Traditional IR models

Jian-Yun Nie

Main IR processes

- Last lecture: Indexing – determine the important content terms

- Next process: Retrieval
  - How should a retrieval process be done?
    - Implementation issues: using index (e.g. merge of lists)
    - (*) What are the criteria to be used?
  - Ranking criteria
    - What features?
    - How should they be combined?
    - What model to use?

Cases

- one-term query:

  The documents to be retrieved are those that include the term

  - Retrieve the inverted list for the term
  - Sort in decreasing order of the weight of the word

- Multi-term query?
  - Combining several lists
  - How to interpret the weight?
  - How to interpret the representation with all the indexing terms for a document?

  (IR model)

What is an IR model?

- Define a way to represent the contents of a document and a query
- Define a way to compare a document representation to a query representation, so as to result in a document ranking (score function)
- E.g. Given a set of weighted terms for a document
  - Should these terms be considered as forming a Boolean expression? a vector? …
  - What do the weights mean? a probability, a feature value, …
  - What is the associated ranking function?

Plan

- This lecture
  - Boolean model
  - Extended Boolean models
  - Vector space model
  - Probabilistic models
    - Binary Independent Probabilistic model
    - Regression models
- Next week
  - Statistical language models

Early IR model – Coordinate matching score (1960s)

- Matching score model
  - Document D = a set of weighted terms
  - Query Q = a set of non-weighted terms

$$R(D,Q) = \sum_{t_i \in Q} w(t_i, D)$$

- Discussion
  - Simplistic representation of documents and queries
  - The ranking score strongly depends on the term weighting in the document
    - If the weights are not normalized, then there will be great variations in *R*

IR model - Boolean model

- Document = Logical conjunction of keywords (not weighted)
- Query = any Boolean expression of keywords
- $R(D, Q) = D \rightarrow Q$

e.g. $D_1 = t_1 \wedge t_2 \wedge t_3$ (the three terms appear in D)

$D_2 = t_2 \wedge t_3 \wedge t_4 \wedge t_5$

$Q = (t_1 \wedge t_2) \vee (t_3 \wedge \neg t_4)$

$D_1 \rightarrow Q$, thus $R(D_1, Q) = 1$.

but $D_2 \not\rightarrow Q$, thus $R(D_2, Q) = 0$.

Properties

- Desirable
  - $R(D,Q \wedge Q)=R(D,Q \vee Q)=R(D,Q)$
  - $R(D,D)=1$
  - $R(D,Q \vee \neg Q)=1$
  - $R(D,Q \wedge \neg Q)=0$

- Undesirable
  - $R(D,Q)=0$ or 1

Boolean model

- Strengths
  - Rich expressions for queries
  - Clear logical interpretation (well studied logical properties)
    - Each term is considered as a logical proposition
    - The ranking function is determine by the validity of a logical implication
- Problems:
  - *R* is either 1 or 0 (unordered set of documents)
    - many documents or few/no documents in the result
    - No term weighting in document and query is used
  - Difficulty for end-users for form a correct Boolean query
    - E.g. documents about *kangaroos* **and** *koalas*
    - *kangaroo* $\wedge$ *koala ?*
    - *kangaroo* $\vee$ *koala ?*
    - Specialized application (Westlaw in legal area)
- Current status in Web search
  - Use Boolean model (ANDed terms in query) for a first step retrieval
  - Assumption: There are many documents containing all the query terms → find a few of them

Extensions to Boolean model (for document ranking)

- D = {…, (t_i, w_i), …}: weighted terms
- Interpretation:
  - Each term or a logical expression defines a fuzzy set
  - (t_i, w_i): D is a member of class t_i to degree w_i.
  - In terms of fuzzy sets, membership function: $\mu_{ti}(D)=w_i$

A possible Evaluation:

$R(D, t_i) = \mu_{ti}(D) \in [0,1]$

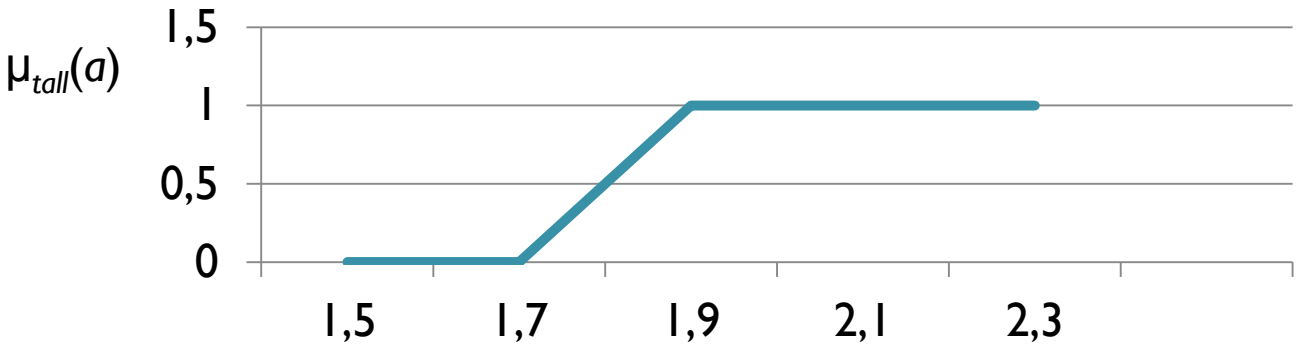
$R(D, Q_1 \wedge Q_2) = \mu_{Q_1 \wedge Q_2}(D) = \min(R(D, Q_1), R(D, Q_2))$;

$R(D, Q_1 \vee Q_2) = \mu_{Q_1 (Q_2}(D) = \max(R(D, Q_1), R(D, Q_2))$;

$R(D, \neg Q_1) = \mu_{\neg Q_1}(D) = 1 - R(D, Q_1)$.

