Chapter 4

# Modeling

Alternative Probabilistic Models
Non-overlapping Lists
Proximal Nodes

# Alternative Probabilistic Models

# Alternative Probabilistic Models

- Three alternative probabilistic models were selected to discuss
  - **BM25**, which is a direct extension of the classic probabilistic model
  - **Belief Network Models**, which are a direct application of bayesian networks to IR
  - **Language Models**, which presents a variant of the idea of using the distribution of index terms in the collection as the basis for ranking

# Alternative Probabilistic Models

## BM25 (Best Match 25)

# BM25 (Best Match 25)

- BM25 was originated as part of the participation of the Okapi system in the TREC conferences
- A good term weighting is based on three principles
  - inverse document frequency
  - term frequency
  - document length normalization
- The classic probabilistic model covers only the first of these principles
- This reasoning led to a series of experiments on variations of the classic probabilistic model
- Such experimentation culminated on the BM25 ranking formula

# BM1, BM11 and BM15 Formulas

- At first, the Okapi system used the Equation below as ranking formula

$$sim(d_j, q) \sim \sum_{k_i \in q \wedge k_i \in d_j} \log \frac{N - n_i + 0.5}{n_i + 0.5}$$

  which is the equation used in probabilistic model when no relevance information is provided

- It was referred to as the BM1 formula (*Best Match 1*)

# BM1, BM11 and BM15 Formulas

- The next idea was to introduce a term-frequency in BM1 ranking formula
- The used factor was based on a 2-poisson model of term occurence within the documents:

$$S_1 \times \frac{f_{i,j}}{K_1 + f_{i,j}}$$

  - where
    - $f_{i,j}$ is the frequency of term $k_i$ within document $d_j$
    - $K_1$ is a constant setup experimentally for each collection
    - $S_1$ is a scaling constant, normally set to $S_1 = (K_1 + 1)$
- If $K_1 = 0$ (and $S_1 = (K_1 + 1)$), this whole factor becomes equal to 1 and bears no effect in the ranking

# BM1, BM11 and BM15 Formulas

- The next step was to introduce document length normalization into the formulation
- This can be attained by changing previous equation to

$$S_1 \times \frac{f_{i,j}}{\frac{K_1 \times len(d_j)}{avg\_doclen} + f_{i,j}}$$

  - where
    - $len(d_j)$ is the length of document $d_j$ (computed, for instance, as the number of terms in the document)
    - $avg\_doclen$ is the average document length for the collection

# BM1, BM11 and BM15 Formulas

- It was also used a correction factor dependent on document length and on the query length:

$$K_2 \times len(q) \times \frac{avg\_doclen - len(d_j)}{avg\_doclen + len(d_j)}$$

  - where
    - $len(q)$ is the query length (number of terms in the query)
    - $K_2$ is a constant
- This factor depends only on the document and query lengths, and was referred as $G_2$

# BM1, BM11 and BM15 Formulas

- An additional factor was applied to term frequencies within queries

$$S_3 \times \frac{f_{i,q}}{K_3 + f_{i,q}}$$

  - where
    - $f_{i,q}$ is the frequency of term $k_i$ within query $q$
    - $K_3$ is a constant
    - $S_3$ is an scaling constant related to $K_3$, normally set to $S_3 = (K_3 + 1)$

# BM1, BM11 and BM15 Formulas

- Introduction of these three factors into Equation of BM1 leads to two BM formulas
  - BM15: $sim(d_j, q) \sim$

$$G_2 + \sum_{k_i[q,d_j]} \frac{S_1 \; f_{i,j}}{(K_1 + f_{i,j})} \times \frac{S_3 \; f_{i,q}}{(K_3 + f_{i,q})} \times \log \frac{N - n_i + 0.5}{n_i + 0.5}$$

