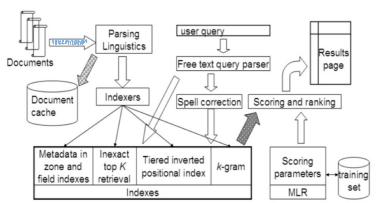
- Process a query as follows:
  - Given query Q, find its nearest *leader* L.
  - Seek K nearest docs from among L's followers.
  - Apply cosine similarity to these

each follower will use cosine rule and whichever leader is closest to it, it will become its follower

Visualization



Complete Search Systems



General Strategies – search

- Tiered Index - A common solution to this issue is the user of tiered indexes, which may be viewed as a generalization of champion lists.
- Query-term proximity - Consider a query with two or more query terms, t1 , t2, . . . , tk. Let ω be the width of the smallest window in a document d that contains all the query terms, measured in the number of words in the window.
- Intuitively, the smaller that ω is, the better that d matches the query.
- Proximity weighting is very subjective.

Conclusion

- Large Scale search systems – perform a lot of optimizations
- If we can present results in a Proxy form and we can make a user HAPPY. There is no issue at all.
- She will never feel the difference.

ishma hafeez notes

Agenda
- Evaluation in IR
- Ad Hoc Information Retrieval
- Standard IR Collections
- Evaluation for Unranked Retrieval
  - Precision
  - Recall
  - F-Score or F- measure
  - Fall-out
- Evaluation for Ranked Retrieval

Agenda
- Evaluation for Ranked Retrieval
  - Precision –Recall Curve
  - Average Precision
  - Mean Average Precision (MAP)
  - Cumulative Gain
  - Discount Cumulative Gain
  - Normalized Discount Cumulative Gain
- Conclusion

Different IR models

↳ what is the best component for
- Ranking function : dot product, cosine
- Term Selection : stopword removal, stemming
- Term weighting : tf, tf-idf

Information Needs

- Information Need
  - Drinking red wine is more effective at reducing your risk of heart attacks than white wine.
- Query
  - wine and red and white and heart and attack and effective

Difficulty in IR Evaluation

↳ effectiveness is related to the relevancy of retrieved items
↳ Relevancy
  ↳ it is typically continous (even if it was binary it can be a difficult judgement to make)
  ↳ from a human standpoint it is:
    ↳ Subjective: Depends upon a specific users judgement
    ↳ Situational: Relates to current users needs
    ↳ Cognitive: Depends on human perception and behavior
    ↳ Dynamic: changes over time

Ad hoc Information Retrival Effectiveness

↳ to test its effectiveness, our test collection should consist of
1. A document collection
2. A test suite of information needs (aka queries)
3. A set of relevance judgements (Standardly a binary assessment of relevant/non relevant for each query doc pair)

Evaluation in IR

↳ It assess how well the search results satisfies the user's query intent
↳ It compares 2 IR systems
↳ it is still an active area of research
  ↳ it initally started with small dataset with 100's doc and 30 queries
    now it has grown to 1/15 of web scale

Standard IR Collections

→ data sets used to evaluate the performance of information retrival systems

- The Cranfield collection.
  - Collected in the United Kingdom starting in the late 1950s, it contains 1398 abstracts of aerodynamics journal articles, a set of 225 queries, and exhaustive relevance judgments of all (query, document) pairs.
- 20 Newsgroups
  - It consists of 1000 articles from each of 20 Usenet newsgroups (the newsgroup name being regarded as the category).

Standard IR Collections

- Cross Language Evaluation Forum (CLEF)
  - This evaluation series has concentrated on EU languages and cross-language information retrieval.
- Reuters-21578 and Reuters-RCV1
  - For text classification, the most used test collection has been the Reuters-21578 collection of 21578 newswire articles
- WebKB
  - This data set contains WWW-pages collected from computer science departments of various universities in January 1997
- Modern IR Collections
  - TREC
  - SemEval
  - LSHTC
  - CLEF
  - MediaEval

TREC Conference Tracks

Image Captioning

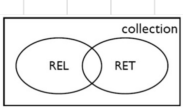
TextCaps dataset

v0.1

Training set	Validation set	Test set
• 109,765 captions (173MB)	• 15,830 captions (25MB)	• Metadata (6.5MB)
• 21,953 images (6.6GB)	• 3,166 images	• 3,289 images (926MB)
• Rosetta OCR tokens [v0.2]	• Rosetta OCR tokens [v0.2]	• Rosetta OCR tokens [v0.2]

→ that results only retrieved ones which system consider relevant

Evaluation of unranked retrival sets



$$\mathcal{P} = \frac{|RET \cap REL|}{|RET|}$$

$$\mathcal{R} = \frac{|RET \cap REL|}{|REL|}$$

Prescion

$$\frac{\text{Relavent Ret}}{\text{Total Ret}}$$

Recall

$$\frac{\text{Relavent Ret}}{\text{Relavent}}$$

Precision

100% → gives all data set

↳ portion of retrived d. that are relevant

↳ measures accuracy

↳ measures how many retrived d. were relavent

$$P = \frac{\text{Relavent Ret}}{\text{Total Ret}}$$

Recall

100% → retrives 1 d. that is also relevant

↳ portion of relevant d. that are retrived

↳ measures how many relavent d. were retrived

$$R = \frac{\text{Relavent Ret}}{\text{Relavent}}$$

IR Evaluation

retrieved \ relevant	Rel	NotRel	
Ret	Ret_{Rel}	Ret_{NotRel}	$Ret = Ret_{Rel} + Ret_{NotRel}$
NotRet	$NotRet_{Rel}$	$NotRet_{NotRel}$	$NotRet = NotRet_{Rel} + NotRet_{NotRel}$

$Relevant = Ret_{Rel} + NotRet_{Rel}$ $\quad$ $Not\ Relevant = Ret_{NotRel} + NotRet_{NotRel}$

$Total\ \#\ of\ documents\ available\ N = Ret_{Rel} + NotRet_{Rel} + Ret_{NotRel} + NotRet_{NotRel}$

- Precision: $P = Ret_{Rel} / Retrieved$
- Recall: $R = Ret_{Rel} / Relevant$

$P = [0,1]$
$R = [0,1]$

Example

	Retrieved	Not retrieved	
Relevant	w=3	x=2	Relevant = w+x= 5
Not relevant	y=3	z=2	Not Relevant = y+z = 5

Retrieved = w+y = 6 $\quad$ Not Retrieved = x+z = 4

Total documents N = w+x+y+z = 10

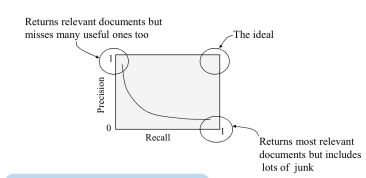
- Precision: P= w / w+y =3/6 =.5
- Recall: R = w / w+x = 3/5 =.6

↳ A SYSTEM CAN MAKE 2 TYPES OF ERRORS

1. A false positive error
   - ↳ system retrives a d. that is non-relevant
   - ↳ affects precision

2. A false negative error
   - ↳ system fails retrives a d. that is relevant
   - ↳ affects recall

Precision Vs. Recall



Precision and Recall are inverse proportional

Precision Vs. Recall

$P \propto \frac{1}{R}$

Precision Critical Tasks	Recall Critical Tasks
Time matters a lot	Time matter less
Tolerance to missed documents	Non tolerance to missed documents
Redundant – many equal information resources	Less redundant information – only one (few resources)
Example: Web search	Example: legal/patent search
Demand: Very high	Demand: moderate
General optimizations	Specific optimizations
affected by false +ve error	affected by false -ve error

↳ Precision and recall stand in opposition
↳ Precision goes ↑, recall goes ↓ (vice versa)

F-Measure

↳ combines Precision and Recall

$$\text{F-Measure} = \frac{(\beta^2 + 1)\, P.R}{\beta^2.P + R}$$

$\beta = 1$, Precision and recall are weighted
$\beta < 1$, Precision favoured
$\beta > 1$, recall favoured

using $\beta = 1$

$$\text{Fmeasure} = \frac{2PR}{P+R}$$

Fallout Rate

↳ Problems with both Precison and recall

1. no. of irrelevant d. in the collection is not taken into account
2. Recall is undefined, when theres no relevant d. in the collection
3. Precision is undefined, when no d. is retrived

opposite of Recall

$$\text{Fallout} = \frac{\text{no. of non Rel items retrived}}{\text{total no. of nonRel items in the collection}} = \frac{\text{Non Relavent Ret}}{\text{total Non Relavent}}$$

Evaluation of Ranked Retrival Results

- Precision, recall, and the F measure are set-based measures. They are computed using unordered sets of documents.
- In a ranked retrieval context, appropriate sets of retrieved documents are naturally given by the top k retrieved documents.
- The system can return any number of results.
- How to compute this ranked result?
  - A precision-recall curve

Ranked Retrieval Example

n	doc #	relevant
1	588	x
2	589	x
3	576	
4	590	x
5	986	
6	592	x
7	984	
8	988	
9	578	
10	985	
11	103	
12	591	
13	772	x
14	990	

Let total # of relevant docs = 6
Check each new recall point:

R=1/6=0.167; P=1/1=1

R=2/6=0.333; P=2/2=1

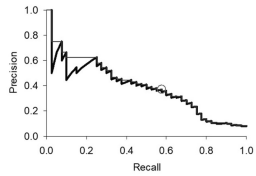
R=3/6=0.5; P=3/4=0.75

R=4/6=0.667; P=4/6=0.667

R=5/6=0.833; P=5/13=0.38

Missing one relevant document. Never reach 100% recall

A precision-recall curve



- Precision - recall curve: A ~~graph~~ is a graphical representation of the relationship between precision and recall for a given retrieval system. It shows how the precision of the retrieved documents changes as recall increases.