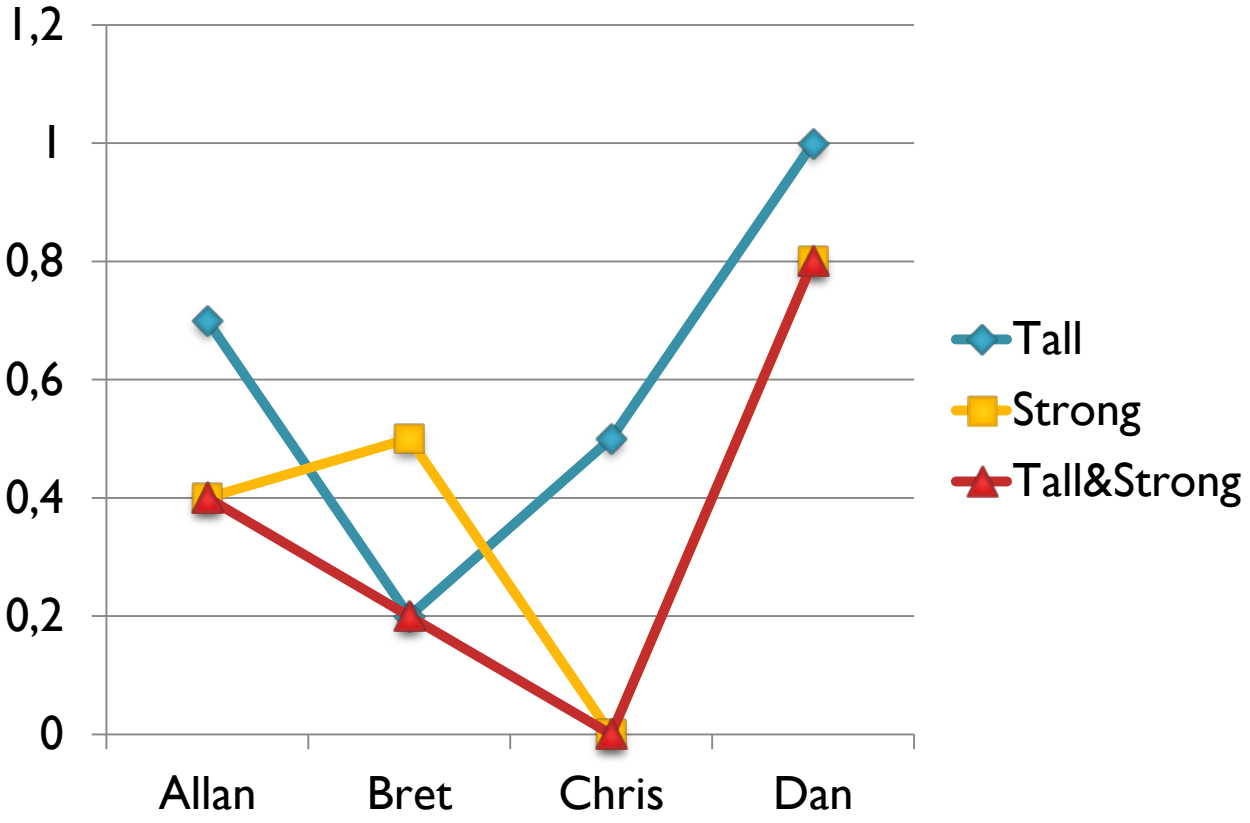
Recall on fuzzy sets

- Classical set
  - *a* belongs to a set *S*: $a \in S$,
  - or no: $a \notin S$
- Fuzzy set
  - *a* belongs to a set *S* to some degree ($\mu_S(a) \in [0,1]$)
  - E.g. someone is *tall*



Recall on fuzzy sets

- Combination of concepts



Extension with fuzzy sets

- Can take into account term weights
- Fuzzy sets are motivated by fuzzy concepts in natural language (tall, strong, intelligent, fast, slow, …)

- Evaluation reasonable?
  - min and max are determined by one of the elements (the value of another element in some range does not have a direct impact on the final value) - counterintuitive
  - Violated logical properties
    - $\mu_{A \vee \neg A}(.) \neq 1$
    - $\mu_{A \wedge \neg A}(.) \neq 0$

Alternative evaluation in fuzzy sets

$R(D, t_i) = \mu_{ti}(D) \in [0,1]$

$R(D, Q_1 \wedge Q_2) = R(D, Q_1) * R(D, Q_2)$;

$R(D, Q_1 \vee Q_2) = R(D, Q_1) + R(D, Q_2) - R(D, Q_1) * R(D, Q_2)$;

$R(D, \neg Q_1) = 1 - R(D, Q_1)$.

- The resulting value is closely related to both values
- Logical properties
  - $\mu_{A \vee \neg A}(.) \neq 1$ $\mu_{A \wedge \neg A}(.) \neq 0$
  - $\mu_{A \vee A}(.) \neq \mu_A(.)$ $\mu_{A \wedge A}(.) \neq \mu_A(.)$
- In practice, better than min-max
- Both extensions have lower IR effectiveness than vector space model

IR model - Vector space model

- Assumption: Each term corresponds to a dimension in a vector space
- Vector space = all the keywords encountered

  $<t_1, t_2, t_3, \ldots, t_n>$
- Document

  $D = < a_1, a_2, a_3, \ldots, a_n>$

  a_i = weight of t_i in D
- Query

  $Q = < b_1, b_2, b_3, \ldots, b_n>$

  b_i = weight of t_i in Q
- R(D,Q) = Sim(D,Q)

Matrix representation

Document space

	t_1	t_2	t_3	…	t_n
D_1	a_{11}	a_{12}	a_{13}	…	a_{1n}
D_2	a_{21}	a_{22}	a_{23}	…	a_{2n}
D_3	a_{31}	a_{32}	a_{33}	…	a_{3n}
…					
D_m	a_{m1}	a_{m2}	a_{m3}	…	a_{mn}
Q		b_1	b_2	b_3	… b_n