  - BM11: $sim(d_j, q) \sim$

$$G_2 + \sum_{k_i[q,d_j]} \frac{S_1 \; f_{i,j}}{\frac{K_1 \; len(d_j)}{avg\_doclen} + f_{i,j}} \times \frac{S_3 \; f_{i,q}}{(K_3 + f_{i,q})} \times \log \frac{N - n_i + 0.5}{n_i + 0.5}$$

  - where $k_i[q, d_j]$ is a short notation for $k_i \in q \wedge k_i \in d_j$

# BM1, BM11 and BM15 Formulas

- Experiments using TREC data indicates that BM11 outperforms BM15
- Some considerations can simplify the previous equations:
  - Empirical evidence suggests a best value of $K_2$ is $0$, which eliminates the $G_2$ factor from these equations
  - Further, good estimates for the scaling constants $S_1$ and $S_3$ are $K_1 + 1$ and $K_3 + 1$, respectively
  - Empirical evidence suggests that making $K_3$ very large is better
  - For short queries, we can assume that $f_{i,q}$ is 1 for all terms

# BM1, BM11 and BM15 Formulas

- These considerations lead to simpler equations as follows
  - BM15: $sim(d_j, q) \sim$

$$\sum_{k_i[q,d_j]} \frac{(K_1+1)f_{i,j}}{(K_1+f_{i,j})} \times \log \frac{N-n_i+0.5}{n_i+0.5}$$

  - BM11: $sim(d_j, q) \sim$

$$\sum_{k_i[q,d_j]} \frac{(K_1+1)f_{i,j}}{\frac{K_1\ len(d_j)}{avg\_doclen}+f_{i,j}} \times \log \frac{N-n_i+0.5}{n_i+0.5}$$

# BM25 Ranking Formula

- BM25 was created as a combination of the BM11 and BM15 ranking formulas
- The motivation was to combine the term frequency factors as follows

$$S_1 \times \frac{f_{i,j}}{K_1 \left((1-b) + b\frac{len(d_j)}{avg\_doclen}\right) + f_{i,j}}$$

  - where $b$ is a constant with values in the interval $[0,1]$
  - If $b = 0$, the equation reduces to the BM15 term frequency factor
  - If $b = 1$, it reduces to the BM11 term frequency factor
  - For values of $b$ between 0 and 1, the equation provides a combination of BM11 with BM15

# BM25 Ranking Formula

- In the place of equations BM11 and BM15, we can write
  - BM25: $sim(d_j, q) \sim$

$$\sum_{k_i[q,d_j]} \frac{(K_1+1)f_{i,j}}{K_1\ ((1-b)+b\frac{len(d_j)}{avg\_doclen})+f_{i,j}} \times \log\ \frac{N-n_i+0.5}{n_i+0.5}$$

  - where $K_1$ and $b$ are empirical constants
  - $K_1 = 1$ works well with real collections
  - $b$ should be kept closer to 1 to emphasize the document length normalization effect present in the BM11 formula
  - For instance, $b = 0.75$ is a reasonable assumption
  - Constants values can be fine tunned for particular collections through proper experimentation

# BM25 Ranking Formula

- Unlike probabilistic model, the BM25 formula can be computed without relevance information
- There is a consensus that BM25 outperforms classic vector model for general collections
- Thus, it has been used as a baseline for comparison, substituting the vector model

# Alternative Probabilistic Models

## Belief Network Models

# Bayesian Inference

- Probability Theory
  - Semantically clear
  - Computationally clumsy
- Why Bayesian Networks?
  - Clear formalism to combine evidences
  - Modularize the world (dependencies)
  - Bayesian Network Models for IR
    - Inference Network (Turtle & Croft, 1991)
    - Belief Network (Ribeiro-Neto & Muntz, 1996)

# Bayesian Inference

- Schools of thought in probability
  - *frequentist*
  - *epistemological*

# Bayesian Inference

- Basic Axioms:
  - $0 \leq P(A) \leq 1$;
  - $P(sure) = 1$;
  - $P(A \vee B) = P(A) + P(B)$ if $A$ and $B$ are mutually exclusive