Average Precision

= the relevant documents

Ranking #1

Recall	0.17	0.17	0.33	0.5	0.67	0.83	0.83	0.83	0.83	1.0
Precision	1.0	0.5	0.67	0.75	0.8	0.83	0.71	0.63	0.56	0.6

Ranking #2

Recall	0.0	0.17	0.17	0.17	0.33	0.5	0.67	0.67	0.83	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.5	0.57	0.5	0.56	0.6

Ranking #1: $(1.0 + 0.67 + 0.75 + 0.8 + 0.83 + 0.6)/6 = 0.78$

Ranking #2: $(0.5 + 0.4 + 0.5 + 0.57 + 0.56 + 0.6)/6 = 0.52$

MAP

= relevant documents for query 1

Ranking #1

Recall	0.2	0.2	0.4	0.4	0.4	0.6	0.6	0.8	1.0
Precision	1.0	0.5	0.67	0.5	0.4	0.5	0.43	0.38	0.44	0.5

= relevant documents for query 2

Ranking #2

Recall	0.0	0.33	0.33	0.33	0.67	0.67	1.0	1.0	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.33	0.43	0.38	0.33	0.3

average precision query 1 $= (1.0 + 0.67 + 0.5 + 0.44 + 0.5)/5 = 0.62$
average precision query 2 $= (0.5 + 0.4 + 0.43)/3 = 0.44$

mean average precision $= (0.62 + 0.44)/2 = 0.53$

Mean Average Precision (MAP)

↳ If a rel d. never gets retrived
  ↳ assume Precison of the rel d. = 0
↳ MAP is macro avging: each query counts equally
↳ It assumes user is interested in finding many rel d. for each query
↳ Req many relevance judgements in text collection

Kappa Statistics

		Judge 2 Relevance: Yes	No	Total
Judge 1 Relevance	Yes	300	20	320
	No	10	70	80
	Total	310	90	400

Observed proportion of the times the judges agreed
$P(A) = (300 + 70)/400 = 370/400 = 0.925$
Pooled marginals
$P(nonrelevant) = (80 + 90)/(400 + 400) = 170/800 = 0.2125$
$P(relevant) = (320 + 310)/(400 + 400) = 630/800 = 0.7878$
Probability that the two judges agreed by chance
$P(E) = P(nonrelevant)^2 + P(relevant)^2 = 0.2125^2 + 0.7878^2 = 0.665$
Kappa statistic
$\kappa = (P(A) - P(E))/(1 - P(E)) = (0.925 - 0.665)/(1 - 0.665) = 0.776$

► **Table 8.2** Calculating the kappa statistic.

Figure 8.1.

Exercise 8.1 [⋆]

An IR system returns 8 relevant documents, and 10 nonrelevant documents. There are a total of 20 relevant documents in the collection. What is its precision of the system on this search, and what is its recall?

Exercise 8.2 [⋆]

Rel = 8 NR = 10

Total Rel = 20 Total Retr = 18

$P = \frac{8}{10} = 0.44$ $R = \frac{8}{20} = 0.4$

Exercise 8.8 [⋆]

Consider an information need for which there are 4 relevant documents in the collection. Contrast two systems run on this collection. Their top 10 results are judged for relevance as follows (the leftmost item is the top ranked search result):

System 1	R (1)	N	R (3)	N	N	N	N	N	R (9)	R (10)
System 2	N	R (2)	N	N	R (5)	N	R (6)	R (7)	N	N

a. What is the MAP of each system? Which has a higher MAP?

b. Does this result intuitively make sense? What does it say about what is important in getting a good MAP score?

c. What is the R-precision of each system? (Does it rank the systems the same as MAP?)

a) S1 MAP: mean avg P $= \left(\frac{1}{1} + \frac{2}{3} + \frac{3}{9} + \frac{4}{10}\right)\frac{1}{4} = 0.6$

S2 $= \left(\frac{1}{2} + \frac{2}{5} + \frac{3}{6} + \frac{4}{7}\right)\frac{1}{4} = 0.493$

b) relavant d. appearing earlyer have higher MAP

c) S1 RP: $\frac{2}{4} = 0.5$ ← 2 R in top 4

S2 RP: $\frac{1}{4} = 0.25$ ← 1 R in top 4

Exercise 8.9 [⋆⋆]

The following list of Rs and Ns represents relevant (R) and nonrelevant (N) returned documents in a ranked list of 20 documents retrieved in response to a query from a collection of 10,000 documents. The top of the ranked list (the document the system thinks is most likely to be relevant) is on the left of the list. This list shows 6 relevant documents. Assume that there are 8 relevant documents in total in the collection.

R R N N N N N R N R N N N R N N N N R

a. What is the precision of the system on the top 20?
b. What is the F_1 on the top 20?
c. What is the uninterpolated precision of the system at 25% recall?
d. What is the interpolated precision at 33% recall?
e. Assume that these 20 documents are the complete result set of the system. What is the MAP for the query?

Assume, now, instead, that the system returned the entire 10,000 documents in a ranked list, and these are the first 20 results returned.

f. What is the largest possible MAP that this system could have?
g. What is the smallest possible MAP that this system could have?
h. In a set of experiments, only the top 20 results are evaluated by hand. The result in (e) is used to approximate the range (f)-(g). For this example, how large (in absolute terms) can the error for the MAP be by calculating (e) instead of (f) and (g) for this query?

Rel = 6
Total Rel = 8
Total = 20

a) $P = \frac{6}{20} = 0.3$

b) $R = \frac{6}{8} = 0.75$

$F = \frac{2(0.3)(0.75)}{0.3+0.75} = 0.43$ $\frac{2PR}{P+R}$

c) uninterpolated P at 25% recall

$8 \times 0.25 = 2$ → no. of rel d. retrived

total Rel

$P = \frac{2}{20} = 0.1$

d) interpolated P at 33% recall

$8 \times 0.33 = 2.64 \approx 3$

$P = \frac{3}{20} = 0.15$

$F_1 = \frac{2PR}{P+R} = \frac{3}{7} = 0.43$

c. 8*0.25 = 2, the uninterpolated precision could be 1, 2/3, 2/4, 2/5, 2/6, 2/7, 1/4

d. Because the highest precision found for any recall level larger than 33% is 4/11 = 0.364, hence the interpolated precision at 33% recall is 4/11 = 0.364.

e. MAP = 1/6 * (1 + 1 + 3/9 + 4/11 + 5/15 + 6/20) = 0.555

e) MAP: $\left(\frac{1}{1} + \frac{2}{2} + \frac{3}{9} + \frac{4}{11} + \frac{5}{15} + \frac{6}{20}\right)\frac{1}{6}$

0.364, hence the interpolated precision at 33% recall is 4/11 = 0.364.
e. MAP = 1/6 * (1 + 1 + 3/9 + 4/11 + 5/15 + 6/20) = 0.555

f. $MAP_{largest} = \frac{1}{8} * \left(1 + 1 + \frac{3}{9} + \frac{4}{11} + \frac{5}{15} + \frac{6}{20} + \frac{7}{21} + \frac{8}{22}\right) = 0.503$

g. $MAP_{smallest} = \frac{1}{8} * \left(1 + 1 + \frac{3}{9} + \frac{4}{11} + \frac{5}{15} + \frac{6}{20} + \frac{7}{9999} + \frac{8}{10000}\right) = 0.417$

h. 0.555-0.417 = 0.138, 0.555-0.503 = 0.052
the error is in [0.052, 0.138]

Exercise 8.10 [⋆⋆]

Below is a table showing how two human judges rated the relevance of a set of 12 documents to a particular information need (0 = nonrelevant, 1 = relevant). Let us assume that you've written an IR system that for this query returns the set of documents {4, 5, 6, 7, 8}.

NR 6
R: 6

NR 6
R: 6

docID	Judge 1	Judge 2
1	0	0
2	0	0
3	1	1
4	1	1
5	1	0
6	1	0
7	1	0
8	1	0
9	0	1
10	0	1
11	0	1
12	0	1

→ agreements

R
R
R
R
NR

a. Calculate the kappa measure between the two judges.

b. Calculate precision, recall, and F_1 of your system if a document is considered relevant only if the two judges agree.

c. Calculate precision, recall, and F_1 of your system if a document is considered relevant if either judge thinks it is relevant.

8.6 A broader perspective: System quality and user utility

a) $P(A) = \frac{4}{12} = \frac{1}{3}$ (4 → no of agreements; 12 → total d.)

$P(NR) = \frac{6+6}{12+12} = 0.5$ $\quad P(R) = \frac{6+6}{12+12} = 0.5$

$P(E) = P(NR)^2 + P(R)^2$

$0.5^2 + 0.5^2 = 0.5$

$K = \frac{P(A) - P(E)}{1 - P(E)} = \frac{1/3 - 0.5}{0.5} = -\frac{1}{3}$

b) Total = 5
Total R = 2
Rel = 1

$P = \frac{1}{5} = 0.2$

$R = \frac{1}{2} = 0.5$

$F = \frac{2(0.2)(0.5)}{0.2+0.5} = 0.286$

c) Total = 5
Total R = 10
Rel = 5

$P = \frac{5}{5} = 1$

$R = \frac{5}{10} = 0.5$

$F = \frac{2(1)(0.5)}{1+0.5} = 0.667$

Hence, Ranking is d2,d3,d1

Evaluation in IR	
Question No. 3	**[Time: 25 Min] [Marks: 15]**

a. There are 20 relevant documents in a collection for a given query "q", The precision of the query is 0.20 and the recall is 0.25 Find How many documents in the results-set retrieved? How many of them would be relevant? [5]

we know,
precision = (relevant-retrieved) / (total-retrieved)
=> 0.2 = (relevant-retrieved) / (result-set) ---------- eq(i)
similarly,
recall = (relevant-retrieved)/ (total-relevant)
=> 0.25 = (relevant-retrieved)/ 20
=> relevant-retrieved= 0.25 * 20 = 5 hence eq(i) => result-set = 5 /0. 2 = 25
Result-set contain 25 documents from which, 5 are relevant.

b. The following list of Rs and Ns represents relevant (R) and non-relevant (N) returned documents in a ranked list of 10 documents retrieved in response to a query from a collection of 1,000 documents. The top of the ranked list (the documents the system thinks are most likely to be relevant) is on the left of the list. This list shows 4 relevant documents. Assume that there are 6 relevant documents in total in the collection for the given query. [10]

R R N N R N R N N N

1. What is the precision of the system on the top 10?

From the given information, we can see that tp=4; fp=6 and fn=6-4=2 so for precision we have Precision = tp / (tp+fp) = 4/10 = 0.4

2. What is the F1 on the top 10?

Let's find recall for F1: we know Recall = tp/ (fn+tp) = 4/6= 0.66 hence
F1= 2 X (Precision * Recall) / (Precision + Recall)
F1= (2X0.4X0.66) / (0.4+0.66) = 0.528 / 1.06 = 0.498

3. What is the largest possible MAP that this system could have?

The maximum MAP possible when the remaining two relevant documents retrieved next to these 10 documents.
MAP = 1/6 * (1/1+2/2+3/5+4/7+5/11+6/12) = 4.1259/6 = 0.687
last 2 docs

4. What is the smallest possible MAP that this system could have?
The minimum MAP possible when the remaining four relevant documents found as the last documents from the collection.

