RETRIEVAL MODELS

Outline

- Intro
- Basics of probability and information theory
- Retrieval models
  - Boolean model
  - Vector space model
  - Probabilistic retrieval
- Retrieval evaluation
- Link analysis
- From queries to top-k results
- Social search

Boolean model

- **Exact match,** i.e., either a document "matches" the query or not
- **No ranking**
- Query composed of **Boolean operators**

  Examples: "Richard ∧ Nixon ∧ Watergate",

  "Schwarzenegger ∧ (governor ∨ politics) ∧ ¬(movie ∨ action)"

- Advantages
  - **Efficient processing**
  - Results are **easy to explain**
- Main disadvantages
  - Effectiveness (i.e., whether user will find all relevant results) depends entirely on user
  - Not practical for web retrieval (result set would be either huge or too small)

Extended Boolean model (1)

- Consider term weights, e.g., for document d and term t

$$w(t,d) = P(t|d) \propto tf(t,d)$$

- Consider the vector $(t_1, \dots, t_m)$ of all terms in all documents (in some order)
- d can be represented as $d = (w(t_1,d), \dots, w(t_m,d))$

- For query $q = t_{i_1} \wedge \cdots \wedge t_{i_k}$

$$sim(d,q) = 1 - \sqrt[p]{\frac{\left(1 - w(t_{i_1},d)\right)^p + \cdots + \left(1 - w(t_{i_k},d)\right)^p}{k}}$$

- For query $q = t_{i_1} \vee \cdots \vee t_{i_k}$

$$sim(d,q) = \sqrt[p]{\frac{w(t_{i_1},d)^p + \cdots + w(t_{i_k},d)^p}{k}}$$

P-norm

- Rank documents by decreasing similarities to query

Extended Boolean model (2)

(a) Conventional Boolean Retrieval

	Terms		Similarity with Query	
	A	*B*	(*A* or *B*)	(*A* and *B*)
Document D_1	1	1	1	1
Document D_2	1	0	1	0
Document D_3	0	1	1	0
Document D_4	0	0	0	0

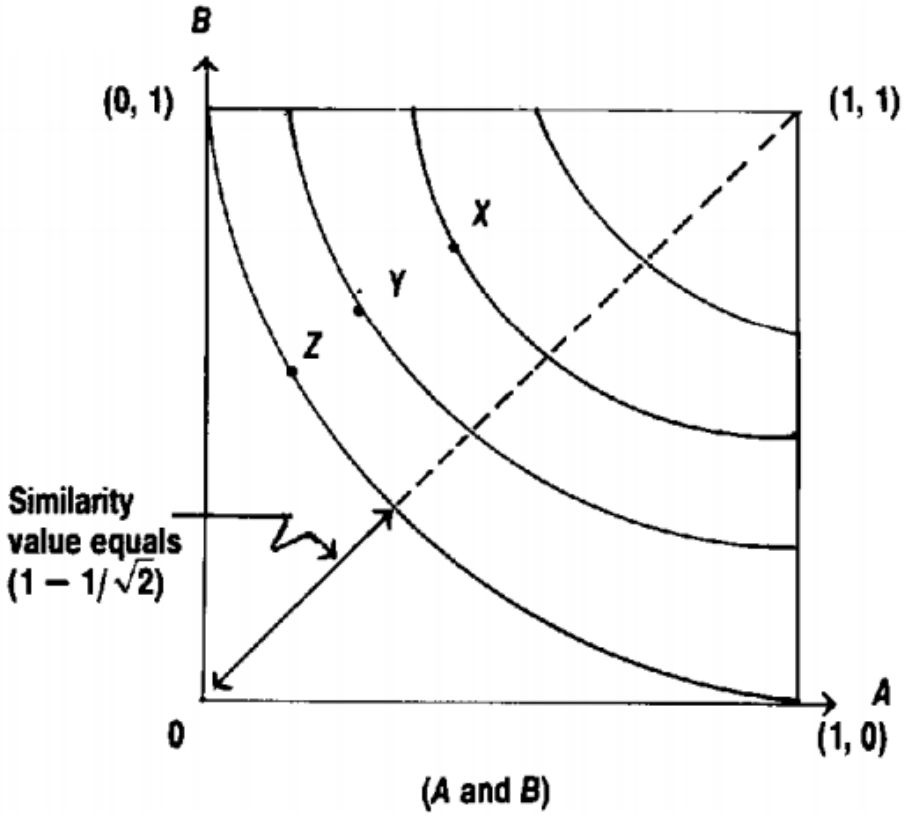
(b) Extended Retrieval

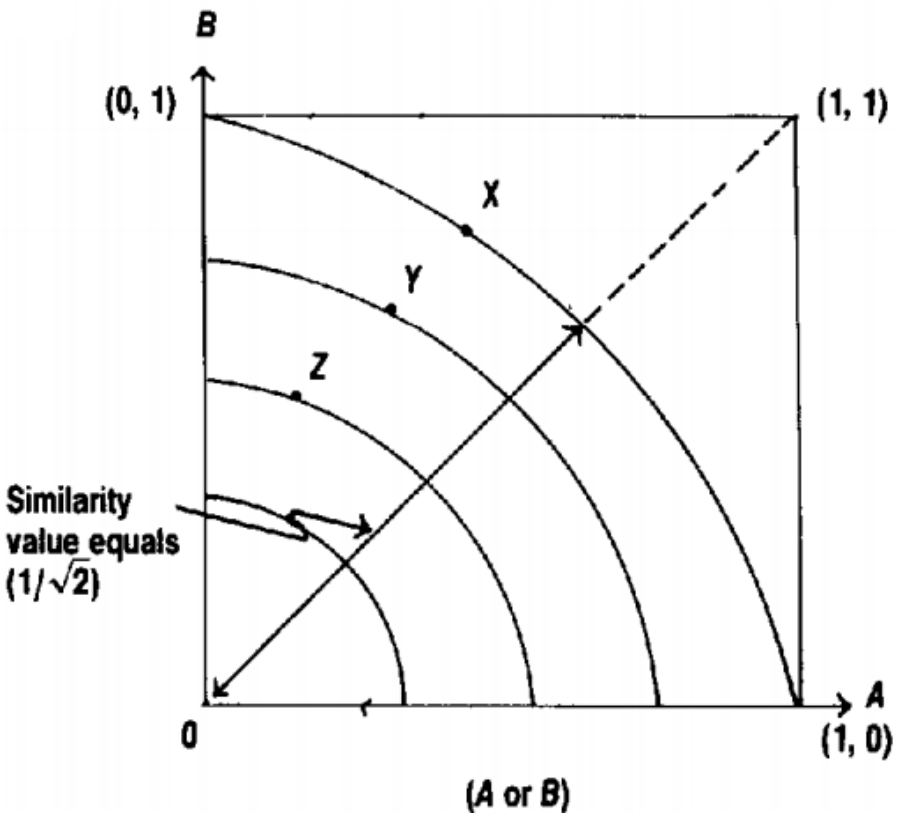
Document D_1	1	1	1	1
Document D_2	1	0	$1/\sqrt{2}$	$1-1/\sqrt{2}$
Document D_3	0	1	$1/\sqrt{2}$	$1-1/\sqrt{2}$
Document D_4	0	0	0	0