← Term vector space

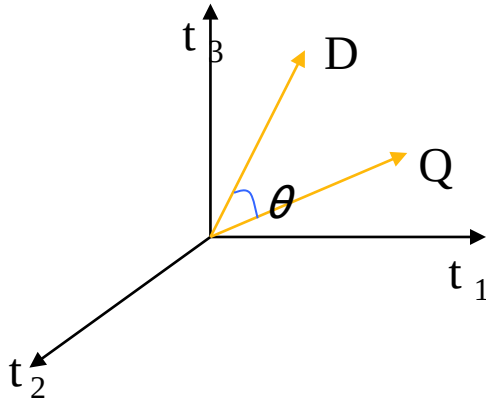
Some formulas for Sim

Dot product

$$Sim(D,Q) = D \bullet Q = \sum_i (a_i * b_i)$$

Cosine

$$Sim(D,Q) = \frac{\sum_i (a_i * b_i)}{\sqrt{\sum_i a_i^2 * \sum_i b_i^2}}$$



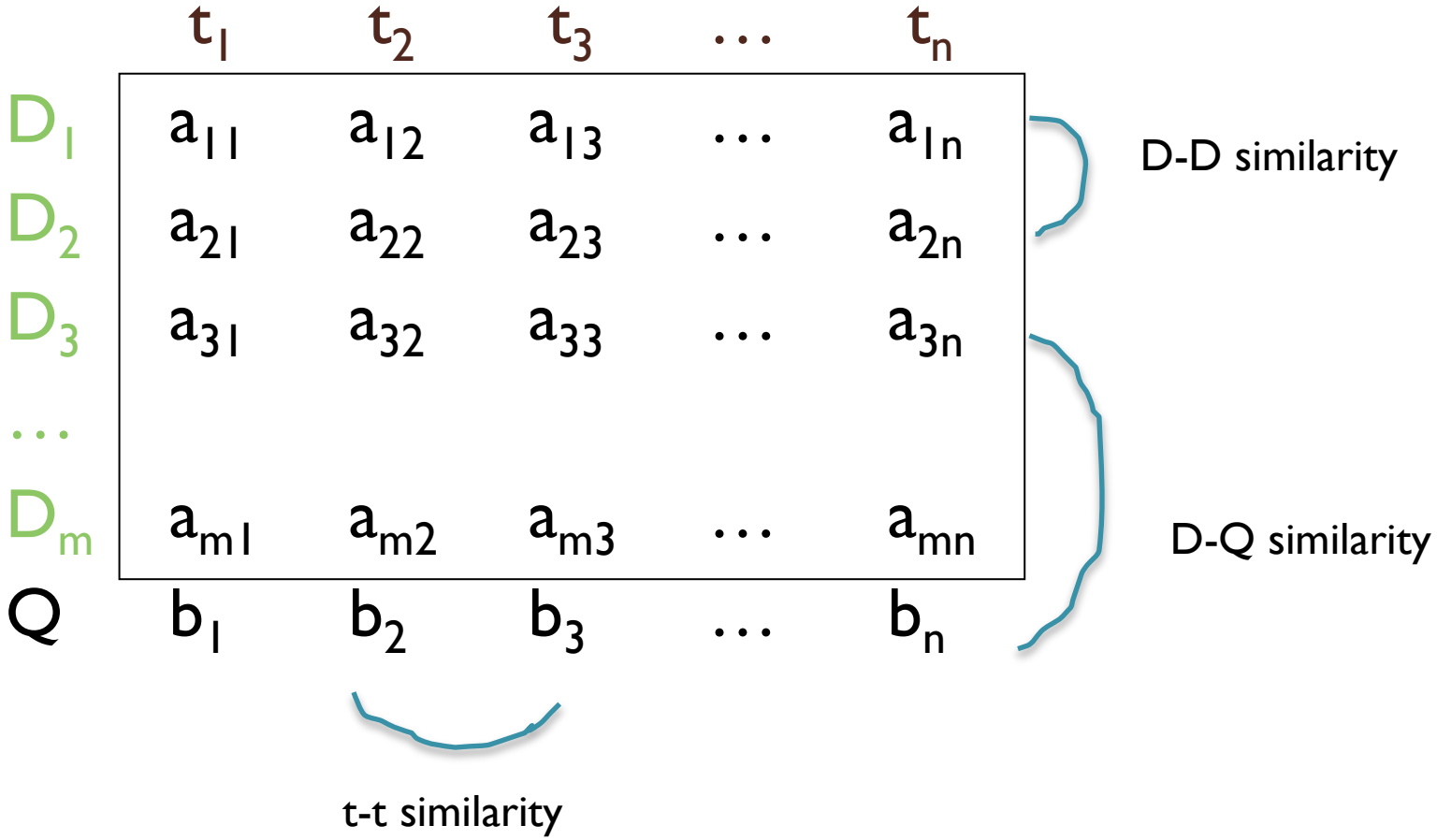
Dice

$$Sim(D,Q) = \frac{2\sum_i (a_i * b_i)}{\sum_i a_i^2 + \sum_i b_i^2}$$

Jaccard

$$Sim(D,Q) = \frac{\sum_i (a_i * b_i)}{\sum_i a_i^2 + \sum_i b_i^2 - \sum_i (a_i * b_i)}$$

Document-document, document-query and term-term similarity



Euclidean distance

$$\left|d_j - d_k\right| = \sqrt{\sum_{i=1}^{n}\left(d_{i,j} - d_{i,k}\right)^2}$$

- When the vectors are normalized (length of 1), the ranking is the same as cosine similarity. (Why?)