MAP = 1/6 * (1/1+2/2+3/5+4/7+5/999+6/1000) = 3.1824/6 = 0.530
last 2 docs

old technique

Cumulative Gain (CG)

↳ CG = sum of relevance scores of all results in search result list

↳ changing the order of any 2 d. does not affect the CG measure

query has following docs

q = D1, D2, D3, D4, D5, D6

relevance scores (respectively) = 3, 2, 3, 0, 1, 2

$CG = \sum_{i=1}^{6} rel_i = 3+2+3+0+1+2 = 11$

$D_1, D_2, D_3, D_4, D_5, D_6$

the user provides the following relevance scores:

3, 2, 3, 0, 1, 2

That is: document 1 has a relevance of 3, document 2 has a relevance of 2, etc. The Cumulative Gain of this search result listing is:

$$CG_6 = \sum_{i=1}^{6} rel_i = 3+2+3+0+1+2 = 11$$

Discount Cumulative Gain (DCG)

↳ DCG = total gain accumulated at particular rank p

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}$$

(p: no. of docs; rel_i: relevance scores)

ALTERNATE FORMULA

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log(1+i)}$$

↳ used by web search companies

↳ emphasis on retriving highly rel d.

i	rel_i	$\log_2(i+1)$	$\frac{rel_i}{\log_2(i+1)}$
1	3	1	3
2	2	1.585	1.262
3	3	2	1.5
4	0	2.322	0
5	1	2.585	0.387
6	2	2.807	0.712

$$DCG_6 = \sum_{i=1}^{6} \frac{rel_i}{\log_2(i+1)} = 3 + 1.262 + 1.5 + 0 + 0.387 + 0.712 = 6.861$$

very popular in evaluating web search

Normalized DCG (NDGC)

↳ Normalize DCG at rank n by the DCG value at rank n of the ideal ranking

↳ ideal ranking would

↳ first return d. with highest relevance level

↳ then the next highest relevance level etc.

↳ normalization is useful for contrasting queries with varying no. of relevant results

Solved Example

i	rel_i	$\log_2(i+1)$	$\frac{rel_i}{\log_2(i+1)}$
1	3	1	3
2	2	1.585	1.262
3	3	2	1.5
4	0	2.322	0
5	1	2.585	0.387
6	2	2.807	0.712

$$DCG_6 = \sum_{i=1}^{6} \frac{rel_i}{\log_2(i+1)} = 3 + 1.262 + 1.5 + 0 + 0.387 + 0.712 = 6.861$$

Ideal Order of the result : 3, 3, 2, 2, 1, 0

$$IDCG_6 = 7.141$$

And so the nDCG for this query is given as:

$$nDCG_6 = \frac{DCG_6}{IDCG_6} = \frac{6.861}{7.141} = 0.961$$

Normalized DCG (Example)

4 documents: d_1, d_2, d_3, d_4

i	Ground Truth		Ranking Function$_1$		Ranking Function$_2$	
	Document Order	r_i	Document Order	r_i	Document Order	r_i
1	d4	2	d3	2	d3	2
2	d3	2	d4	2	d2	1
3	d2	1	d2	1	d4	2
4	d1	0	d1	0	d1	0
	$NDCG_{GT}$=1.00		$NDCG_{RF1}$=1.00		$NDCG_{RF2}$=0.9203	

$$DCG_{GT} = 2 + \left(\frac{2}{\log_2 2} + \frac{1}{\log_2 3} + \frac{0}{\log_2 4}\right) = 4.6309$$

$$DCG_{RF1} = 2 + \left(\frac{2}{\log_2 2} + \frac{1}{\log_2 3} + \frac{0}{\log_2 4}\right) = 4.6309$$

$$DCG_{RF2} = 2 + \left(\frac{1}{\log_2 2} + \frac{2}{\log_2 3} + \frac{0}{\log_2 4}\right) = 4.2619$$

$MaxDCG = DCG_{GT} = 4.6309$

System Quality

- There are many practical benchmarks on which to rate an information retrieval system beyond its retrieval quality.
- System Quality is also a concern.
  - How fast does it index, that is, how many documents per hour does it index for a certain distribution over document lengths?
  - How fast does it search, that is, what is its latency as a function of index size?
  - How expressive is its query language? How fast is it on complex queries?
  - How large is its document collection, in terms of the number of documents or the collection having information distributed across a broad range of topics?

User Utility

- What we would really like is a way of quantifying aggregate user happiness, based on the relevance, speed, and user interface of a system.
- One indirect measure of such users is that they tend to return to the same engine.

Evaluation of System changes

↳ A/B testing

↳ for this test 1 thing is changed b/w current and proposed system

↳ a small portion of traffic (10% of users) is randomly directed to the variant system, while most users to current (is not)

↳ click through log analysis/clickstream mining to see wether user liked it or not

↳ the basis of A/B testing is

↳ running a bunch of single variable tests (either in sequence or parallel)

↳ for each test only 1 parameter is varied from the control (the current live system)

Search Snippets

↳ useful for rewing search results

↳ It has 2 basic kind of summaries

1. Static Summaries

   ↳ which are always the same regardless of the query

2. Dynamic Summaries

   ↳ which are customised according to users info need as deduced from query

   ↳ it explains why a particular doc was retrived for the query at hand

   Keyword-in-context (KWIC) snippets

Conclusion

- Get as much of what we want while at the same time getting as little junk as possible.
- Recall is the percentage of relevant documents returned compared to everything that is available!
- Precision is the percentage of relevant documents compared to what is returned!
- The desired trade-off between precision and recall is specific to the scenario we are in?
- What do we want?
  - Find everything relevant – high recall
  - Only retrieve what is relevant – high precision

week 9

Agenda
- Problem of IR Systems
  - Recall /Precision
- Query Refinement
  - Global vs Local methods
- Global Methods
- Relevance Feedback
  - Direct / Indirect / Pseudo relevance feedback
- Relevance Feedback in Vector Space
  - Rocchio Algorithm

Agenda
- When Relevance Feedback work
- Relevance Feedback on Web
- Query Expansion
  - Global vs Local
  - Automatic thesaurus generation

Problems with IR systems

1. Same concept can be reffered by diff words (synonyms)
   ↳ this impacts recall

↓

Sol: Query refinement / expansion
   ↳ often done by users, manually refining a query

↓

Query Refinement Methods

1. Global methods
2. Local methods

1. Global method (change search terms)

↳ techniques for expanding or reformulating query terms independent of the query and results returned from the query

↳ changes in query wording will cause new query to match other semantically similar items

↳ Query expansion via
- ↳ Thesauras
- ↳ WordNet
- ↳ Automatic Thesauras

Query reformulation via
- ↳ Thesauras
- ↳ WordNet

↳ Techniques like spelling correction → ?

2. Local method (change search term based on feedback)

↳ adjusts a query relative to the docs that initially appear to match the query

1. Relevance feedback (most successful, used approach)
   ↳ involves users to get feedback for results returned for a given query
2. Pseudo relevance feedback (aka Blind relevance feedback)
3. Indirect relevance feedback

1. Relavance feedback (RF)

↳ involes the user in the retrival process to improve final result set

Steps

↳ user issues a query (short, simple)

↳ System returns a set of retrival results (initial)

↳ user marks some returned docs as Rel or Non Rel

↳ system uses this feedback to compute a better rep. of info

↳ System displays a revised set of retrival results

Pros

↳ improves recall and precision

↳ most usefull for increasing recall (in situations where recall is important)

- Users can be expected to review results and to take time to iterate

Sec. 9.1.1

Example 2: Initial query/results

- Initial query: *New space satellite applications*
  1. 0.539, 08/13/91, NASA Hasn't Scrapped Imaging Spectrometer
  2. 0.533, 07/09/91, NASA Scratches Environment Gear From Satellite Plan
  3. 0.528, 04/04/90, Science Panel Backs NASA Satellite Plan, But Urges Launches of Smaller Probes
  4. 0.526, 09/09/91, A NASA Satellite Project Accomplishes Incredible Feat: Staying Within Budget
  5. 0.525, 07/24/90, Scientist Who Exposed Global Warming Proposes Satellites for Climate Research
  6. 0.524, 08/22/90, Report Provides Support for the Critics Of Using Big Satellites to Study Climate
  7. 0.516, 04/13/87, Arianespace Receives Satellite Launch Pact From Telesat Canada
  8. 0.509, 12/02/87, Telecommunications Tale of Two Companies
- User then marks relevant documents with "+".

Sec. 9.1.1

Expanded query after relevance feedback

2.074 new	15.106 space
30.816 satellite	5.660 application
5.991 nasa	5.196 eos
4.196 launch	3.972 aster
3.516 instrument	3.446 arianespace
3.004 bundespost	2.806 ss
2.790 rocket	2.053 scientist
2.003 broadcast	1.172 earth
0.836 oil	0.646 measure

Sec. 9.1.1

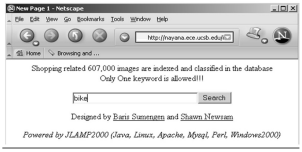
Results for expanded query

2 1. 0.513, 07/09/91, NASA Scratches Environment Gear From Satellite Plan
1 2. 0.500, 08/13/91, NASA Hasn't Scrapped Imaging Spectrometer
3. 0.493, 08/07/89, When the Pentagon Launches a Secret Satellite, Space Sleuths Do Some Spy Work of Their Own
4. 0.493, 07/31/89, NASA Uses 'Warm' Superconductors For Fast Circuit
5. 0.492, 12/02/87, Telecommunications Tale of Two Companies
8 6. 0.491, 07/09/91, Soviets May Adapt Parts of SS-20 Missile For Commercial Use
7. 0.490, 07/12/88, Gaping Gap: Pentagon Lags in Race To Match the Soviets In Rocket Launchers
8. 0.490, 06/14/90, Rescue of Satellite By Space Agency To Cost $90 Million

Sec. 9.1.1

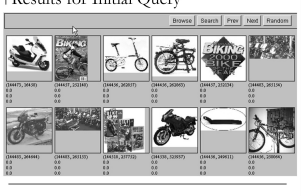
Relevance Feedback: Example

- Image search engine
  http://nayana.ece.ucsb.edu/imsearch/imsearch.html



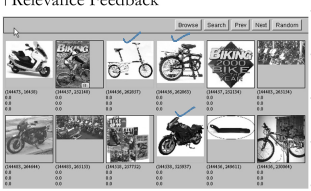
Sec. 9.1.1

Results for Initial Query



Sec. 9.1.1

Relevance Feedback



Sec. 9.1.1

Results after Relevance Feedback

2. Pseudo Relevance Feedback

aka Blind Relevance Feedback

better than global analysis

↳ provides a method for automatic local analysis

↳ it automates the manual part of relevance feedback so user gets improved performance w/o extended interaction

3. Indirect Relevance Feedback

aka implicit feedback

↳ uses direct sources of evidence rather than explicit feedback

↳ implicit feedback is less reliable than explicit feedback but, more useful than pseudo feedback

Pseudo < implicit < explicit

PROs

↳ as users are often reluctant to provide explicit feedback

↳ it is easy to collect implicit feedback that too in large quantities for a high volume system
eg web search engine

Relevance Feedback in vector spaces

↳ we can modify the query based on relevance feedback and apply standard vector space model

↳ use only docs that were marked

Centroid

↳ center of mass of a set of points

$$\vec{\mu}(C) = \frac{1}{|C|} \sum_{d \in C} \vec{d}$$

(C: set of documents)

*Recall in high-dim space docs are represented as points

ROCCHIO ALGORITHM

↳ it incorporates relevance feedback into into vector space model

$$\vec{q}_{opt} = \arg\max \left[\cos(\vec{q}, \vec{\mu}(C_r)) - \cos(\vec{q}, \vec{\mu}(C_{nr})) \right]$$

C_{nr}: set of non Rel doc

↳ Optimal query vector for seperating Rel and non Rel docs with cosine sim

$$\vec{\theta}_{opt} = \frac{1}{|C_r|} \sum_{\vec{d}_j \in C_r} \vec{d}_j - \frac{1}{N - |C_r|} \sum_{\vec{d}_j \notin C_r} \vec{d}_j$$

$\vec{\theta}_{opt}$: Optimal query; C_r: set of Rel doc vectors; N: collection size

CONS

↳ unrealistic as we dont know Rel docs

ROCCHIO ALGORITHM: method used for relevance feedback in information retrieval systems. It aims to improve search results by adjusting the query based on feedback from users about which documents are relevant and non-relevant.