# Bayesian Inference

- Other formulations
  - $P(A) = P(A \wedge B) + P(A \wedge \neg B)$
  - $P(A) = \sum_{\forall_i} P(A \wedge B_i)$, where $B_{i,\forall_i}$ is a set of exhaustive and mutually exclusive events
  - $P(A) + P(\neg A) = 1$
  - $P(A|K)$ belief in $A$ given the knowledge $K$
  - if $P(A|B) = P(A)$, we say: $A$ and $B$ are independent
  - if $P(A|B \wedge C) = P(A|C)$, we say: $A$ and $B$ are conditionally independent, given $C$
  - $P(A \wedge B) = P(A|B)P(B)$
  - $P(A) = \sum_{\forall_i} P(A|B_i)P(B_i)$

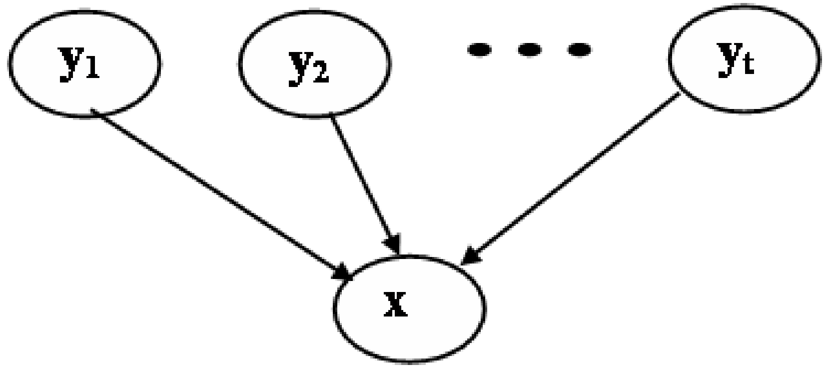
# Bayesian Inference

- **Bayes' Rule**: the heart of Bayesian techniques
  $$P(H|e) = \frac{P(e|H)P(H)}{P(e)}$$
  where
  - $H$ : a hypothesis and $e$ is an evidence
  - $P(H)$ : prior probability
  - $P(H|e)$ : posterior probability
  - $P(e|H)$ : probability of e if H is true
  - $P(e)$ : a normalizing constant, then we write:
    $P(H|e) \sim P(e|H)P(H)$

# Bayesian Networks

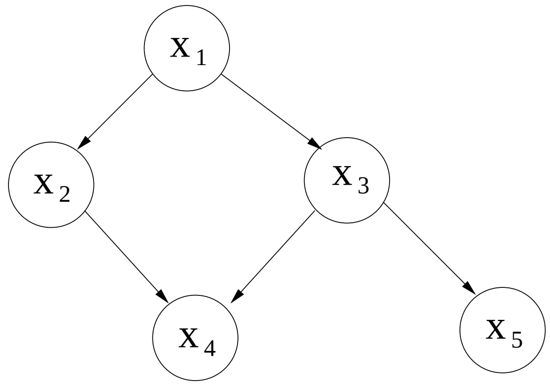
- Definition:
  - Bayesian networks are directed acyclic graphs (DAGS) in which the nodes represent random variables, the arcs portray causal relationships between these variables, and the strengths of these causal influences are expressed by conditional probabilities