Implementation (space)

- Matrix is very sparse: a few 100s terms for a document, and a few terms for a query, while the term space is large (>100k)

- Stored as:

  $D_1 \rightarrow \{(t_1, a_1), (t_2,a_2), \ldots\}$

  $t_1 \rightarrow \{(D_1,a_1), \ldots\}$

(recall possible compressions: Y code)

Implementation (time)

- The implementation of VSM with dot product:
  - Naïve implementation: Compare Q with each D
  - O(m*n): m doc. & n terms
  - Implementation using inverted file:

Given a query = $\{(t_1,b_1), (t_2,b_2), (t_3,b_3)\}$:

1. find the sets of related documents through inverted file for each term
2. calculate the score of the documents to each weighted query term

$$(t_1,b_1) \rightarrow \{(D_1,a_1*b_1), \ldots\}$$

3. combine the sets and sum the weights ($\sum$)

- O(|t|*|Q|*log(|Q|)):
  - |t|<<m (|t|=avg. length of inverted lists),
  - |Q|*log|Q|<<n (|Q|=length of the query)

Pre-normalization

- Cosine:

$$Sim(D,Q) = \frac{\sum_i (a_i * b_i)}{\sqrt{\sum_j a_i^2 * \sum_j b_i^2}} = \sum_i \frac{a_i}{\sqrt{\sum_j a_j^2}} \frac{b_i}{\sqrt{\sum_j b_j^2}}$$

- use $1/\sqrt{\sum_j a_j^2}$ and $1/\sqrt{\sum_j b_j^2}$ to normalize the weights after indexing of document and query
- Dot product

  (Similar operations do not apply to Dice and Jaccard)

Best p candidates

- Can still be too expensive to calculate similarities to all the documents (Web search)
- → p best
- Preprocess: Pre-compute, for each term, its p nearest docs.
  - (Treat each term as a 1-term query.)
  - lots of preprocessing.
  - Result: "preferred list" for each term.
- Search:
  - For a $|Q|$-term query, take the union of their $|Q|$ preferred lists – call this set S, where $|S| \leq p|Q|$.
  - Compute cosines from the query to only the docs in S, and choose the top k.
  - If too few results, search in extended index

Need to pick $p>k$ to work well empirically.

Discussions on vector space model

- Pros:
  - Mathematical foundation = geometry
    - Q: How to interpret?
  - Similarity can be used on different elements
  - Terms can be weighted according to their importance (in both D and Q)
  - Good effectiveness in IR tests
- Cons
  - Users cannot specify relationships between terms
    - *world cup*: may find documents on *world* or on *cup* only
    - A strong term may dominate in retrieval
  - Term independence assumption (in all classical models)

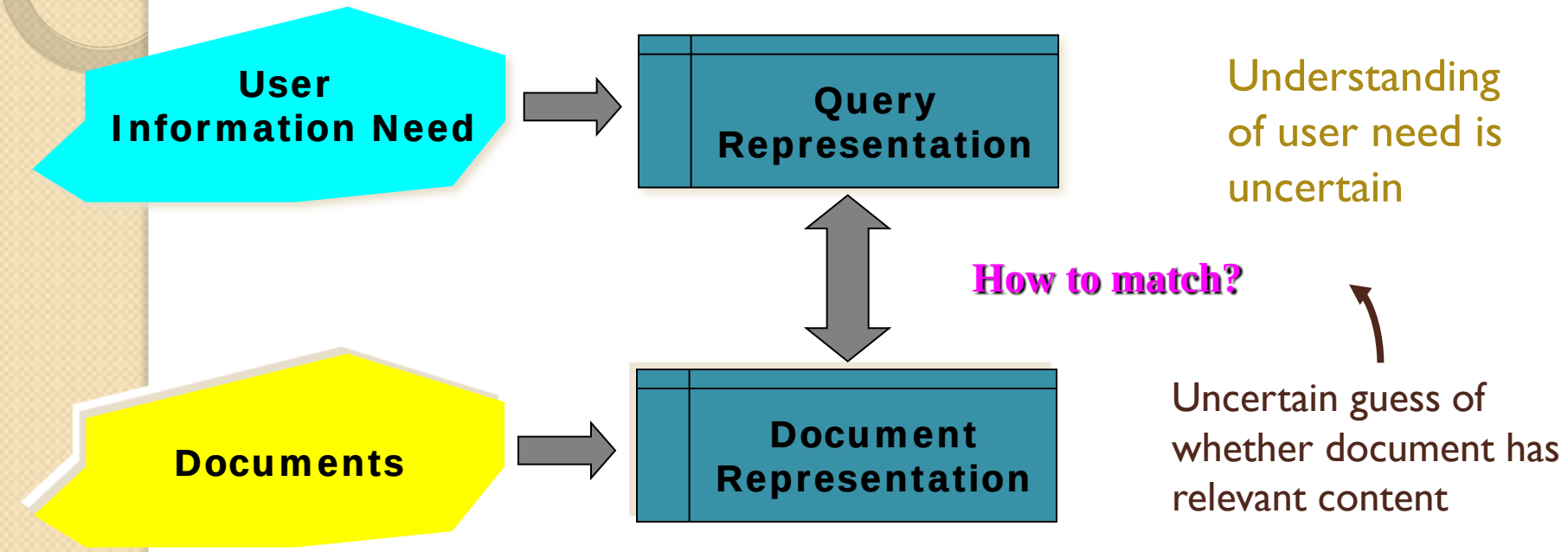
Comparison with other models

- Coordinate matching score – a special case
- Boolean model and vector space model: two extreme cases according to the difference we see between AND and OR (Gerard Salton, Edward A. Fox, and Harry Wu. 1983. Extended Boolean information retrieval. *Commun. ACM* 26, 11, 1983)
- Probabilistic model: can be viewed as a vector space model with probabilistic weighting.

Probabilistic relevance feedback

- If user has told us some relevant and some irrelevant documents, then we can proceed to build a probabilistic classifier, such as a Naive Bayes model:
  - $P(t_k|R) = |\mathbf{D}_{rk}| / |\mathbf{D}_r|$
  - $P(t_k|NR) = |\mathbf{D}_{nrk}| / |\mathbf{D}_{nr}|$
    - t_k is a term; $\mathbf{D}_r$ is the set of known relevant documents; $\mathbf{D}_{rk}$ is the subset that contain t_k; $\mathbf{D}_{nr}$ is the set of known irrelevant documents; $\mathbf{D}_{nrk}$ is the subset that contain t_k.

Why probabilities in IR?



In traditional IR systems, matching between each document and query is attempted in a semantically imprecise space of index terms.

Probabilities provide a principled foundation for uncertain reasoning.
Can we use probabilities to quantify our uncertainties?