β: relavent data

γ: non relavent data

ROCCHIO ALGORITHM (SMART)

1971

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

weights: hand chosen or set experimentally

$\vec{q}_m$: modified query vector; $\vec{q}_0$: original query vector; D_r: set of known Rel Doc Vectors; D_{nr}: set of known IrRel Doc Vectors

↳ new query moves towards Rel docs and away from irRel docs

↳ Trade off α vs β/γ

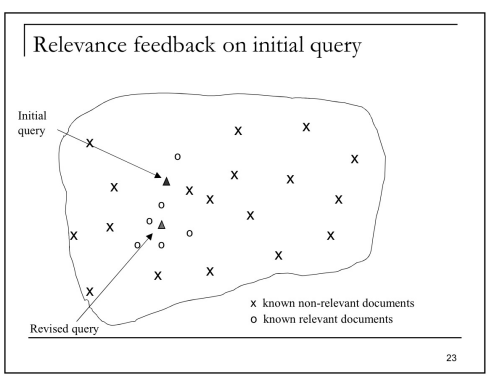
↳ if we have a lot of judged docs

↳ we want higher β/γ

↳ otherwise we want α

↳ Term weight can go negative

↳ so we ignore negative term weights



Exercise 9.3 (book)

Suppose that a user's initial query is **cheap CDs cheap DVDs extremely cheap CDs**. The user examines two documents, $d1$ and $d2$. She judges $d1$, with the content ***CDs cheap software cheap CDs*** relevant and $d2$ with content ***cheap thrills DVDs*** nonrelevant. Assume that we are using direct term frequency (with no scaling and no document frequency). There is no need to length-normalize vectors. Using Rocchio relevance feedback as in Equation (9.3) what would the revised query vector be after relevance feedback? Assume *alpha*= 1, *beta*= 0.75, *gamma*= 0.25.

Solution

	cheap	CDs	DVDs	extremely	software	thrills
q_0	3	2	1	1	0	0
d_1	2	2	0	0	1	0
d_2	1	0	1	0	0	1

Using Rocchio algorithm, $q_m = q_0 + 0.75*d_1 - 0.25*d2$. Negative weights are set to 0.

q_m	4.25	3.5	0.75	1	0.75	0

$$q_m = (1)(q_0) + 0.75(d_1) - 0.25(d_2)$$?

Relevance Feedback: Assumptions

- A1: User has sufficient knowledge for initial query.

Sec. 9.1.3

Violation of A1

- User does not have sufficient initial knowledge.
- Examples:
  - Misspellings (Brittany Speers).
  - Cross-language information retrieval (hígado).
  - Mismatch of searcher's vocabulary vs. collection vocabulary
    - Cosmonaut/astronaut

solve alone. Cases where relevance feedback alone is not sufficient include:

- Misspellings. If the user spells a term in a different way to the way it is spelled in any document in the collection, then relevance feedback is unlikely to be effective. This can be addressed by the spelling correction techniques of Chapter 3.
- Cross-language information retrieval. Documents in another language are not nearby in a vector space based on term distribution. Rather, documents in the same language cluster more closely together.
- Mismatch of searcher's vocabulary versus collection vocabulary. If the user searches for laptop but all the documents use the term notebook computer, then the query will fail, and relevance feedback is again most likely ineffective.

- A2: Relevance prototypes are "well-behaved".
  - Term distribution in relevant documents will be similar
  - Term distribution in non-relevant documents will be different from those in relevant documents
    - Either: All relevant documents are tightly clustered around a single prototype.
    - Or: There are different prototypes, but they have significant vocabulary overlap.
    - Similarities between relevant and irrelevant documents are small

Sec. 9.1.3

Violation of A2

- There are several relevance prototypes.
- Examples:
  - Burma/Myanmar
  - Contradictory government policies
  - Pop stars that worked at Burger King
- Often: instances of a general concept
- Good editorial content can address problem
  - Report on contradictory government policies

Excite Relevance Feedback

↳ only 4% of users used relevance feedback option
  expressed as "More like this" next to each result
↳ but about 70% users only looked at first page of results and didn't pursue things further
  ↳ so 4% is about 1/8 of people extending search
↳ relevance feedback improved results 2/3 of the time

Relevance Feedback Problems

1. Long queries are inefficient for typical IR engine
   ↳ Long response time for user
   ↳ High cost for retrival system
   → Partial solution → only reweigh certain prominent terms
   ↳ Perhaps top 20 by term frequency
2. Users often reluctant to provide explicit feedback
3. It's often harder to understand why a particular doc was retrived after relevance feedback is applied

Relevance Feedback Approaches

↳ Term weighting w/o query expansion

↳ Query Expansion w/o term weighting

↳ Query Expansion with term weighting

Relevance Feedback – Approaches

- Noreault (1979) tried an entirely different approach to relevance feedback. After users had selected a relevant document(s), he used this (these) document(s) as a "new query," effectively ranking (using the cosine correlation as implemented in the SIRE system) all the documents in the collection against this document(s). The top 30 of these retrieved documents were then added to those initially selected by the original query. He found on average that this added 5.1 new relevant documents per query. This feedback method would work with any type of retrieval system such as Boolean systems, provided some method existed for selecting related documents, as the query itself is not modified during the retrieval process.

Relevance Feedback – Approaches

- Attar and Fraenkel (1981) used an approach based on local feedback only. They produced an ordered list of the terms in all the documents retrieved in a first iteration search, with the list ordered by the frequency of a term in that set and by its "distance" from the initial query. These terms were shown to a user for selection as new query terms or sifted by an automatic procedure for addition to the query. They found that both expert and nonexpert users could correctly select terms from the suggested list, but that their automatic sifting mechanism could not do this well. Note that terms are selected from all retrieved documents, not just relevant ones, so that this technique could be used as a feedback technique for queries retrieving no relevant documents on the first iteration.

Relevance Feedback – Approaches

- Dillon et al. (1983) used a hybrid approach to provide relevance feedback to a Boolean environment. They devised a new query based only on terms from previously retrieved documents, with terms weighted by a formula similar to the term precision formula used by Salton (but with significant differences). The weighted terms were not used, however, to produce a ranked list of retrieved documents, but were used to automatically construct a revised Boolean query. Results suggest that this method can be effective if very careful attention is paid to the construction of the revised Boolean query.

Relevance Feedback Summary

Relevance feedback summary

- Relevance feedback has been shown to be very effective at improving relevance of results.
- Its successful use requires queries for which the set of relevant documents is medium to large.
- Full relevance feedback is often onerous for the user, and its implementation is not very efficient in most IR systems.

Relevance feedback summary

- Other use of relevance feedback
  - Following a changing information need (e.g., names of car models of interest change over time)
  - Maintaining an information filter (e.g., for a news feed).
  - Active learning (deciding which examples it is most useful to know the class of to reduce annotation costs).

Query Expansion vs Relevance Feedback

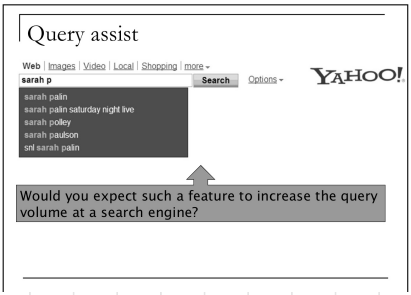
Good/bad search term

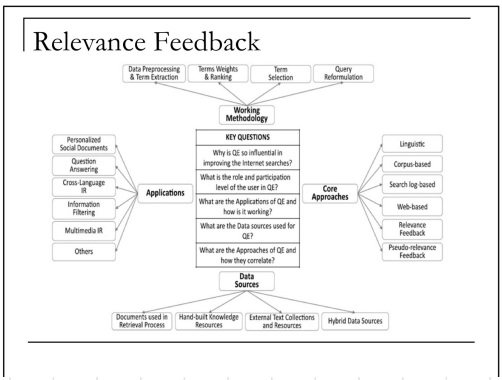
↳ users give additional input on words/phrases

↳ helps in extending user queries with related terms in order to solve lexical gap problem in IR

Rel/non Rel

↳ user gives additional input on documents which is used to reweight terms in docs





↳ HOW TO AUGMENT USER QUERY

1. **Manual Thesauras**

   e.g. MedLine: Physician
   Syn: doc, doctor, MD, medico

   ↳ can be query rather than just synonyms

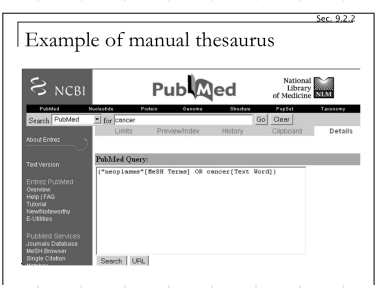
2. **Global Analysis** → Static: of all docs in collection

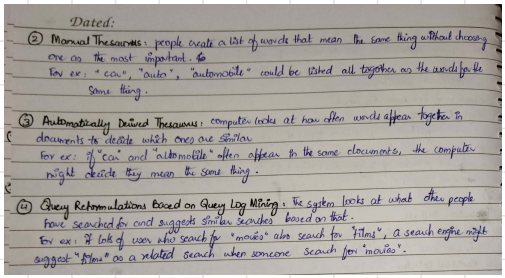
   ↳ automatically derived thesauras

   ↳ refinements based on query log mining

3. **Local Analysis** → dynamic

   ↳ Analysis of docs in result set





Dated:

(2) Manual Thesaurus: people create a list of words that mean the same thing without choosing one as the most important. For ex: "car", "auto", "automobile" could be listed all together as the words for the same thing.