# Bayesian Networks



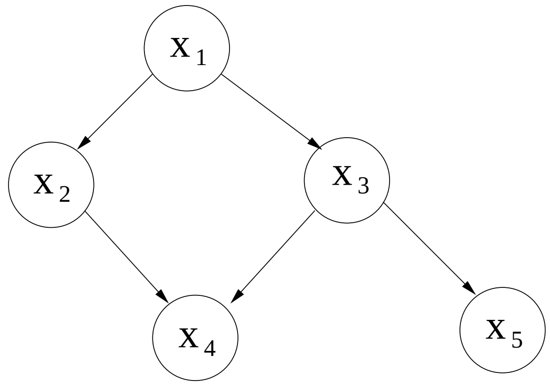

- $y_i$ : parent nodes (in this case, root nodes)
- $x$ : child node
- $y_i$ cause $x$
- $Y$ the set of parents of $x$
- The influence of $Y$ on $x$ can be quantified by any function $F(x, Y)$ such that $\sum_{\forall_x} F(x, Y) = 1$, $0 < F(x, Y) < 1$
  - For example, $F(x, Y) = P(x|Y)$

# Bayesian Networks




- Given the dependencies declared in a Bayesian Network, the expression for the joint probability can be computed as a product of **local** conditional probabilities, for example,
  $P(x_1, x_2, x_3, x_4, x_5) =$
  $P(x_1)P(x_2|x_1)P(x_3|x_1)P(x_4|x_2, x_3)P(x_5|x_3)$
  - $P(x_1)$ : prior probability of the root node

# Bayesian Networks




- In a Bayesian network each variable $x$ is conditionally independent of all its non-descendants, given its parents
- For example:

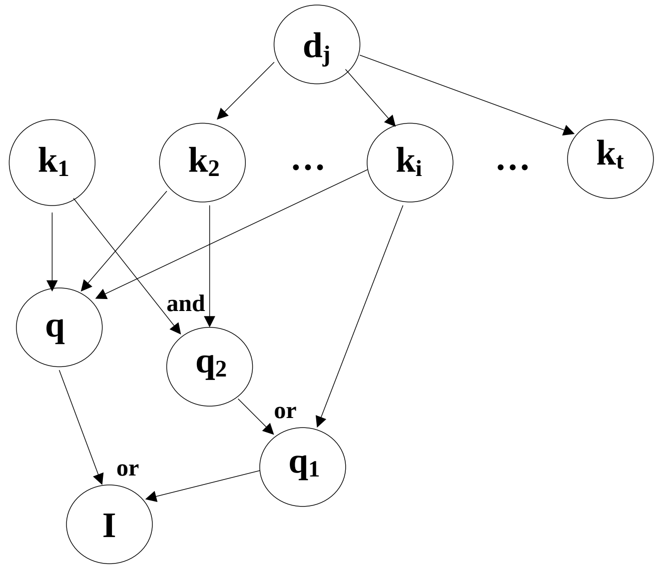
$$P(x_4, x_5 | x_2, x_3) = P(x_4 | x_2, x_3) P(x_5 | x_3)$$

# Inference Network Model

# Inference Network Model

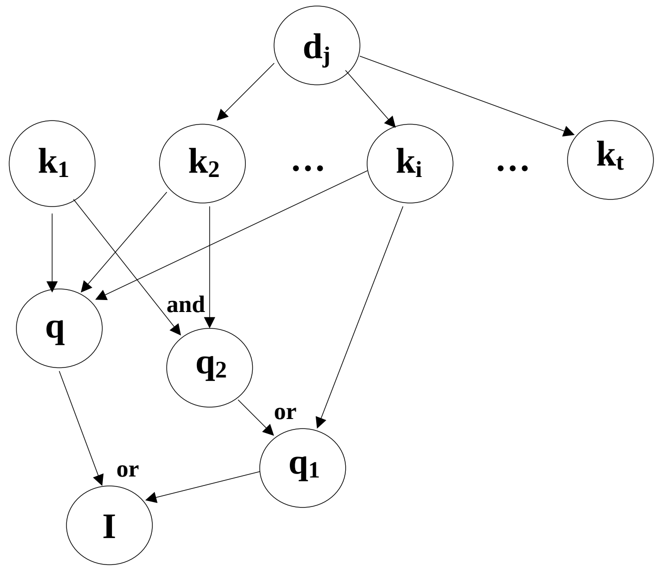
- Epistemological view of the IR problem
- Random variables associated with documents, index terms and queries
- A random variable associated with a document $d_j$ represents the event of observing that document