Probabilistic IR topics

- Classical probabilistic retrieval model
  - Probability ranking principle, etc.
- (Naïve) Bayesian Text Categorization/classification
- Bayesian networks for text retrieval
- Language model approach to IR
  - An important emphasis in recent work

- *Probabilistic methods are one of the oldest but also one of the currently hottest topics in IR.*
  - *Traditionally: neat ideas, but they've never won on performance. It may be different now.*

The document ranking problem

- We have a collection of documents
- User issues a query
- A list of documents needs to be returned
- **Ranking method is core of an IR system:**
  - **In what order do we present documents to the user?**
  - We want the "best" document to be first, second best second, etc….
- **Idea: Rank by probability of relevance of the document w.r.t. information need**
  - P(relevant|document$_i$, query)

Recall a few probability basics

- For events *a* and *b*:
- Bayes' Rule

$$p(a,b) = p(a \cap b) = p(a \mid b)\,p(b) = p(b \mid a)\,p(a)$$

$$p(\overline{a} \mid b)\,p(b) = p(b \mid \overline{a})\,p(\overline{a})$$

$$p(a \mid b) = \frac{p(b \mid a)\,p(a)}{p(b)} = \frac{p(b \mid a)\,p(a)}{\sum_{x=a,\overline{a}} p(b \mid x)\,p(x)}$$

Prior

Posterior

- Odds:

$$O(a) = \frac{p(a)}{p(\overline{a})} = \frac{p(a)}{1 - p(a)}$$

The Probability Ranking Principle

> "If a reference retrieval system's response to each request is a ranking of the documents 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."

- [1960s/1970s] S. Robertson, W.S. Cooper, M.E. Maron; van Rijsbergen (1979:113); Manning & Schütze (1999:538)

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.**

R={0,1} vs. NR/R

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

$$p(R \mid x) = \frac{p(x \mid R)\,p(R)}{p(x)}$$

$$p(NR \mid x) = \frac{p(x \mid NR)\,p(NR)}{p(x)}$$

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

$$p(R \mid x) + p(NR \mid x) = 1$$

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

Probability Ranking Principle (PRP)

- Simple case: no selection costs or other utility concerns that would differentially weight errors
- ***Bayes' Optimal Decision Rule***
  - ***x*** is **relevant** iff $p(R|x) > p(NR|x)$
- PRP in action: Rank all documents by $p(R|x)$
- Theorem:
  - Using the PRP is optimal, in that it minimizes the loss (Bayes risk) under 1/0 loss
  - Provable if all probabilities correct, etc. [e.g., Ripley 1996]

Probability Ranking Principle

- More complex case: retrieval costs.
  - Let *d* be a document
  - *C* - cost of retrieval of relevant document
  - *C'* - cost of retrieval of non-relevant document
- Probability Ranking Principle: if

$$C \cdot p(R|d) + C' \cdot (1 - p(R|d)) \leq C \cdot p(R|d') + C' \cdot (1 - p(R|d'))$$

for all *d' not yet retrieved*, then *d* is the next document to be retrieved

- We won't further consider loss/utility from now on

Probability Ranking Principle

- How do we compute all those probabilities?
  - Do not know exact probabilities, have to use estimates
  - Binary Independence Retrieval (BIR) – which we discuss later today – is the simplest model
- Questionable assumptions
  - "Relevance" of each document is independent of relevance of other documents.
    - Really, it's bad to keep on returning **duplicates**
  - Boolean model of relevance (relevant or irrelevant)
  - That one has a single step information need
    - Seeing a range of results might let user refine query

Probabilistic Retrieval Strategy

- Estimate how terms contribute to relevance
  - How do things like tf, df, and length influence your judgments about document relevance?
    - One answer is the Okapi formulae (S. Robertson)
- Combine to find document relevance probability
- Order documents by decreasing probability

Probabilistic Ranking

Basic concept:

"For a given query, if we know some documents that are relevant, terms that occur in those documents should be given greater weighting in searching for other relevant documents.

By making assumptions about the distribution of terms and applying Bayes Theorem, it is possible to derive weights theoretically."

Van Rijsbergen

Binary Independence Model

- Traditionally used in conjunction with PRP
- **"Binary" = Boolean**: documents are represented as binary incidence vectors of terms:
  - $\vec{x} = (x_1, \ldots, x_n)$
  - $x_i = 1$ iff term *i* is present in document *x*.
- **"Independence" :** terms occur in documents independently
- Different documents can be modeled as same vector

- Bernoulli Naive Bayes model (cf. text categorization!)

Binary Independence Model

- Queries: binary term incidence vectors
- Given query ***q***,
  - for each document ***d*** need to compute ***p(R|q,d)***.
  - replace with computing ***p(R|q,x)*** where ***x*** is binary term incidence vector representing ***d*** Interested only in ranking
- Will use odds and Bayes' Rule:

$$O(R \mid q, \vec{x}) = \frac{p(R \mid q, \vec{x})}{p(NR \mid q, \vec{x})} = \frac{\dfrac{p(R \mid q)\, p(\vec{x} \mid R, q)}{p(\vec{x} \mid q)}}{\dfrac{p(NR \mid q)\, p(\vec{x} \mid NR, q)}{p(\vec{x} \mid q)}}$$

Binary Independence Model

$$O(R \mid q, \vec{x}) = \frac{p(R \mid q, \vec{x})}{p(NR \mid q, \vec{x})} = \frac{p(R \mid q)}{p(NR \mid q)} \cdot \frac{p(\vec{x} \mid R, q)}{p(\vec{x} \mid NR, q)}$$

Constant for a given query

Needs estimation

- Using **Independence** Assumption:

$$\frac{p(\vec{x} \mid R, q)}{p(\vec{x} \mid NR, q)} = \prod_{i=1}^{n} \frac{p(x_i \mid R, q)}{p(x_i \mid NR, q)}$$

- So : $$O(R \mid q, d) = O(R \mid q) \cdot \prod_{i=1}^{n} \frac{p(x_i \mid R, q)}{p(x_i \mid NR, q)}$$