(3) Automatically Derived Thesaurus: computer looks at how often words appear together in documents to decide which ones are similar. For ex: if "car" and "automobile" often appear in the same documents, the computer might decide they mean the same thing.

(4) Query Reformulations based on Query Log Mining: The system looks at what other people have searched for and suggests similar searches based on that. For ex: if lots of users who search for "movies" also search for "films", a search engine might suggest "films" as a related search when someone search for "movies".

Conclusion

- Relevance Feedback is an important part of learning the query intent from a user.

hand, out of vocabulary is also a problem in effectively classifying the test cases.

d. Give at least 2 assumptions of the Binary Independence Model (BIM) for IR? [2.5]

Binary Independence Model (BIM) for IR has the following two string assumptions:

1. The Binary Independence Assumption is that documents are binary vectors. That is, only the presence or absence of terms in documents are recorded.

2. Terms are independently distributed in the set of relevant documents and they are also independently distributed in the set of irrelevant documents.

simplify the expression for probabilistic IR

g. Every term in the document is a random variable. What does this assumption signify in probability ranking principle?

Every term is coming from a sample space Σ^* over a finite symbol set λ. The probabilities are computed using a random variable x=t for a term.

i. How term independence assumptions are incorporated in both vector space model and probabilistic information retrieval?

In Vector Space Model (VSM), every term is considered as a different dimension hence there is no relationship between terms. They are independent. While in probabilistic information retrieval, we assume that the probability of each term is independent of other term, hence simplify the computation required for probability of relevance.

Vector Space Model	Probabilistic Model	La
Idea: The document and query are represented in vector space of terms appears in both. Similarity is define as the distance between document and query vectors.	Idea: Address the uncertainty in the query. Try to estimate a query term probability to appear in a relevant document. The documents are presented to user with decreasing probability of terms appears in the relevant documents.	Id It pr a r an pr ge m rel alr do co
Opportunity: - Based on strong mathematical rigor. - Rank documents based on the similarity of query and documents. - Partial and full matching possible.	Opportunity: - Based on general idea of probability theory. - Rank documents based on their probability of relevance to a query. - Partial and full match possible.	Op - - -
Limitations: - No contextual information in the model - Model tuning based on statistical term weighting - High dimensionality	Limitations: - Strong assumption of term independence - Term with zero probabilities need some kind of smoothing	Li - -

j. What are the differences between standard vector space tf-idf weighting and the BIM probabilistic retrieval model (in the case where no document relevance information is available)?

When no relevance is given we estimate the probability of a term belong to non-relevant document is $u(t)= \log (N-df_t / df_t)$ where df_t is the document frequency of term t and as most documents are non-relevant we have $u(t) = \log (N/df_t)$ which is similar to tf*idf factor of vector space model. Hence the two are similar.

Answer the following questions briefly using 1-3 lines of answer book. Be precise, accurate and to the point, only answer genuine query in the question. Each question is of 2 marks.

a. How tf*idf weighting scheme assigned weights to most frequent, less frequent and rear terms of documents in vector space model for IR?

The tf-idf weighting scheme assigns term t a weight in document d that is
1. highest when t occurs many times within a small number of documents (thus lending high discriminating power to those documents);
2. lower when the term occurs fewer times in a document, or occurs in many documents (thus offering a less pronounced relevance signal);
3. lowest when the term occurs in virtually all documents.

Relevance Feedback is difficult to explain to a common user.

e. Explain how Rocchio's algorithm is used to fetch pages similar to a user's sample provided page (pages like this options)?

In order to use Rocchio's algorithm for retrieving "Find pages like this one". There is only on relevant feedback item and that is the sample page. We can completely ignore the query (q_0), and we do not want to apply negative judgment and hence alpha (α)= gamma(γ) =0; This implies beta (β)= 1 for the modified query.

c. Discuss the pros and cons of implicit (indirect) vs. explicit (direct) feedbacks.

Implicit(Indirect) feedback	Explicit (Direct) feedback
- Implicit (indirect) feedback does not bother user for explicit actions. It is fast and can be possible for large IR system. - It is less reliable and possibly introduce a problem of query drift.	- Explicit (Direct) feedback requires user to marks document relevant. It is slow process and does not scale to large systems. - It is more reliable and generally save from query drift.

d. In Rocchio algorithm what will be the condition (values of α, β and γ) for which original query ($\mathbf{q_o}$) is more close to centroid of relevant documents than modified query ($\mathbf{q_m}$).

The original query ($\mathbf{q_o}$) is more close to centroid of relevant documents than modified query ($\mathbf{q_m}$), if the value β is very small and γ is very large and we keep α=1.

modified query (q_m), if the value β is very small and γ is very large and we keep α=1.

e. What is probability ranking principle? Explain.

Let $R_{d,q}$ be an indicator random variable that says whether d is relevant with respect to a given query q. That is, it takes on a value of 1 when the document is relevant and 0 otherwise. The obvious order in which to present documents to the user is to rank documents by their estimated probability of relevance with respect to the information need: P(R = 1|d, q). This is the basis of the Probability Ranking Principle (PRP).

f. In Binary Independence Model (BIM) what does P(R=1/ x, q) + P(R=0/ x, q) =1 assumption represent?

The assumption that a document "d" represented in vector space of terms as "x" is either belong to set of relevant document or set of non-relevant document given a fixed query. [A document is either relevant or non-relevant to a query]

(q_o) to expand the result-set.

i. What does the assumption "a query term is equally likely to be present or absent from a randomly pick relevant document" in BIM signify?

With this assumption we have the probability that a query term appears in a relevant document is P (t) =0.5, and a query term absent from relevant document is also Q (t) =0.5 where Q (t) = 1-P (t) [only for relevant collection against a query] which is practical for 1-P (t) and P (t) cancel out each other and thus simplify the expression for BIM.

j. What does the assumption "if a term is not in a query, it is equally likely to occur in relevant and non-relevant collection" in BIM signify?

With this assumption all non-query terms get equal likely value for both relevant and non-relevant sub-collections hence cancel out each other and only the query terms that appears in documents are used for actual calculation in BIM expression.

probability of P (R / di,q) from a collection. {Probability of relevance given the query and document}.

i. Every term in the document is a random variable. What does this assumption signify in probability ranking principle?

Every term is coming from a sample space Σ^* over a finite symbol set λ. The probabilities are computed using a random variable x=t for a term. The assumption signify that are terms in a language are equal likely and hence it is treated as a random variable.

j. What is meant by (i) empty-document and (ii) by assumption P(empty-doc/R=1) = P(empty-doc/R=0) =0?

An empty document is a document that contains no selected vocabulary word hence it contains zero feature of interest. An empty document has an equal likely chances of part of a relevant or non-relevant collection for a fixed query. The assumption to P(empty-doc/R=1) = P(empty-doc/R=0) is to simplify the computation and the model.

d2= {w3 w2}
d3= {w1 w2}
Relevant Documents = d1, d3
Non-Relevant Document= d2
q = {w1 w2 w3}
Using the Probability Ranking Principle (PRP) rank these documents. Using Prior from the given information.
Assume a document d4= {w4 w2 w3} check whether it is relevant or not?

D	w1	w2	w3	w4	w5	R/NR
d1	2	2	0	0	1	R
d2	0	1	1	0	0	NR
d3	1	1	0	0	0	R

A)

B)

P(R) = 2/3

P(NR) = 1/3

P(R | d4) = P(R) * {P(w2 | R) * P(w3 | R) * P(w4 | R)}

= 2/3 * { 2.5/3 * 0.5/3 * .5/3} = 5/324 = 0.01543209877

P(NR | d4) = P(NR) * {P(w2 | NR) * P(w3 | NR) * P(w4 | NR)}

= 1/3 * { 1.5/2 * 1.5/2 * .5/2} = 3/64 = 0.046875

Since P(NR | d4) > P(R | d4), therefore d4 is not relevant.

Probabilistic Model: Information Retrieval as Classification

So how does calculating P(D|R) get us to the probability of relevance? It turns out there is a relationship between P(R|D) and P(D|R) that is expressed by **Bayes' Rule** :

$$P(R|D) = \frac{P(D|R)P(R)}{P(D)}$$

where P(R) is the a priori probability of relevance (in other words, how likely any document is to be relevant), and P(D) acts as a normalizing constant.

classify a document as relevant if P(D|R)P(R) > P(D|NR)P(NR). This is the same as classifying a document as relevant if:

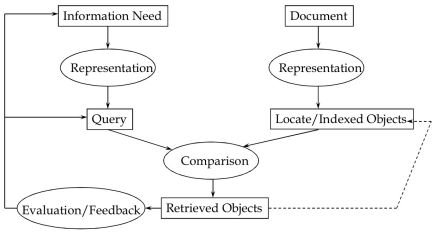
likelihood ratio

$$\frac{P(D|R)}{P(D|NR)} > \frac{P(NR)}{P(R)}$$

Agenda

- Why use Probabilities in IR
- Basic Probability
- Probability for IR
- Probability Ranking Principle (PRP)
- Binary Independence Model (BIM)
- Relevance feedback in PIR
- Conclusion

Basic Information Retrieval Process



Why Probability in IR

↳ IR deals with uncertain info
  ↳ Probability is a measure of uncertainty
↳ Probabilistic Ranking Principles
  ↳ Provable
  ↳ minimization of risk
↳ Probabilistic Inference
  ↳ To justify your decision

Basic Probability

↳ Probability is the study of randomness and uncertainty
↳ It can be defined as

$$P = \frac{\text{favorable events}}{\text{total events}}$$

Axioms of Probability

↳ Probability is always $0 \leq P \leq 1$
↳ $P(\Omega) = 1$ (Ω → sample space)

- If $A_1, A_2, \ldots A_n$ is a partition of A, then
  $P(A) = P(A_1) + P(A_2) + \ldots + P(A_n)$
  ($A_1, A_2, \ldots A_n$ is called a partition of A if $A_1 \cup A_2 \cup \ldots \cup A_n = A$ and $A_1, A_2, \ldots A_n$ are mutually exclusive.)