Source: G. Salton, *Extended Boolean Information Retrieval*. CACM 1983

Extended Boolean model: example

- Documents *X, Z, Y* and queries $A \wedge B$ and $A \vee B$





Source: G. Salton, *Extended Boolean Information Retrieval*. CACM 1983

Vector space model (1)

- Term-document matrix

	d_1	d_2	d_3	d_4	d_5	d_6
champion	3	2	0	0	0	0
football	2	0	0	0	0	0
goal	4	3	0	1	0	0
law	0	0	2	3	0	0
party	0	0	6	5	0	0
politician	0	0	4	4	0	0
rain	0	0	0	0	3	3
score	4	5	0	0	0	0
soccer	0	3	0	0	0	0
weather	0	0	0	0	5	4
wind	0	1	0	0	2	3

Vector space model (2)

- Main idea
  - Terms span a vector space
  - Documents and queries are vectors in that space

- Distance measured by **cosine similarity:**

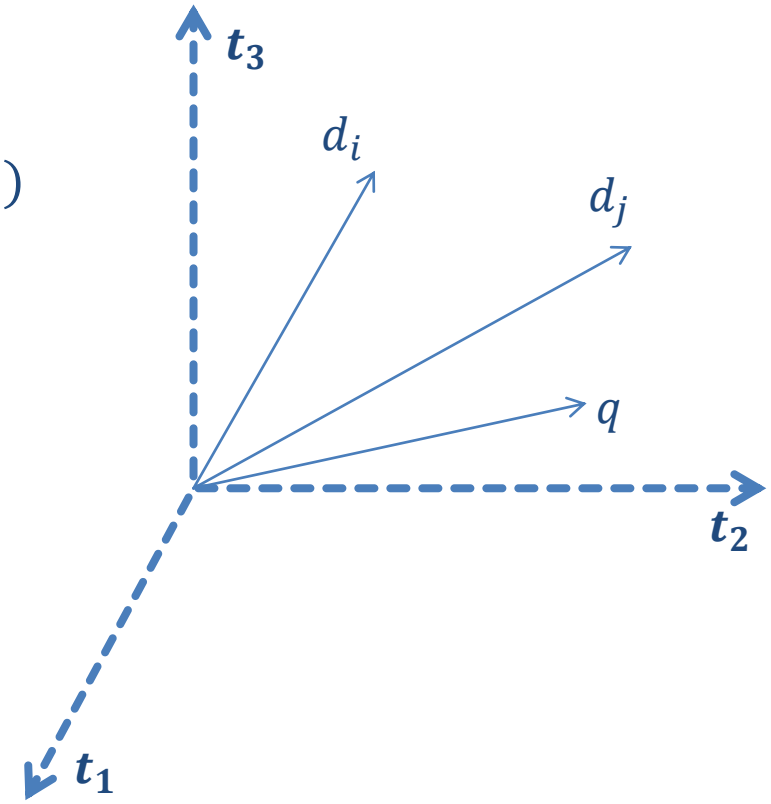
For $d_i = (t_{i1}, \ldots, t_{im})$ and $q = (q_1, \ldots, q_m)$

$$sim(d_i, q) = \frac{\sum_j t_{ij} q_j}{\sqrt{\sum_j {t_{ij}}^2 \sum_j {q_j}^2}}$$

Measures angle-based affinity between query and doc vector

Feature value of doc term (e.g., tf-idf)

Feature value of query term



Vector space model: example

- Query: “football score”

- Query vector: $q = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$

	d_1	d_2	d_3	d_4	d_5	d_6
champion	3	2	0	0	0	0
football	2	0	0	0	0	0
goal	4	3	0	1	0	0
law	0	0	2	3	0	0
party	0	0	6	5	0	0
politician	0	0	4	4	0	0
rain	0	0	0	0	3	3
score	4	5	0	0	0	0
soccer	0	3	0	0	0	0
weather	0	0	0	0	5	4
wind	0	1	0	0	2	3

- Ranking by using term frequencies
  - $tf(football, d_1) \approx 0.15, \quad tf(score, d_1) \approx 0.31, \quad tf(football, d_2) \approx 0, \quad tf(score, d_2) \approx 0.36$
  - $sim(d_1, q) = \frac{{d_1}^T q}{|d_1||q|} \approx \frac{0.15 \cdot 1 + 0.31 \cdot 1}{0.52 \cdot 1.41} = 0.63, \quad sim(d_2, q) \approx \frac{0 \cdot 1 + 0.36}{0.49 \cdot 1.41} = 0.52$

Relevance feedback

- User issues simple query
- System returns initial results
- User marks some results as **relevant** others as **non-relevant**
- Initial query can be reformulated to yield better results based on above relevance feedback

$$q_{opt} = \arg max_q \{sim(q, R) - sim(q, \bar{R})\}$$

Relevant docs

Non-relevant docs

- In vector space model

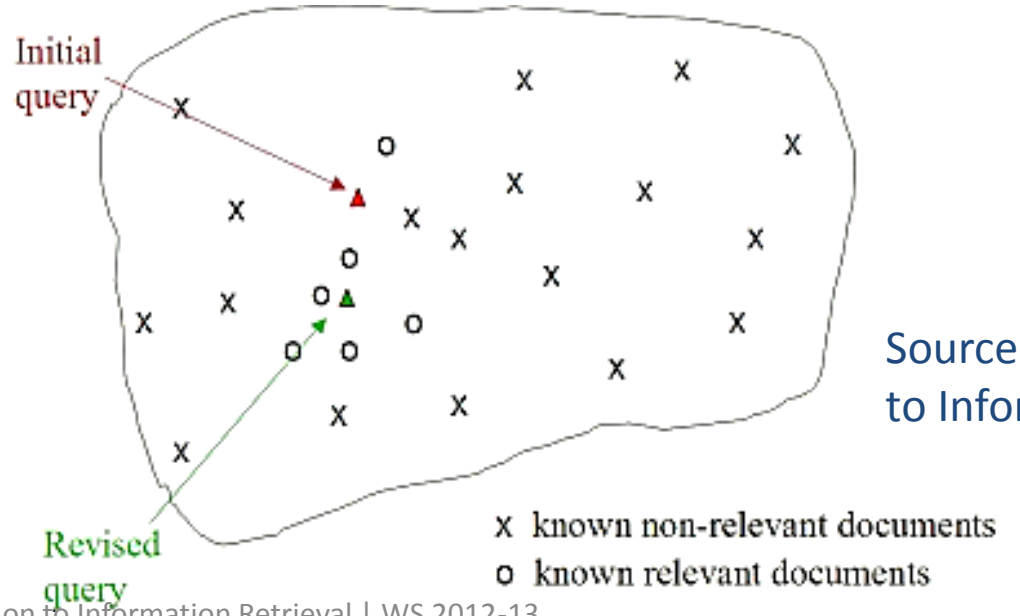
$$q_{opt} = \arg max_q \left\{ \frac{1}{|R|} \sum_{d_j \in R} sim(d_j, q) - \frac{1}{|\bar{R}|} \sum_{d_j \in \bar{R}} sim(d_j, q) \right\}$$