Binary Independence Model

$$O(R \mid q,d) = O(R \mid q) \cdot \prod_{i=1}^{n} \frac{p(x_i \mid R,q)}{p(x_i \mid NR,q)}$$

- Since x_i is either 0 or 1:

$$O(R \mid q,d) = O(R \mid q) \cdot \prod_{x_i=1} \frac{p(x_i=1 \mid R,q)}{p(x_i=1 \mid NR,q)} \cdot \prod_{x_i=0} \frac{p(x_i=0 \mid R,q)}{p(x_i=0 \mid NR,q)}$$

- Let $p_i = p(x_i=1 \mid R,q); \quad r_i = p(x_i=1 \mid NR,q);$

- Assume, for all terms not occurring in the query ($q_i=0$) $p_i = r_i$

Then…

This can be changed (e.g., in relevance feedback)

Binary Independence Model

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{x_i = q_i = 1} \frac{p_i}{r_i} \cdot \prod_{\substack{x_i = 0 \\ q_i = 1}} \frac{1 - p_i}{1 - r_i}$$

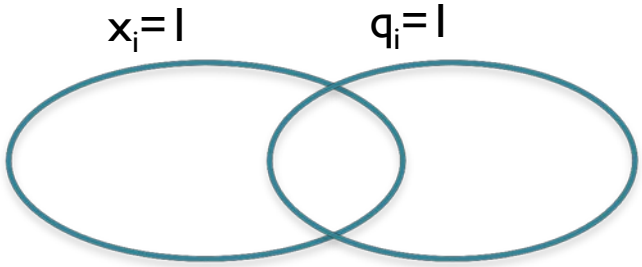
All matching terms

Non-matching query terms

$$= O(R \mid q) \cdot \prod_{x_i = q_i = 1} \frac{p_i (1 - r_i)}{r_i (1 - p_i)} \cdot \prod_{q_i = 1} \frac{1 - p_i}{1 - r_i}$$

All matching terms

All query terms



Binary Independence Model

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{x_i = q_i = 1} \frac{p_i (1 - r_i)}{r_i (1 - p_i)} \cdot \prod_{q_i = 1} \frac{1 - p_i}{1 - r_i}$$

Constant for each query

Only quantity to be estimated for rankings

- Retrieval Status Value:

$$RSV = \log \prod_{x_i = q_i = 1} \frac{p_i (1 - r_i)}{r_i (1 - p_i)} = \sum_{x_i = q_i = 1} \log \frac{p_i (1 - r_i)}{r_i (1 - p_i)}$$

Binary Independence Model

- All boils down to computing RSV.

$$RSV = \log \prod_{x_i = q_i = 1} \frac{p_i(1-r_i)}{r_i(1-p_i)} = \sum_{x_i = q_i = 1} \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$$

$$RSV = \sum_{x_i = q_i = 1} c_i; \qquad c_i = \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$$

So, how do we compute c_i's from our data ?

Binary Independence Model

- Estimating RSV coefficients.
- For each term *i* look at this table of document counts:

Documens	Relevant	Non-Relevant	Total
$x_i=1$	s	$n-s$	n
$x_i=0$	$S-s$	$N-n-S+s$	$N-n$
Total	S	$N-S$	N

- Estimates: $p_i \approx \frac{s}{S} \qquad r_i \approx \frac{(n-s)}{(N-S)}$

$$c_i \approx K(N,n,S,s) = \log \frac{s/(S-s)}{(n-s)/(N-n-S+s)}$$

$$c_i = \log \frac{(s_i+0.5)/(S-s+0.5)}{(n-s+0.5)/(N-n-S+s+0.5)}$$

Sparck-Jones-Robertson formula

Estimation – key challenge

- If non-relevant documents are approximated by the whole collection, then r_i (prob. of occurrence in non-relevant documents for query) *is* n/N and
  - $\log (1 - r_i)/r_i = \log (N - n)/n \approx \log N/n$ = IDF!
- p_i (probability of occurrence in relevant documents) can be estimated in various ways:
  - from relevant documents if know some
    - Relevance weighting can be used in feedback loop
  - constant (Croft and Harper combination match) – then just get idf weighting of terms
  - proportional to prob. of occurrence in collection
    - more accurately, to log of this (Greiff, SIGIR 1998)

Iteratively estimating p_i

1. Assume that p_i constant over all x_i in query
   - p_i = 0.5 (even odds) for any given doc
2. Determine guess of relevant document set:
   - V is fixed size set of highest ranked documents on this model (note: now a bit like tf.idf!)
3. We need to improve our guesses for p_i and r_i, so
   - Use distribution of x_i in docs in V. Let V_i be set of documents containing x_i
     - $p_i = |V_i| / |V|$
   - Assume if not retrieved then not relevant
     - $r_i = (n_i - |V_i|) / (N - |V|)$
4. Go to 2. until converges then return ranking

Probabilistic Relevance Feedback

1. Guess a preliminary probabilistic description of *R* and use it to retrieve a first set of documents V, as above.
2. Interact with the user to refine the description: learn some definite members of R and NR
3. Reestimate p_i and r_i on the basis of these
   - Or can combine new information with original guess (use Bayesian prior):

$$p_i^{(2)} = \frac{|V_i| + \kappa p_i^{(1)}}{|V| + \kappa}$$

κ is prior weight

4. Repeat, thus generating a succession approximations to *R*.

PRP and BIR

- Getting reasonable approximations of probabilities is possible.
- Requires restrictive assumptions:
  - ***term independence***
  - ***terms not in query don't affect the outcome***
  - ***Boolean representation of documents/queries/relevance***
  - ***document relevance values are independent***
- Some of these assumptions can be removed
- Problem: either require partial relevance information or only can derive somewhat inferior term weights

Removing term independence

- In general, index terms aren't independent
- Dependencies can be complex
- van Rijsbergen (1979) proposed model of simple tree dependencies
- Each term dependent on one other
- In 1970s, estimation problems held back success of this model