Properties of Probability

1. $P(A^c) = 1 - P(A)$
2. If $A \subset B$ then $P(A) \leq P(B)$
3. $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
4. $P(A \cup B \cup C) = P(A) + P(B) + P(C) - P(A \cap B) - P(B \cap C) - P(A \cap C) + P(A \cap B \cap C)$
5. $P(A, B, C) = P(A)\,P(B)\,P(C)$ → Independent Probability
6. $P(A|B)P(B) = P(B|A)P(A)$ → Bayes Theorem
7. $P(\bar{A}, B) = P(B|\bar{A})P(\bar{A})$ → How compliments work
8. $P(B) = P(A, B) + P(\bar{A}, B)$ → Partition Rule
9. $P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \left[\frac{P(B|A)}{\sum_{X \in \{A, \bar{A}\}} P(B|X)P(X)}\right] P(A)$

6. $P(A) \geq 0 \ \forall A \in \Omega$
7. $P(\Omega) = 1$
8. $A_i \cap A_j = \emptyset \ \forall_{i,j} \Rightarrow P(\cup_{i=1}^{n} A_i) = \sum_{i=1}^{n} P(A_i)$
9. $P(\emptyset) = 0$

→ Ω: Sample space
↳ has all possible outcomes
↳ A in Ω: Set of outcome of interest

Basic Probability Theory

Bayes' Rule for inverting conditional probabilities:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \left[\frac{P(B|A)}{\sum_{X \in \{A, \bar{A}\}} P(B|X)P(X)}\right] P(A)$$

Can be thought of as a way of updating probabilities:

- Start off with prior probability $P(A)$ (initial estimate of how likely event A is in the absence of any other information)
- Derive a posterior probability $P(A|B)$ after having seen the evidence B, based on the likelihood of B occurring in the two cases that A does or does not hold

Odds of an event provide a kind of multiplier for how probabilities change:

Odds: $O(A) = \frac{P(A)}{P(\bar{A})} = \frac{P(A)}{1 - P(A)}$

Odds: $O(A) = \frac{P(A)}{P(\bar{A})} = \frac{P(A)}{1 - P(A)}$

Conditional Probability

$P(A|B) = \frac{P(A \cap B)}{P(B)}$

A will occur given that B already

BAYES RULE

$P(A_i|B) = \frac{P(B|A_i)\,P(A_i)}{\sum_{i=1}^{n} P(B|A_i)P(A_i)}$

$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$

↳ $P(A|B) = \frac{P(A \cap B)}{P(B)}$ $\quad P(B|A) = \frac{P(A \cap B)}{P(A)}$

↳ $P(A|B)P(B) = P(B|A)P(A)$

Probability Ranking Principle

↳ collection of documents

↳ user issues a query

↳ a set of docs needs to be returned

Q) In what order to present docs to user

↳ Intuitively, want the 'best' docs to be first, second best - second etc

↳ need a formal way to judge the "goodness" of docs w.r.t queries

Idea: Probability of relevance of the docs w.r.t. query

↳ If a reference revival system response to each req is a ranking of the docs in the collection in order of decreasing probability of relevance to the user who submitted the request.

↳ where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose

↳ the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data.

Probability Ranking Principle

Let x be a document in the collection.
Let R represent **relevance** of a document w.r.t. given (fixed) query and let NR represent **non-relevance.**

Need to find p($R|x$) - probability that a retrieved document x is **relevant.**

$$p(R|x) = \frac{p(x|R)p(R)}{p(x)}$$

$$p(NR|x) = \frac{p(x|NR)p(NR)}{p(x)}$$

p(R),p(NR) - prior probability of retrieving a (non) relevant document

p($x|R$), p($x|NR$) - probability that if a relevant (non-relevant) document is retrieved, it is x.

Probability Ranking Principle

$$p(R|x) = \frac{p(x|R)p(R)}{p(x)}$$

$$p(NR|x) = \frac{p(x|NR)p(NR)}{p(x)}$$

Ranking Principle (Bayes' Decision Rule):

If p($R|x$) > p($NR|x$) then x is relevant, otherwise x is not relevant

$$P(R|x) = \frac{P(x|R)\,P(R)}{P(x)}$$

→ P(R), P(NR)
↳ prior probability of retrieving a non relevant docs

$$P(NR|x) = \frac{P(x|NR)\,P(NR)}{P(x)}$$

if $P(R|x) > P(NR|x)$
then x is relevant
otherwise x is not relevant

Assumption

↳ A1: One Random variable for each term(word)

↳ A2: d(w) are mutually independence given R

↳ A3: $P(0|R=1) = P(0|R=0) = 0$

↳ A4: If the word is not in the query
it is equally likely to occur in relavent and non relevant docs
Practically, we only need to calculate probabilities of common words in query and docs

↳ A5: On avg, a query word will occur in half the relevant docs.
Practical pw and (1-pw) will cancel out

↳ A6: Non-relavent set approximately by collection as a whole. → Most docs are non-relevant

↳ Let we have docs collection D, with |D| number of docs in it

↳ Consider a fixed query "Q"

↳ Let N_1 is the docs set from D, which is relavent to Q
Let N_0 is the docs set D which is non-relavent to Q
If we know the prior for R and NR we can use this for calculation. {P(R) and P(NR)}

↳ Let x is a RV for a term appearing in a doc.

↳ Let x be the docs vectors for a doc d,
its relevance to query Q can be given as

$$P\left(\frac{R}{d\to x, q}\right) = \frac{P\left(\frac{x}{R}\right) * P(R)}{P(x)} \qquad P:\left(\frac{NR}{d\to x, q}\right) = \frac{P\left(\frac{x}{NR}\right) * P(NR)}{P(x)}$$

↳ In order to estimate $x \to (w_1, w_2, \ldots w_n)$
we need to define two probabilities

↳ $p_w = \frac{N_1(w) + 0.5}{N_1 + 1.0}$ $\qquad q_w = \frac{N_0(w) + 0.5}{N_0 + 1.0}$

Probability Ranking Principle

- Now, $P(R/ d\to x, Q) = \{P(x/R) * P(R)\} / P(x)$
  $P(R/ d_i, Q) = \{P(d_i\to x/R) * P(R)\} / P(d_i\to x)$
- The retrieval value status (RSV) is given by Odd of R/NR probabilities (rank not probabilities- practically we have)

$$RSV(d_i, Q) = \frac{P(R/ d_i, Q) = \{P(d_i\to x/R) * P(R)\} / P(d_i\to x)}{P(NR/ d_i, Q) = \{P(d_i\to x/NR) * P(NR)\} / P(d_i\to x)}$$

Dated:

* BINARY INDEPENDENCE MODEL (BIM):

The model estimates the probability of a document being relevant or non-relevant given its term incidence vector $\bar{x}$ (presence or absence of terms) in a document and the query vector $\bar{q}$.

$P(R=1 \mid \bar{x}, \bar{q})$ → probability of a document being relevant
$P(R=0 \mid \bar{x}, \bar{q})$ → probability of a document being non-relevant.

$$P(R=1 \mid \bar{x}, \bar{q}) = \frac{P(\bar{x} \mid R=1, \bar{q}) P(R=1 \mid \bar{q})}{P(\bar{x} \mid \bar{q})}$$

∴ the probability that the document is relevant to your query.

$P(x \mid R=1, q)$ → how likely it is to see the words in the document (x) if the document is indeed relevant to your question (q).

$P(R=1 \mid q)$ → how likely it is for any document to be relevant to your query (q) before looking at the specific words in the document.

$$\boxed{P(R=1 \mid \bar{x}, \bar{q}) + P(R=0 \mid \bar{x}, \bar{q}) = 1}$$

$$RSV(d_i,Q) = \frac{P(R/\ d_i,Q) = \{P(d_i\text{->}x/R) * P(R)\} / P(d_i\text{->}x)}{P(NR/\ d_i,Q) = \{P(d_i\text{->}x/NR) * P(NR)\} / P(d_i\text{->}x)}$$

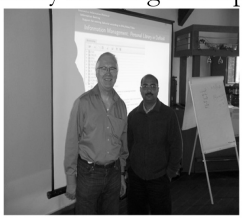
$$RSV(R/di,Q) = \pi_{(\text{for all } w \in di)} \{ p_w * (1-q_w) / q_w * (1-p_w) \}$$

Cost in PRP

- Lets associate cost to PRP

$$C_0 \cdot P(R=0|\vec{d}) - C_1 \cdot P(R=1|\vec{d}) \leq C_0 \cdot P(R=0|\vec{d}') - C_1 \cdot P(R=1|\vec{d}')$$

Probability Ranking Principle



The Gerard Salton Award is presented by the Association for Computing Machinery (ACM) Special Interest Group on Information Retrieval (SIGIR) 2012 - Norbert Fuhr, University of Duisburg-Essen: "Information Retrieval as Engineering Science."

Binary Independence Model

Given a query q, we wish to order returned documents by descending $P(R = 1|d, q)$. Under the BIM, this is modeled as ordering by $P(R = 1|\vec{x}, \vec{q})$. Rather than estimating this probability directly, because we are interested only in the ranking of documents, we work with some other quantities which are easier to compute and which give the same ordering of documents. In particular, we can rank documents by their odds of relevance (as the odds of relevance is monotonic with the probability of relevance). This makes things easier, because we can ignore the common denominator in (11.8), giving:

$$O(R|\vec{x},\vec{q}) = \frac{P(R=1|\vec{x},\vec{q})}{P(R=0|\vec{x},\vec{q})} = \frac{\frac{P(R=1|\vec{q})P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|\vec{q})}}{\frac{P(R=0|\vec{q})P(\vec{x}|R=0,\vec{q})}{P(\vec{x}|\vec{q})}} = \frac{P(R=1|\vec{q})}{P(R=0|\vec{q})} \cdot \frac{P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|R=0,\vec{q})}$$

Binary Independence Model

The left term in the rightmost expression of Equation (11.10) is a constant for a given query. Since we are only ranking documents, there is thus no need for us to estimate it. The right-hand term does, however, require estimation, and this initially appears to be difficult: How can we accurately estimate the probability of an entire term incidence vector occurring? It is at this point that we make the *Naive Bayes conditional independence assumption* that the presence or absence of a word in a document is independent of the presence or absence of any other word (given the query):

$$\frac{P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|R=0,\vec{q})} = \prod_{t=1}^{M} \frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})}$$

So:

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t=1}^{M} \frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})}$$

Binary Independence Model

Since each x_t is either 0 or 1, we can separate the terms to give:

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=1} \frac{P(x_t=1|R=1,\vec{q})}{P(x_t=1|R=0,\vec{q})} \cdot \prod_{t:x_t=0} \frac{P(x_t=0|R=1,\vec{q})}{P(x_t=0|R=0,\vec{q})}$$

Henceforth, let $p_t = P(x_t = 1|R = 1, \vec{q})$ be the probability of a term appearing in a document relevant to the query, and $u_t = P(x_t = 1|R = 0, \vec{q})$ be the probability of a term appearing in a nonrelevant document. These quantities can be visualized in the following contingency table where the columns add to 1:

	document	relevant ($R = 1$)	nonrelevant ($R = 0$)
Term present	$x_t = 1$	p_t	u_t
Term absent	$x_t = 0$	$1 - p_t$	$1 - u_t$

Binary Independence Model

Let us make an additional simplifying assumption that terms not occurring in the query are equally likely to occur in relevant and nonrelevant documents: that is, if $q_t = 0$ then $p_t = u_t$. (This assumption can be changed, as when doing relevance feedback in Section 11.3.4.) Then we need only consider terms in the products that appear in the query, and so,

$$O(R|\vec{q},\vec{x}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t}{u_t} \cdot \prod_{t:x_t=0,q_t=1} \frac{1-p_t}{1-u_t}$$

The left product is over query terms found in the document and the right product is over query terms not found in the document.