Relevance feedback: Rocchio algorithm

- Used as **relevance feedback** algorithm in the SMART system

$$q_* = \alpha q_0 + \beta \frac{1}{|R|} \sum_{d_j \in R} d_j - \gamma \frac{1}{|\bar{R}|} \sum_{d_j \in \bar{R}} d_j$$

- With typical parameter values $\alpha = 8, \beta = 16, \gamma = 4$
- Negative feature values are set to 0



Source: Introduction to Information Retrieval

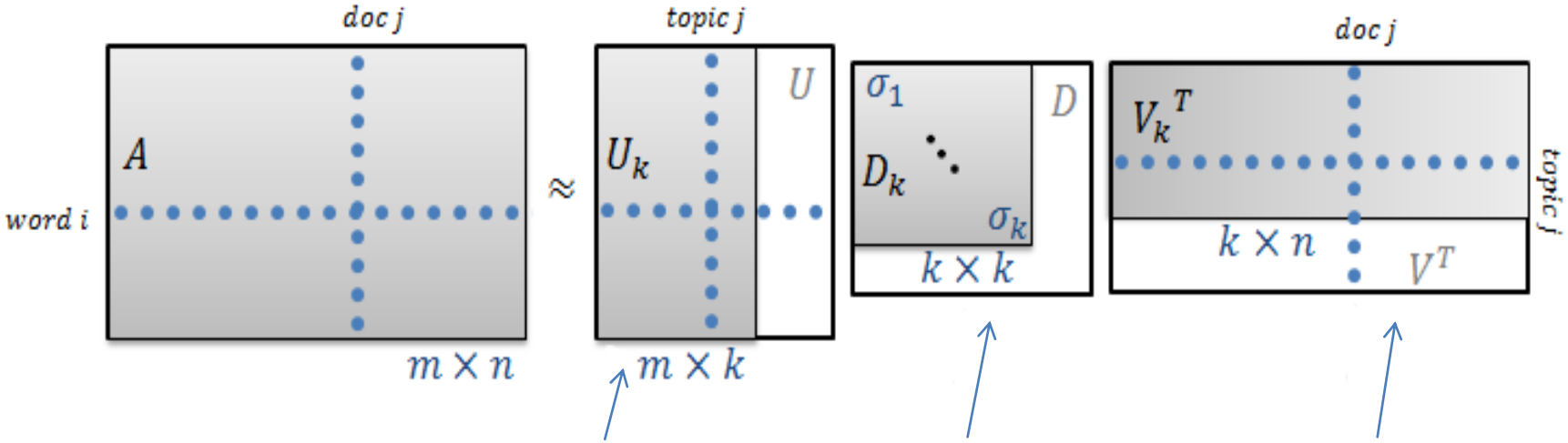
Vector space model for Latent Semantic Indexing (1)

- Map documents to corresponding latent topics

	d_1	d_2	d_3	d_4	d_5	d_6
champion	3	2	0	0	0	0
football	2	0	0	0	0	0
goal	4	3	0	1	0	0
law	0	0	2	3	0	0
party	0	0	6	5	0	0
politician	0	0	4	4	0	0
rain	0	0	0	0	3	3
score	4	5	0	0	0	0
soccer	0	3	0	0	0	0
weather	0	0	0	0	5	4
wind	0	1	0	0	2	3

Vector space model for Latent Semantic Indexing (2)

- Basic idea (further details in the **Data Mining** lecture…)



holds orthonormal Eigenvectors of AA^T (doc-doc-similarity)

positive roots of the Eigenvalues (singular values) of $A^T A$

holds orthonormal Eigenvectors of $A^T A$ (term-term similarity)

Mapping of docs $\vec{d}_i, \vec{d}_j$ into latent space:

$$\vec{d}_i \mapsto {U_k}^T \vec{d}_i = \vec{d}_i' = \left(\left(D_k {V_k}^T \right)_{(i)} \right)$$

$$\vec{q} \mapsto {U_k}^T \vec{q} = \vec{q}' \text{ (map keyword query)}$$

… and measure cosine-similarity

Mapping of new doc d' into latent space:

$$\vec{d} \mapsto {U_k}^T \vec{d} = \vec{d}'$$

Add $\vec{d}'$ as last column of ${V_k}^T$

Vector space model: pros & cons

- Advantages
  - Simple framework for ranking
  - Other similarity measures or similarity weighting schemes can be used, e.g.:

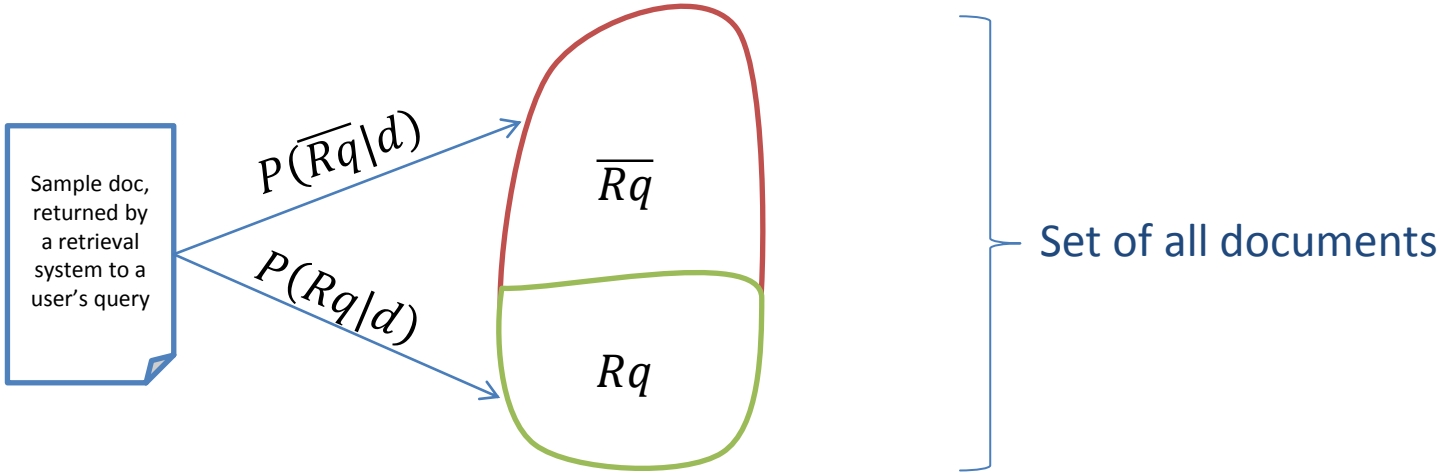
Metric distance	Definition
Euclidean	$\|\mathbf{x}-\mathbf{y}\| = \sqrt{\sum_i (x_i - y_i)^2}$
Manhattan	$\|\mathbf{x}-\mathbf{y}\|_1 = \sum_i \|x_i - y_i\|$
Maximum	$\|\mathbf{x}-\mathbf{y}\|_\infty = \max_i \|x_i - y_i\|$

with $sim(\mathbf{x},\mathbf{y}) := \frac{1}{1+dist(\mathbf{x},\mathbf{y})}$ or $sim(\mathbf{x},\mathbf{y}) := \frac{1}{e^{dist(\mathbf{x},\mathbf{y})}}$

- Disadvantages
  - Term independence assumption
  - Impractical for relevance prediction

Probabilistic retrieval

- Basic idea
  - **Return documents to a query ranked by decreasing probability of relevance** (with respect to the query)
  - If probabilities are calculated on all available data, the retrieval effectiveness should be best!



- **Bayes decision rule:** document d is relevant if

$$P(Rq|d) > P(\overline{Rq}|d) \Leftrightarrow P(Rq,d) > P(\overline{Rq},d)$$

Binary independence model with Naïve Bayes

- Estimate probabilities

$$P(Rq, d) = P(d|Rq)P(Rq)$$

- Assume independence between terms in the document given the class (e.g., Rq or $\overline{Rq}$)

$$P(d|Rq) = \prod_{t_i \in d} P(t_i|Rq)$$

- Document d can be classified as relevant if

$$\frac{P(d|Rq)}{P(d|\overline{Rq})} > \frac{P(\overline{Rq})}{P(Rq)}$$

- Note: a document is a vector of binary features (i.e., 0, 1 indicating term occurrence or non-occurrence respectively)

Bayes decision with binary independence model

- Let $r_i := P(t_i|Rq)$ and $\overline{r_i} := P(t_i|\overline{Rq})$

$$\frac{P(d|Rq)}{P(d|\overline{Rq})} = \prod_{i,t_i=1} \frac{r_i}{\overline{r_i}} \cdot \prod_{i,t_i=0} \frac{1-r_i}{1-\overline{r_i}}$$

$$= \prod_{i,t_i=1} \frac{r_i}{\overline{r_i}} \cdot \left(\prod_{i,t_i=1} \frac{1-\overline{r_i}}{1-r_i} \cdot \prod_{i,t_i=1} \frac{1-r_i}{1-\overline{r_i}} \right) \prod_{i,t_i=0} \frac{1-r_i}{1-\overline{r_i}}$$

$$= \prod_{i,t_i=1} \frac{r_i(1-\overline{r_i})}{\overline{r_i}(1-r_i)} \cdot \prod_{i} \frac{1-r_i}{1-\overline{r_i}}$$

- In practice, score is computed by taking the logarithm

$$\frac{P(d|Rq)}{P(d|\overline{Rq})} \approx \sum_{i,t_i=1} \log \frac{r_i(1-\overline{r_i})}{\overline{r_i}(1-r_i)}$$

BM25

- **Contingency table**

	Relevant	Non-relevant	All
$t_i = 1$	R_i	$n_i - R_i$	n_i
$t_i = 0$	$R - R_i$	$N - R - (n_i - R_i)$	$N - n_i$
All	R	$N - R$	N

- **Smoothed maximum likelihood estimations** for r_i and $\overline{r_i}$

$$r_i = \frac{R_i + 0.5}{R + 1} \qquad \overline{r_i} = \frac{n_i - R_i + 0.5}{N - R + 1}$$

- **BM25** (short for Best Match 25)

$$\frac{P(d|Rq)}{P(d|\overline{Rq})} \approx \sum_{i,t_i=1} \log \frac{r_i(1-\overline{r_i})}{\overline{r_i}(1-r_i)} = \sum_{i,t_i=1} \log \frac{(R_i + 0.5)(N - R - n_i + R_i + 0.5)}{(n_i - R_i + 0.5)(R - R_i + 0.5)}$$

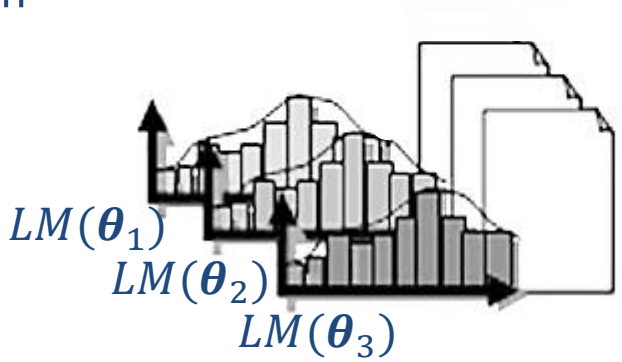
$$BM25(d,q) \approx \sum_{i,t_i=1} \log \frac{(R_i + 0.5)(N - R - n_i + R_i + 0.5)}{(n_i - R_i + 0.5)(R - R_i + 0.5)} \cdot \frac{(\alpha + 1)f_i}{A + f_i} \cdot \frac{(\beta + 1)qf_i}{\beta + qf_i}$$

α, β, A are parameters, where $A = \alpha\left((1-\gamma) + \gamma \frac{|d|}{av(|d_k|)}\right)$

Language models

- Text is generated from probability distribution over terms (i.e., **language model**)

- Terms are drawn with probability of their occurrence



- Suppose **each document** $d = (t_1, \dots, t_m)$ **is represented by a language model**, e.g., multinomial distribution:

$$P(t_1 = k_1, \dots, t_m = k_m; p_1, \dots, p_m) = \frac{|d|!}{k_1! \dots k_m!} {p_1}^{k_1} \dots {p_m}^{k_m}$$

with $\boldsymbol{\theta}_d = (p_1, \dots, p_m)$, such that $\sum_i p_i = 1$ and $\sum_i k_i = |d|$

- Given a query $q = (q_1, \dots, q_n)$, what is the likelihood $P(q|\boldsymbol{\theta}_d)$ that q is a sample from a document d?