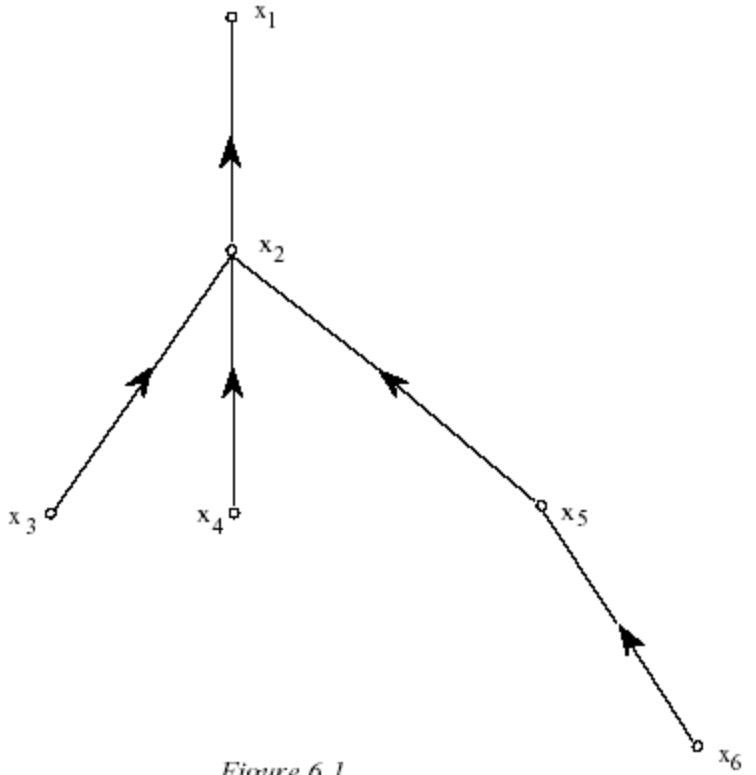


Figure 6.1.

Food for thought

- Think through the differences between standard tf.idf and the probabilistic retrieval model in the first iteration
- Think through the retrieval process of probabilistic model similar to vector space model

Good and Bad News

- Standard Vector Space Model
  - Empirical for the most part; success measured by results
  - Few properties provable
- Probabilistic Model Advantages
  - Based on a firm theoretical foundation
  - Theoretically justified optimal ranking scheme
- Disadvantages
  - Making the initial guess to get V
  - Binary word-in-doc weights (not using term frequencies)
  - Independence of terms (can be alleviated)
  - Amount of computation
  - Has never worked convincingly better in practice

BM25 (Okapi system) – Robertson et al.

Consider tf, qtf, document length

$$Score(D,Q) = \sum_{t_i \in Q} c_i \frac{(k_1+1)tf_i}{K+tf_i} \frac{(k_3+1)qtf_i}{k_3+qtf_i} + k_2 |Q| \frac{avdl - dl}{avdl + dl}$$

$$K = k_1((1-b)+b\frac{dl}{avdl-dl})$$

TF factors

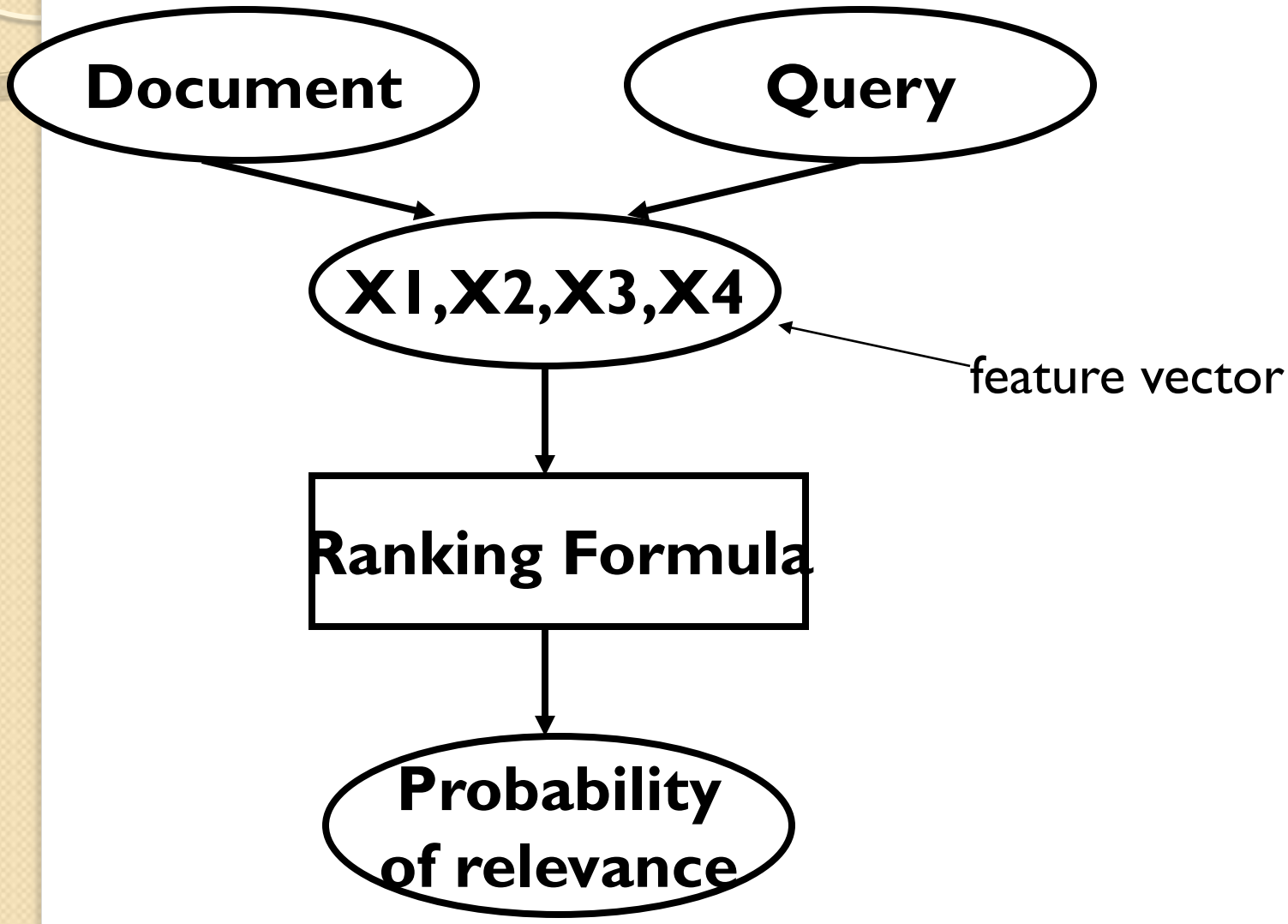
Doc. length normalization

- *k1*, *k2*, *k3*, *b*: parameters
- *qtf*: query term frequency
- *dl*: document length
- *avdl*: average document length