Binary Independence Model

We can manipulate this expression by including the query terms found in the document into the right product, but simultaneously dividing through by them in the left product, so the value is unchanged. Then we have:

$$O(R|\vec{q},\vec{x}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} \cdot \prod_{t:q_t=1} \frac{1-p_t}{1-u_t}$$

The left product is still over query terms found in the document, but the right product is now over all query terms. That means that this right product is a constant for a particular query, just like the odds $O(R|\vec{q})$. So the only quantity that needs to be estimated to rank documents for relevance to a query is the left product. We can equally rank documents by the logarithm of this term, since log is a monotonic function. The resulting quantity used for ranking is called the *Retrieval Status Value* (RSV) in this model:

$$RSV_d = \log \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} = \sum_{t:x_t=q_t=1} \log \frac{p_t(1-u_t)}{u_t(1-p_t)}$$

Binary Independence Model

So everything comes down to computing the *RSV*. Define c_t:

$$c_t = \log \frac{p_t(1-u_t)}{u_t(1-p_t)} = \log \frac{p_t}{(1-p_t)} + \log \frac{1-u_t}{u_t}$$

The c_t terms are log odds ratios for the terms in the query. We have the odds of the term appearing if the document is relevant ($p_t/(1-p_t)$) and the odds of the term appearing if the document is nonrelevant ($u_t/(1-u_t)$). The *odds ratio* is the ratio of two such odds, and then we finally take the log of that quantity. The value will be 0 if a term has equal odds of appearing in relevant and nonrelevant documents, and positive if it is more likely to appear in relevant documents. The c_t quantities function as term weights in the model, and the document score for a query is $RSV_d = \sum_{x_t=q_t=1} c_t$. Operationally, we sum them in accumulators for query terms appearing in documents, just as for the vector space model calculations discussed in Section 7.1 (page 135). We now turn to how we estimate these c_t quantities for a particular collection and query.

Binary Independence Model

Probability estimates in theory

For each term t, what would these c_t numbers look like for the whole collection? (11.19) gives a contingency table of counts of documents in the collection, where df_t is the number of documents that contain term t:

	documents	relevant	nonrelevant	Total
Term present	$x_t = 1$	s	$\mathrm{df}_t - s$	df_t
Term absent	$x_t = 0$	$S - s$	$(N - \mathrm{df}_t) - (S - s)$	$N - \mathrm{df}_t$
	Total	S	$N - S$	N

Using this, $p_t = s/S$ and $u_t = (\mathrm{df}_t - s)/(N - S)$ and

$$c_t = K(N, \mathrm{df}_t, S, s) = \log \frac{s/(S-s)}{(\mathrm{df}_t - s)/((N - \mathrm{df}_t) - (S - s))}$$

Binary Independence Model

	documents	relevant	nonrelevant	Total
Term present	$x_t = 1$	s	$\mathrm{df}_t - s$	df_t
Term absent	$x_t = 0$	$S - s$	$(N - \mathrm{df}_t) - (S - s)$	$N - \mathrm{df}_t$
	Total	S	$N - S$	N

Using this, $p_t = s/S$ and $u_t = (\mathrm{df}_t - s)/(N - S)$ and

$$c_t = K(N, \mathrm{df}_t, S, s) = \log \frac{s/(S-s)}{(\mathrm{df}_t - s)/((N - \mathrm{df}_t) - (S - s))}$$

To avoid the possibility of zeroes (such as if every or no relevant document has a particular term) it is fairly standard to add $\frac{1}{2}$ to each of the quantities in the center 4 terms of (11.19), and then to adjust the marginal counts (the totals) accordingly (so, the bottom right cell totals $N + 2$). Then we have:

$$\hat{c}_t = K(N, \mathrm{df}_t, S, s) = \log \frac{(s + \frac{1}{2})/(S - s + \frac{1}{2})}{(\mathrm{df}_t - s + \frac{1}{2})/(N - \mathrm{df}_t - S + s + \frac{1}{2})}$$

Pseudo Relevance Feedback

- Retrieval Status Value (RSV)

	documents	relevant	nonrelevant	total
term present	$x_t = 1$	s	$\mathrm{df}_t - s$	df_t
term absent	$x_t = 0$	$S - s$	$(N - \mathrm{df}_t) - (S - s)$	$N - \mathrm{df}_t$
	total	S	$N - S$	N

Using this, $p_t = s/S$ and $u_t = (\mathrm{df}_t - s)/(N - S)$ and

$$RSV_d = \log \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} = \sum_{t:x_t=q_t=1} \log \frac{p_t(1-u_t)}{u_t(1-p_t)}.$$

$$c_t = \log \frac{p_t(1-u_t)}{u_t(1-p_t)} = \log \frac{p_t}{(1-p_t)} + \log \frac{1-u_t}{u_t}.$$

$$c_t = K(N, \mathrm{df}_t, S, s) = \log \frac{s/(S-s)}{(\mathrm{df}_t - s)/((N - \mathrm{df}_t) - (S - s))}.$$

Relevance Feedback in Prob. IR

1. Guess initial estimates of p_t and u_t. This can be done using the probability estimates of the previous section. For instance, we can assume that p_t is constant over all x_t in the query, in particular, perhaps taking $p_t = \frac{1}{2}$.
2. Use the current estimates of p_t and u_t to determine a best guess at the set of relevant documents $R = \{d : R_{d,t} = 1\}$. Use this model to retrieve a set of candidate relevant documents, which we present to the user.
3. We interact with the user to refine the model of R. We do this by learning from the user relevance judgments for some subset of documents V. Based on relevance judgments, V is partitioned into two subsets: $VR = \{d \in V, R_{d,t} = 1\} \subset R$ and $VNR = \{d \in V, R_{d,t} = 0\}$, which is disjoint from R.
4. We reestimate p_t and u_t on the basis of known relevant and nonrelevant documents. If the sets VR and VNR are large enough, we may be able to estimate these quantities directly from these documents as maximum likelihood estimates:

$$p_t = |VR_t|/|VR|$$

Relevance Feedback in Prob. IR

where VR_t is the set of documents in VR containing x_t. In practice, we usually need to smooth these estimates. We can do this by adding $\frac{1}{2}$ to both the count $|VR_t|$ and to the number of relevant documents not containing the term, giving:

$$p_t = \frac{|VR_t| + \frac{1}{2}}{|VR| + 1}.$$

However, the set of documents judged by the user (V) is usually very small, and so the resulting statistical estimate is quite unreliable (noisy), even if the estimate is smoothed. So it is often better to combine the new information with the original guess in a process of Bayesian updating. In this case we have:

$$p_t^{(k+1)} = \frac{|VR_t| + \kappa p_t^{(k)}}{|VR| + \kappa}.$$

Relevance Feedback in Prob. IR

Here $p_t^{(k)}$ is the k^{th} estimate for p_t in an iterative updating process and is used as a Bayesian prior in the next iteration with a weighting of κ. Relating this equation back to Equation (11.4) requires a bit more probability theory than we have presented here (we need to use a beta distribution prior, conjugate to the Bernoulli random variable X_t). But the form of the resulting equation is quite straightforward: Rather than uniformly distributing pseudocounts, we now distribute a total of κ pseudocounts according to the previous estimate, which acts as the prior distribution. In the absence of other evidence (and assuming that the user is perhaps indicating roughly five relevant or nonrelevant documents) then a value of around $\kappa = 5$ is perhaps appropriate. That is, the prior is strongly weighted so that the estimate does not change too much from the evidence provided by a very small number of documents.

Relevance Feedback in Prob. IR

5. Repeat the above process from Step 2, generating a succession of approximations to R and hence p_t, until the user is satisfied.

Pseudo Relevance Feedback

1. Assume initial estimates for p_t and u_t as above.
2. Determine a guess for the size of the relevant document set. If unsure, a conservative (too small) guess is likely to be best. This motivates use of a fixed size set V of highest ranked documents.
3. Improve our guesses for p_t and u_t. We choose from the methods of Equations (11.23) and (11.25) for reestimating p_t, except now based on the set V instead of VR. If we let V_t be the subset of documents in V containing x_t and use add $\frac{1}{2}$ smoothing, we get:

$$p_t = \frac{|V_t| + \frac{1}{2}}{|V| + 1}$$

and if we assume that documents that are non retrieved are nonrelevant then we can update our u_t estimates as:

$$u_t = \frac{\mathrm{df}_t - |V_t| + \frac{1}{2}}{N - |V| + 1}.$$

Pseudo Relevance Feedback

4. Go to Step 2 until the ranking of the returned results converges.

Once we have a real estimate for p_t, then the c_t weights used in the RSV value look almost like a tf–idf value. For instance, using Equation (11.18), Equation (11.22), and Equation (11.26), we have:

$$c_t = \log\left[\frac{p_t}{1-p_t} \cdot \frac{1-u_t}{u_t}\right] \approx \log\left[\frac{|V_t| + \frac{1}{2}}{|V| - |V_t| + 1} \cdot \frac{N}{\mathrm{df}_t}\right].$$

But things aren't quite the same: $p_t/(1-p_t)$ measures the (estimated) proportion of relevant documents that the term t occurs in, not term frequency. Moreover, if we apply log identities:

$$c_t = \log \frac{|V_t| + \frac{1}{2}}{|V| - |V_t| + 1} + \log \frac{N}{\mathrm{df}_t}$$

we see that we are now *adding* the two log-scaled components rather than multiplying them.

Probability Ranking Principle (PRP)

↳ sum of all CTs → in BM1

CONS
↳ computationally demanding
↳ not very good

PRP with Relevance

steps

↳ $P(R) = \frac{\text{Rel}}{\text{Total}}$ ← Prior

$P(NR) = \frac{\text{Non R}}{\text{Total}}$

↳ $P(t) = \frac{W_1 + 0.5}{R + 1}$ (count of term in R doc; total R docs)

$U(t) = \frac{W_1 + 0.5}{NR + 1}$ (count of term in NR doc; total NR docs; 1 → Laplace Smoothing)

↳ $RSV(R, x, y) = P(R) \times \frac{P(t)}{1 - P(t)} \times \frac{1 - U(t)}{U(t)}$ (common terms of x and y)

$RSV(NR, x, y)\ P(NR) \times \frac{1 - U(t)}{U(t)} \times \frac{P(t)}{1 - P(t)}$

↳ if R > NR
mon R

Question No. 2 **[Time: 15 Min] [Marks: 10]**

Consider the following given document-collection and a query.

d1: virus microscopic organism → R
d2: virus infects cell organism → R
d3: virus infects computers → NR
d4: tiny virus security → NR
q: virus tiny organism

$P(R) = \frac{R}{T} = \frac{2}{4} = 0.5$

$P(NR) = \frac{R}{T} = \frac{2}{4} = 0.5$

From the system d1 and d2 are relevant to this query and d3 and d4 are not. Using the probabilistic model for IR with given relevance, rank these documents using probability ranking principle. Show all intermediates steps of calculations. Compute whether the document **"d5: virus computer virus"** is relevant or not.