- Rank documents by decreasing likelihoods of having generated q.

Language models: the query likelihood model

- Start with joint probability distribution

$$P(\boldsymbol{\theta}_d, q) = P(q|\boldsymbol{\theta}_d)P(\boldsymbol{\theta}_d)$$

likelihood prior

- Assume uniform prior and conditional independence between query terms given the parameters $\boldsymbol{\theta}_d$

$$P(q|\boldsymbol{\theta}_d) = \prod_i P(q_i|\boldsymbol{\theta}_d) \quad \text{(unigram model)}$$

- Rank documents by decreasing likelihoods $P\left(q|\boldsymbol{\theta}_{d_j}\right)$

Language models: estimating probabilities

$$P(q|\boldsymbol{\theta}_d) = \prod_i P(q_i|\boldsymbol{\theta}_d)$$

- Estimate $P(q_i|\boldsymbol{\theta}_d)$ by

$$P(q_i|\boldsymbol{\theta}_d) = \frac{freq(q_i, d)}{|d|}$$

Maximum likelihood estimation

- Problem
  - What if q_i does not occur in d?

Language models: smoothing

$$P(q_i|\boldsymbol{\theta}_d) = \frac{freq(q_i, d)}{|d|}$$

Maximum likelihood estimation

- Problem
  - What if q_i does not occur in d?
- Use smoothing: technique for estimating probabilities of missing words
- Mixture model for smoothing

$$P(q|\boldsymbol{\theta}_d) = \prod_i \big((1-\alpha)P(q_i|\boldsymbol{\theta}_d) + \alpha P(q_i|C)\big)$$

C: whole document corpus

Background model

Smoothing & tf-idf

- Mixture model for smoothing

$$P(q|\boldsymbol{\theta}_d) = \prod_i \big((1-\alpha)P(q_i|\boldsymbol{\theta}_d) + \alpha P(q_i|C)\big)$$

$$\simeq \sum_i \log\big((1-\alpha)P(q_i|\boldsymbol{\theta}_d) + \alpha P(q_i|C)\big)$$

$$= \sum_{i,f\,q_i>0} \log\big((1-\alpha)P(q_i|\boldsymbol{\theta}_d) + \alpha P(q_i|C)\big) + \sum_{i,f\,q_i=0} \log\big(\alpha P(q_i|C)\big)$$

$$= \sum_{i,f\,q_i>0} \log\left(\frac{(1-\alpha)P(q_i|\boldsymbol{\theta}_d) + \alpha P(q_i|C)}{\alpha P(q_i|C)}\right) + \sum_i \log\big(\alpha P(q_i|C)\big)$$

$$\approx \sum_{i,f\,q_i>0} \log\left(\frac{(1-\alpha)P(q_i|\boldsymbol{\theta}_d)}{\alpha P(q_i|C)} + 1\right)$$

What does this have to do with tf-idf ???

Jelineck-Mercer smoothing

$$P(q|\boldsymbol{\theta}_d) = \prod_i \big((1-\alpha)P(q_i|\boldsymbol{\theta}_d) + \alpha P(q_i|C)\big)$$

- Set α to a constant λ and use maximum likelihood estimations

$$P(q|\boldsymbol{\theta}_d) = \prod_i \left((1-\lambda)\frac{freq(q_i,d)}{|d|} + \lambda\frac{freq(q_i,C)}{|C|}\right)$$

- In practice: use logarithms for ranking

$$Score(q,d) = \sum_i \log\left((1-\lambda)\frac{freq(q_i,d)}{|d|} + \lambda\frac{freq(q_i,C)}{|C|}\right)$$

where $|C| = \sum_{t\in C} freq(t,d)$

Dirichlet smoothing

$$P(q|\boldsymbol{\theta}_d) = \prod_i \left((1-\alpha)P(q_i|\boldsymbol{\theta}_d) + \alpha P(q_i|C)\right)$$

- Basic idea
  - Term that does not occur in a long document should be assigned low smoothed probability (the longer the document, the lower this probability)

- Set $\alpha = \frac{\mu}{\mu+|d|}$ and use maximum likelihood estimations

$$P(q|\boldsymbol{\theta}_d) = \prod_i \left(\frac{freq(q_i,d) + \mu \cdot freq(q_i,C)/|C|}{\mu + |d|}\right)$$

$$Score(d,q) = \sum_i \log\left(\frac{freq(q_i,d) + \mu \cdot freq(q_i,C)/|C|}{\mu + |d|}\right)$$

Dirichlet smoothing: side note

- Motivated by Bayesian reasoning

$$P(\boldsymbol{\theta}|D) \propto P(D|\boldsymbol{\theta})P(\boldsymbol{\theta})$$

posterior — likelihood — prior

- Dirichlet distribution is **conjugate prior** of the Multinomial distribution (enables closed-form calculation: prior has same functional form as posterior)

- In Dirichlet smoothing: Dirichlet prior is assumed over the model parameters, likelihood is multinomially distributed

- Priors are updated as more and more data is observed

- Further details in the **Data Mining** lecture …

Laplace/Lidstone smoothing

- Estimate $P(q_i|\boldsymbol{\theta}_d)$ by

$$P(q_i|\boldsymbol{\theta}_d) = \frac{freq(q_i,d)+1}{\sum_{t\in C}(freq(t,d)+1)} = \frac{freq(q_i,d)+1}{|C|+|d|}$$

- For ranking

$$Score(q,d) = \sum_i \log\left(\frac{freq(q_i,d)+1}{|C|+|d|}\right)$$

- **Advantage:** easy-to-use (i.e., prameter-less) additive smoothing technique, no background model
- **Disadvantage:** for corpus with many short documents probability estimations are biased towards corpus frequencies