Regression models

- Extract a set of features from document (and query)
- Define a function to predict the probability of its relevance
- Learn the function on a set of training data (with relevance judgments)

Probability of Relevance



Regression model (Berkeley – Chen and Frey)

N	number of match terms
qf_i	query frequency of the ith match term
df_i	document frequency of the ith match term
cf_i	collection frequency of the ith match term
ql	query length
dl	document length
cf	collection length

Relevance Features

$$X_1 = \sum_{i=1}^{N} \frac{qf_i}{ql + 35}$$

$$X_2 = \sum_{i=1}^{N} \log \frac{df_i}{dl + 80}$$

$$X_3 = \sum_{i=1}^{N} \log \frac{cf_i}{cl}$$

$$X_4 = N$$

X_1, X_2, and X_3 are nomalized by $(1 + \sqrt{N})$.

Sample Document/Query Feature Vector

Relevance Features

X1	X2	X3	X4	Relevance value
0.0031	-2.406	-3.223	1	1
0.0429	-9.796	-15.55	8	1
0.0430	-6.342	-9.921	4	1
0.0195	-9.768	-15.096	6	0
0.0856	-7.375	-12.477	5	0

Representing one document/query pair in the training set

Probabilistic Model: Supervised Training

Training Data Set:
Document/Query Pairs with known relevance value.

1. Model training: estimate the unknown model parameters using training data set.

Model: Logistic Regression
Unknown parameters:
b1,b2,b3, b4

Test Data Set:
New document/query pairs

2. Using the estimated parameters to predict relevance value for a new pair of document and query.

Logistic Regression Method

- **Model**: The log odds of the relevance dependent variable is a linear combination of the independent feature variables.

$$\log it(R|X) \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4$$

relevance variable

feature variables

$$\log it(p) = \log(\tfrac{p}{1-p})$$

- **Task:** Find the optimal coefficients
- **Method:** Use statistical software package such as S-plus to fit the model to a training data set.

$$P(R|X) = \frac{1}{1+e^{-\log it(R|X)}}$$

Logistic regression

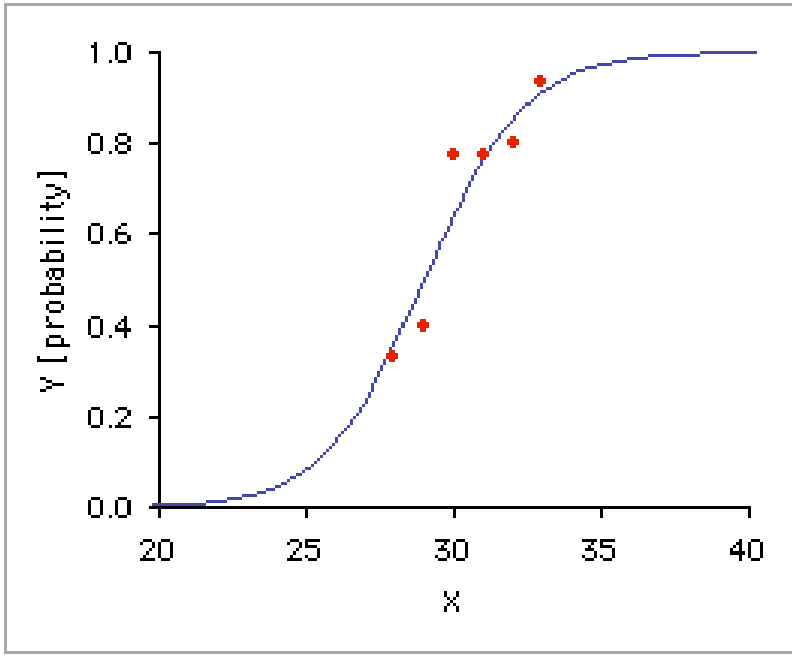
- The function to learn: $f(z)$:

$$f(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

- The variable z is usually defined as

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k$$

  - x_i = *feature variables*
  - β_i=parameters/coefficients



Document Ranking Formula

$$\log O(R \mid D,Q) = -3.51 + 37.4 \times X_1 + 0.330 \times X_2 - 0.1937 \times X_3 + 0.0929 \times X_4$$

$$X_1 = \frac{1}{1+\sqrt{N}} \sum_{i=1}^{N} \frac{qf_i}{ql+35}$$

$$X_2 = \frac{1}{1+\sqrt{N}} \sum_{i=1}^{N} \log \frac{df_i}{dl+35}$$

$$X_3 = \frac{1}{1+\sqrt{N}} \sum_{i=1}^{N} \log \frac{cf_i}{cl}$$

$$X_4 = N$$

N is the number of matching terms between document D and query Q.

Discussions

- Usually, terms are considered to be independent
  - *algorithm* independent from *computer*
  - *computer architecture:* 2 independent dimensions
- Different theoretical foundations (assumptions) for IR
  - Boolean model:
    - Used in specialized area
    - Not appropriate for general search alone – often used as a pre-filtering
  - Vector space model:
    - Robust
    - Good experimental results
  - Probabilistic models:
    - Difficulty to estimate probabilities accurately
    - Modified version (BM25) – excellent results
    - Regression models:
      - Need training data
      - Widely used (in a different form) in web search
      - Learning to rank (a later lecture)
- More recent model on statistical language modeling (robust model relying on a large amount of data – next lecture)