Words	cell	computers	infects	microscopic	organism	security	tiny	virus
R $N_1(W)$	1	0	1	1	2	0	0	2
P(t)	0.5	0.1667	0.5	0.5	0.833	0.1667	0.1667	0.833
NR $N_0(W)$	0	1	1	0	0	1	1	2
U(t)	0.1667	0.5	0.5	0.1667	0.1667	0.5	0.5	0.833

$\frac{2+0.5}{2+1} = P(t)$

$\frac{0+0.5}{2+1} = U(t)$

Now, we want to rank all documents with the probability of relevance with the query.
P(R=1/ di,q) for i=1,2,3, and 4.

RSV for d1= P(R=1/ d1,q) $=^{rank}$ (0.833/0.1667)X(0.833/0.1667)X0.5 = 12.48

RSV for d2= P(R=1/ d2,q) $=^{rank}$ (0.833/0.1667)X(0.833/0.1667)X0.5 = 12.48

RSV for d3= P(R=1/ d3,q) $=^{rank}$ (0.833/0.1667)X(0.1667/0.833)X0.5 = 0.5

RSV for d4= P(R=1/ d4,q) $=^{rank}$ (0.1667/0.833 x 0.5/0.5) X(0.833/0.1667 X 0.1667/0.833)X0.5 = 0.1

$P(R) \times \frac{0.833}{1-0.833} \times \frac{1-0.833}{0.833} \times \frac{0.833}{1-0.833} \times \frac{1-0.1667}{0.1667}$

only do terms that use common b/w q and d

Hence ranking will be either d1, d2, d3 and d4 or d2, d1, d3 and d4. → get ranking

Now, knowing the class of d5;

P(R=1/ d5,q) = ((0.833/0.1667)X(0.1667/0.833))X0.5 = 0.5

$\frac{0.833}{0.1667} \times \frac{0.1667}{0.833} \times 0.5$ → common b/w q and ds

P(R=0/ d5,q) = ((0.1667/0.833)X (0.833/0.1667))X0.5 = 0.5

P(R=1/ d5,q) is not greater than P(R=0/ d5,q) hence d5 is non-relevant.

$\frac{0.1667}{0.833} \times \frac{0.833}{0.1667} 0.5$ → common b/w q and ds

PRP W/O RELEVANCE

Application 3

↳ relevance isn't given

↳ estimate using collection size

$$P(N_w) = \frac{N - N_w + 0.5}{N_w + 0.5}$$

$RSV(d1,q)$: → combon terms b/w q and d1

Consider the document collection given in Question 2 once again:
D={d1,d2,d3,d4}

d1= mary had a little lamb
d2= little lamb mary had
d3= mary went with lamb
d4= lamb went with mary

when NR and R not given

Now use probabilistic model of IR to represent these documents in probabilistic vector form. **Using the probabilities term values of each term, find the similarity of each document with the query "mary went". Which document is most similar to the query?**

$\frac{4-1+0.5}{1+0.5}$ $\frac{4-2+0.5}{2+0.5}$

N= number of documents = 4
$P(N_w) = (N - N_w + 0.5) / (N_w + 0.5)$

	a	had	lamb	little	mary	went	with
N_w	1	2	2	1	4	2	2
$P(N_w)$	2.33	1	0.11	1	0.11	1	1

→ df

RSV(d1,q) = products of all terms common between document and query = (0.11)
RSV(d2,q) = (0.11)
RSV(d3,q) = (0.11)
RSV(d4,q) = (0.11)

all documents are at the same rank (equally relevant.)

$R(D_1,q) = 0.11$ (mary)
$R(D_2,q) = 0.11$ (lamb)
$R(d3,q) = 0.11 \times 1 = 0.11$ (mary went)
$R(d4,q) = 0.11 \times 1 = 0.11$

Question No. 3 **[Time: 15 Min] [Marks: 10]**

Consider the following given document-collection and a query.

d1: w1 w2 w4 w6
d2: w1 w2 w7 w3
d3: w8 w5 w4 w5 w6
q: w2 w5 w6

$\frac{1+0.5}{3+1}$

Using the probabilistic model for IR, Probability Ranking Principle (PRP) without given relevance, rank these documents using PRP. Show all intermediates steps of calculations in tabular form.

Using the Probability Ranking Principle (PRP) without given relevance, we will rank the document collection is decreasing order of P(R=1/d,q). Each term P(wi) = (N(w) + 0.5) / (N+1)

Words	W1	W2	W3	W4	W5	W6	W7	W8
N(wi)	2	2	1	2	1	2	1	1
P(wi)	0.625	0.625	0.375	0.625	0.375	0.625	0.375	0.375

↳ $\frac{2+0.5}{3+1}$

Now, we want to rank all documents with the probability of relevance with the query. P(R=1/ di,q) for i=1,2 and 3.

RSV for d1= P(R=1/ d1,q) $=^{rank}$ { common terms w2 and w6} = (0.625/0.375) X (0.625/0.375) (w2, w6)

$=^{rank}$ 2.78

RSV for d2= P(R=1/ d2,q) $=^{rank}$ { common terms w2 } = (0.625/0.375) (w2)

$=^{rank}$ 1.67

RSV for d3= P(R=1/ d3,q) $=^{rank}$ { common terms w5 and w6 } = (0.375/0,625) x (0.375/0,625) x (0.625/0.375) (w5, w6)

$=^{rank}$ 0.6

Thus ranking order will be D1, D2 and D3.

IR APPROACHES

1. Boolean Model

↳ based on notion of sets

↳ exact match

↳ no rankings on retrived d

2. Vector Space Model (VSM)

↳ based on geometry, the notion of vectors in high dimensional space

↳ d are ranked based on similarity to q

↳ best/partial match

↳ df

3. Probabilistic Models (PM)

↳ based on notion of probabilities present in the text

↳ d are ranked based on probability that the text present in d based on q terms

↳ Best/Partial match

LM vs. VSM

- LMs vs. vector space model: commonalities
  - Term frequency is directly in the model. → but not scaled in LM
  - Probabilities are inherently "length-normalized". → cosine similarity does similar in VSM
  - Mixing document and collection frequencies has an effect similar to idf. terms rare in collection, but common in some d will have higher ranking
- LMs vs. vector space model: differences
  - LMs: based on probability theory
  - Vector space: based on similarity, a geometric/ linear algebra notion
  - Collection frequency vs. document frequency
  - Details of term frequency, length normalization etc.

4. Language Models (LM)

↳ based on notion of probabilities and process for generating text

↳ d are ranked based on probability that they generated the query

↳ Best/Partial match

↳ cf

LANGUAGE MODELS (LM)

↳ Basic idea

Users come up with good queries (by thinking of words that would likely appear in relavant document) and use those words in queries

- The language modeling approach to IR directly models that idea: a document is a good match to a query if the document model is likely to generate the query, which will in turn happen if the document contains the query words often.

LM

↳ builds a Probabilistic language model M_d from each document d

↳ ranks me d based on me me probability of me model generating me query

$P(q|M_d)$

↳ calculates MLE using multinomial

VS

PM

↳ Overtly models me probability of revalance of a document d to a query q

$P(R=1|q;d)$

↳ calculates MLE using word counts

Language Model

- A language model is a function that puts a probability measure over strings drawn from some vocabulary. That is, for a language model M over an alphabet S:

$$\sum_{s \in \Sigma^*} P(s) = 1$$

- One simple kind of language model is equivalent to a probabilistic finite automaton consisting of just a single node with a single probability distribution over producing different terms, so that $\sum_{t \in V} P(t) = 1$,

Example 1 P= P(words) x P(stopping)

Example 12.1: To find the probability of a word sequence, we just multiply the probabilities that the model gives to each word in the sequence, together with the probability of continuing or stopping after producing each word. For example,

from said not toad likes frog

$$P(\text{frog said that toad likes frog}) = (0.01 \times 0.03 \times 0.04 \times 0.01 \times 0.02 \times 0.01) \times (0.8 \times 0.8 \times 0.8 \times 0.8 \times 0.8 \times 0.8 \times 0.2) \approx 0.000000000001573$$

→ stopping

As you can see, the probability of a particular string/document is usually a very small number! Here we stopped after generating *frog* the second time. The first line of numbers are the term emission probabilities, and the second line gives the probability of continuing or stopping after generating each word. An explicit stop probability is needed for a finite automaton

Example 1

the	0.2
a	0.1
frog	0.01
toad	0.01
said	0.03
likes	0.02
that	0.04
…	…

$P(\text{STOP}|q_1) = 0.2$

Figure 12.2 A one-state finite automaton that acts as a unigram language model. We show a partial specification of the state emission probabilities.

Example 2 P= P(words)

Example 12.2: Suppose, now, that we have two language models M_1 and M_2, shown partially in Figure 12.3. Each gives a probability estimate to a sequence of terms, as already illustrated in Example 12.1. The language model that gives the higher probability to the sequence of terms is more likely to have generated the term sequence. This time, we will omit STOP probabilities from our calculations. For the sequence shown, we get:

s	frog	said	that	toad	likes	that	dog
M_1	0.01	0.03	0.04	0.01	0.02	0.04	0.005
M_2	0.0002	0.03	0.04	0.0001	0.04	0.04	0.01

$P(s|M_1) = 0.00000000000048$
$P(s|M_2) = 0.000000000000000384$

and we see that $P(s|M_1) > P(s|M_2)$. We present the formulas here in terms of products of probabilities, but, as is common in probabilistic applications, in practice it is usually best to work with sums of log probabilities (cf. page 239).

Example 2

model M_1		model M_2	
the	0.2	the	0.15
a	0.1	a	0.12
frog	0.01	frog	0.0002
toad	0.01	toad	0.0001
said	0.03	said	0.03
likes	0.02	likes	0.04
that	0.04	that	0.04
dog	0.005	dog	0.01
cat	0.003	cat	0.015
monkey	0.001	monkey	0.002
…	…	…	…

Figure 12.3 Partial specification of two unigram language models.