Smoothing: examples

- Jelineck-Mercer smoothing ($\lambda = 0.2$)

$$P(party|d_1) = 0.8\frac{0}{13} + 0.2\frac{11}{72} \approx 0.031$$

$$P(wind|d_2) = 0.8\frac{1}{14} + 0.2\frac{6}{72} \approx 0.074$$

- Dirichlet smoothing ($\mu = 0.2$)

$$P(party|d_1) = \frac{0+0.2\frac{11}{72}}{0.2+13} \approx 0.0023$$

$$P(wind|d_2) = \frac{1+0.2\frac{6}{72}}{0.2+14} \approx 0.072$$

- Laplace/Lidstone smoothing

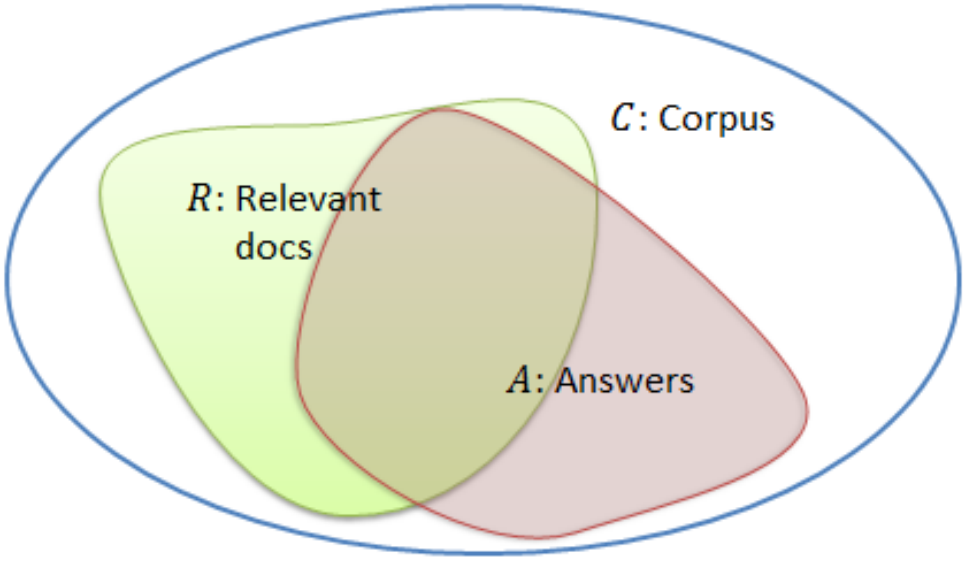
$$P(party|d_1) = \frac{0+1}{72+13} \approx 0.01$$

$$P(wind|d_2) = \frac{1+1}{72+14} \approx 0.023$$

	d_1	d_2	d_3	d_4	d_5	d_6
champion	3	2	0	0	0	0
football	2	0	0	0	0	0
goal	4	3	0	1	0	0
law	0	0	2	3	0	0
party	0	0	6	5	0	0
politician	0	0	4	4	0	0
rain	0	0	0	0	3	3
score	4	5	0	0	0	0
soccer	0	3	0	0	0	0
weather	0	0	0	0	5	4
wind	0	1	0	0	2	3

Language models vs. tf-idf

Precision-recall results on TREC dataset.
Source: *A Language Modeling Approach to Information Retrieval* by M. Ponte



$$Precision = \frac{|R \cap A|}{|A|} \qquad Recall = \frac{|R \cap A|}{|R|}$$

More on evaluation measures, next lecture…

	tf.idf	LModel
Relevant:	10485	10485
Rel. ret.:	5818	6105
Precision		
at 0.00	0.7274	0.7805
at 0.10	0.4861	0.5002
at 0.20	0.3898	0.4088
at 0.30	0.3352	0.3626
at 0.40	0.2826	0.3064
at 0.50	0.2163	0.2512
at 0.60	0.1561	0.1798
at 0.70	0.0913	0.1109
at 0.80	0.0510	0.0529
at 0.90	0.0179	0.0152
at 1.00	0.0005	0.0004
Avg:	0.2286	0.2486
Precision at:		
5 docs:	0.5320	0.5960
10 docs:	0.5080	0.5260
15 docs:	0.4933	0.5053
20 docs:	0.4670	0.4890
30 docs:	0.4293	0.4593
100 docs:	0.3344	0.3562
200 docs:	0.2670	0.2852
500 docs:	0.1797	0.1881
1000 docs:	0.1164	0.1221
R-Precision:	0.2836	0.3013

Smoothing: summary

- Jelineck-Mercer smoothing
  - Flexible combination of term frequencies with background model
  - Empirically: the longer the queries the larger λ (i.e., the more smoothing is needed)
  - Does not take document length into account
- Dirichlet smoothing
  - Probabilistic (i.e. Bayesian) reasoning about prior distribution on model parameters
  - Empirically: the longer the queries the larger μ
  - However, μ is collection dependent
- Laplace/Lidstone smoothing
  - Easy-to-use additive smoothing technique
  - Probability estimations may be biased towards corpus frequencies

- Two-stage smoothing is possible (i.e., Jelineck-Mercer smoothing on a Dirichlet smoothed estimation)

Language models for relevance (1)

- In **document language models** we were interested in estimating the query likelihood $P(q|\boldsymbol{\theta}_d)$
- In **relevance language models** we are interested in $P(d|\boldsymbol{\theta}_{Rq})$, the probability of a document being generated by a relevance model (i.e., document is sample from this model)
- Can the two models be combined for ranking?
- One possibility: rank documents by increasing dissimilarity between $P(q|\boldsymbol{\theta}_d)$ and $P(d|\boldsymbol{\theta}_{Rq})$

Language models for relevance (2)

- Measure the negative Kullback-Leibler divergence

$$-KL\left(P\left(d|\boldsymbol{\theta}_{Rq}\right)\|P(q|\boldsymbol{\theta}_d)\right) = -\sum_t P\left(t|\boldsymbol{\theta}_{Rq}\right)\log\frac{P\left(t|\boldsymbol{\theta}_{Rq}\right)}{P(t|\boldsymbol{\theta}_d)}$$

$$= \sum_t P\left(t|\boldsymbol{\theta}_{Rq}\right)\log P(t|\boldsymbol{\theta}_d) - \sum_t P\left(t|\boldsymbol{\theta}_{Rq}\right)\log P\left(t|\boldsymbol{\theta}_{Rq}\right)$$

- Simple estimation of $P\left(t|\boldsymbol{\theta}_{Rq}\right)$:

$$P\left(t|\boldsymbol{\theta}_{Rq}\right) \approx P(t|q_1,\dots,q_n) = \frac{P(t,q_1,\dots,q_n)}{P(q_1,\dots,q_n)}$$

$$P(t,q_1,\dots,q_n) = \sum_{d\in C} P(d)P(t,q_1,\dots,q_n|d)$$

$$= \sum_{d\in C} P(d)P(t\,|d)\prod_i P(q_i|d)$$

conditional independence assumption

Pseudo-relevance feedback algorithm

- Estimation of $P(t|\boldsymbol{\theta}_{Rq})$:

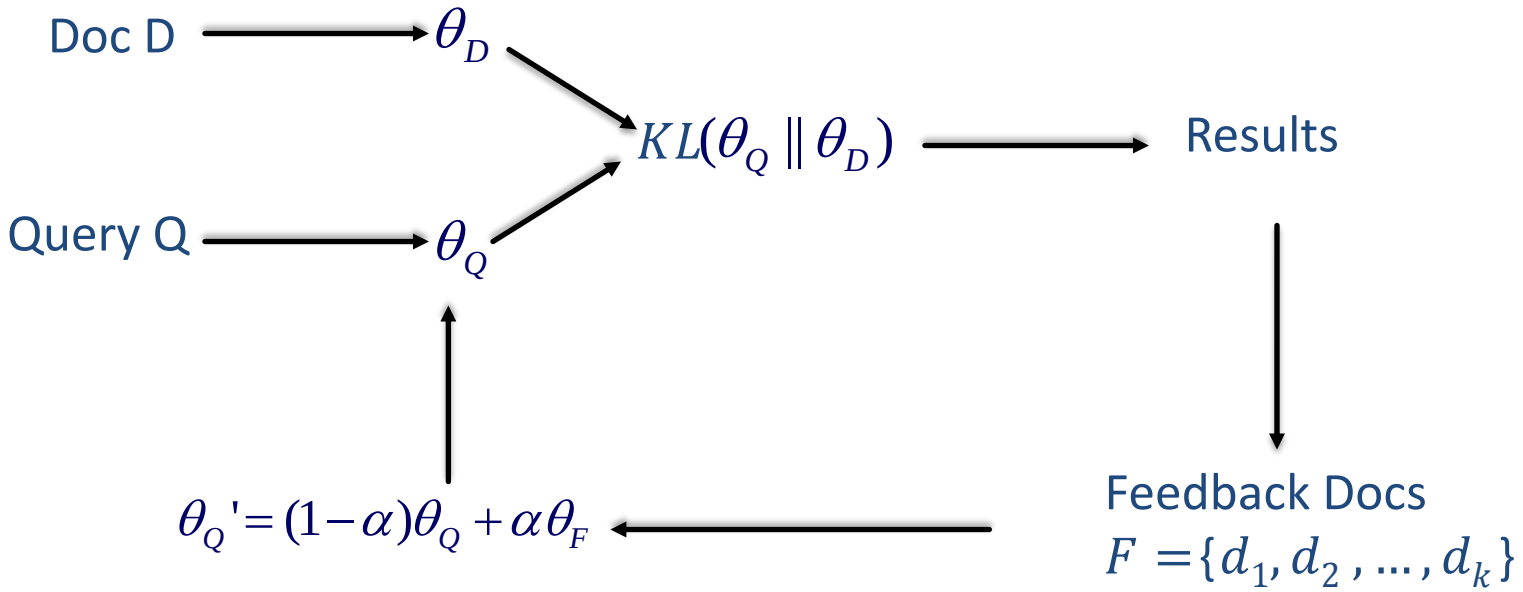
$$P(t|\boldsymbol{\theta}_{Rq}) \approx P(t|q_1,\dots,q_n) = \frac{P(t,q_1,\dots,q_n)}{P(q_1,\dots,q_n)}$$

$$P(t,q_1,\dots,q_n) = \sum_{d\in C} P(d)P(t,q_1,\dots,q_n|d)$$

$$= \sum_{d\in C} P(d)P(t|d)\prod_i P(q_i|d)$$

1. Rank documents by $P(q|\boldsymbol{\theta}_d)$
2. Confine the corpus to C = {top-k ranked documents}
3. Estimate $P(d|\boldsymbol{\theta}_{Rq})$ on this new corpus
4. Rerank documents by using $-KL(P(d|\boldsymbol{\theta}_{Rq})||P(q|\boldsymbol{\theta}_d))$

Integrating relevance feedback in language models

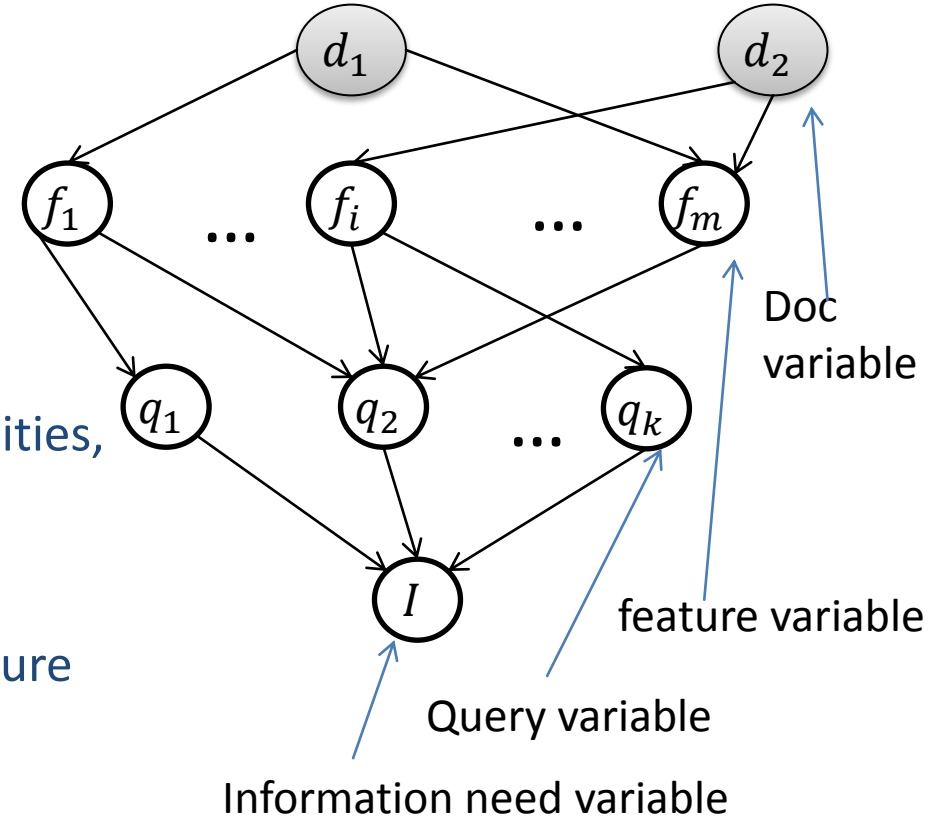


Language models: summary

- Tractable retrieval framework with appealing interpretation
- Smoothed maximum likelihood estimations for unigrams work well in practice
- Estimations work less well for multigram likelihoods, problem known as **curse of dimensionality**
- Relevance model is possible
- Relevance feedback is difficult to integrate

Advanced probabilistic models: Bayesian networks

- Bayesian network: directed graph representing **dependencies between random variables**
- **Local Markov property:** a variable node is independent of all its non-descendants given its parents.
- One can reason about joint probabilities, e.g., $P(d_i, I)$
- More details in the Data Mining lecture

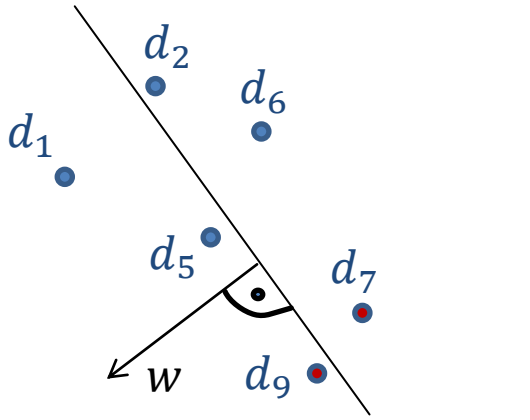


Ranking support vector machine (Ranking SVM)

- Motivation: feedback is cheap and plentiful
- Input: $(q, r), (q', r'), \ldots$, where $(d_i, d_j) \in r$ if d_i ranks higher than d_j
- Can we learn a weight vector w such that

  $\forall\, r$:

  $$\forall (d_i, d_j) \in r\colon\ w^T d_i > w^T d_j$$



$(d_1, d_2), (d_1, d_6), (d_5, d_6) \in r$
$(d_5, d_2), (d_5, d_7), (d_7, d_9) \in r'$

- Optimization problem

  $$minimize \quad \frac{1}{2}\|w\| + \lambda \sum_{i,j,r} \xi_{i,j,r}$$

  $$subject\ to$$

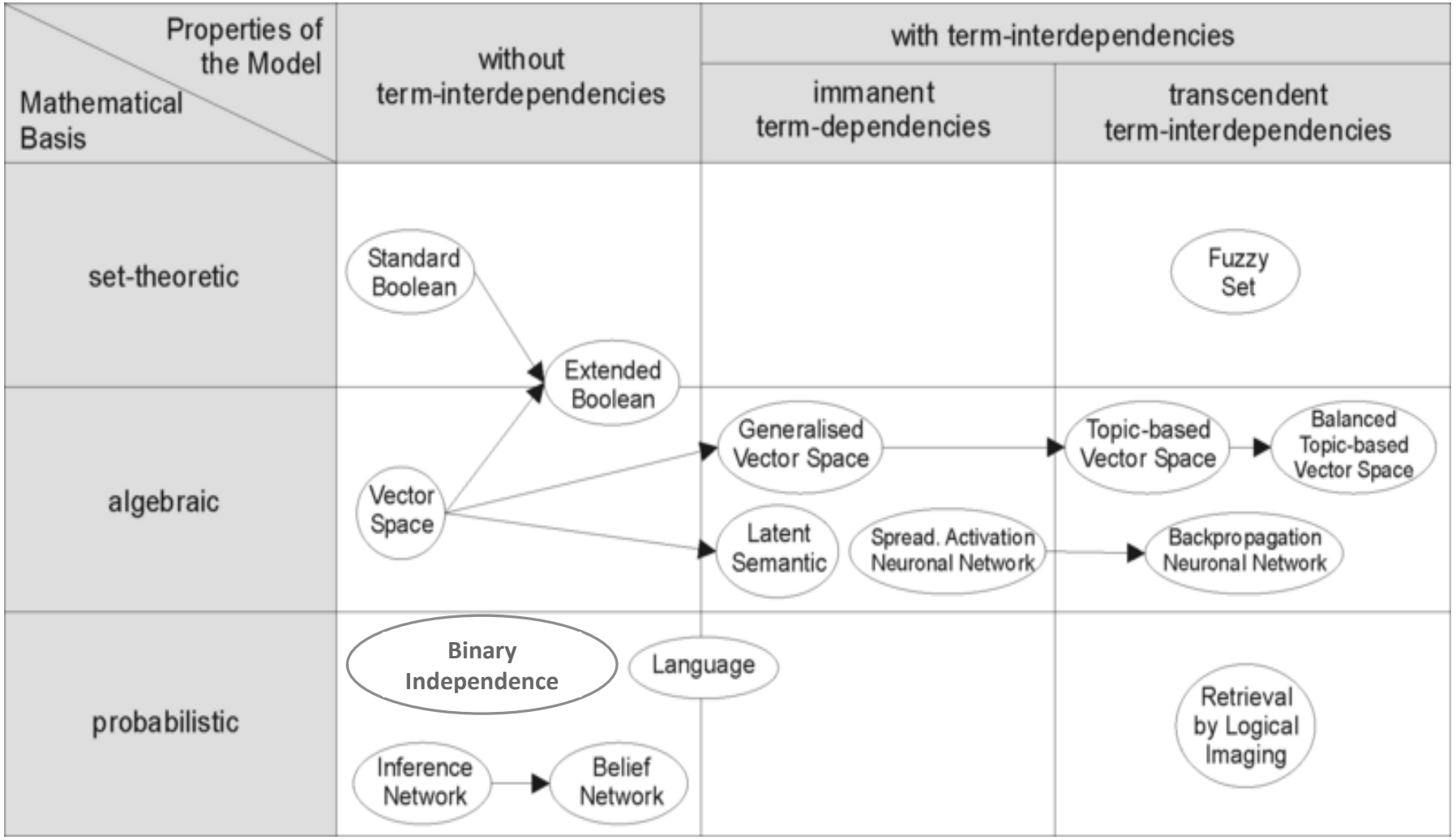
  $\forall\, r$:

  $$\forall (d_i, d_j) \in r\colon\ w^T d_i > w^T d_j + 1 - \xi_{i,j,r}$$

Slack variable, allows some misclassification

- More details on SVMs in the Data Mining lecture

Retrieval models (overview)



Source: http://en.wikipedia.org/wiki/File:Information-Retrieval-Models.png

Summary

- Boolean model
  - Concatenation of query keywords by Boolean operators
  - Either match or no match, no ranking
  - Allows efficient processing but is impractical for large-scale retrieval
- Vector space model
  - Vector representations of query and documents allow algebraic methods for measuring similarity
  - Good for ranking, but not good dealing with uncertainty or predicting relevance
- Probabilistic and language models
  - Extensible and versatile
  - Can handle uncertainty (by reasoning about probabilities)
  - Maximum likelihood estimations work well for unigram models, less well for multigram models
  - Difficult to maintain at large scale and in the presence of frequent updates