# Inference Network Model



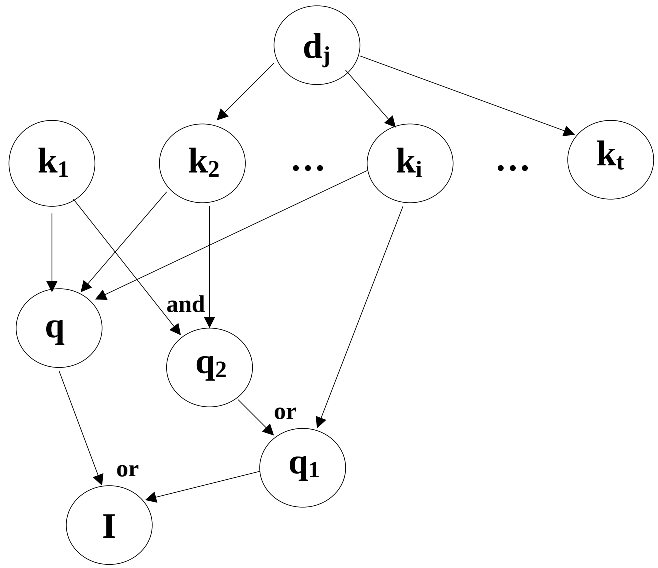

- Nodes
  - documents $(d_j)$
  - index terms $(k_i)$
  - queries $(q, q_1, and\ q_2)$
  - user information need $(I)$

# Inference Network Model




- Edges
  - from $d_j$ to its index term nodes $k_i$ indicate that the observation of $d_j$ increase the belief in the variables $k_i$

# Inference Network Model




- $d_j$ has index terms $k_2$, $k_i$, and $k_t$
- $q$ has index terms $k_1$, $k_2$, and $k_i$
- $q_1$ and $q_2$ model boolean formulation
- $q_1 = (k_1 \wedge k_2) \vee k_i$)
- $I = (q \vee q_1)$

# Inference Network Model

- Definitions:
  - $k_1$, $d_j$, and $q$ random variables
  - $\vec{k} = (k_1, k_2, \ldots, k_t)$ a t-dimensional vector
  - $k_i \in \{0, 1\}$, then $k$ has $2^t$ possible states
  - $d_j \in \{0, 1\}$; $q \in \{0, 1\}$
  - the ranking of a document $d_j$ is computed as $P(q \wedge d_j)$ where $q$ and $d_j$ are short representations for $q = 1$ and $d_j = 1$ ($d_j$ stands for a state where $d_j = 1$ and $\forall_{l \neq j} \Rightarrow d_l = 0$, because we observe one document at a time)

# Inference Network Model

$$
\begin{aligned}
P(q \wedge d_j) &= \sum_{\forall \vec{k}} P(q \wedge d_j | \vec{k}) \times P(\vec{k}) \\
&= \sum_{\forall \vec{k}} P(q \wedge d_j \wedge \vec{k}) \\
&= \sum_{\forall \vec{k}} P(q | d_j \times \vec{k}) \times P(d_j \times \vec{k}) \\
&= \sum_{\forall \vec{k}} P(q | \vec{k}) \times P(\vec{k} | d_j) \times P(d_j) \\
P(\overline{q \wedge d_j}) &= 1 - P(q \wedge d_j)
\end{aligned}
$$

# Inference Network Model

- As the instantiation of $d_j$ makes all index term nodes mutually independent $P(k|d_j)$ can be a product, then

$$\begin{aligned} P(q \wedge d_j) &= \sum_{\forall \vec{k}} P(q|\vec{k}) \times \\ &\left( \prod_{\forall i | g_i(\vec{k})=1} P(k_i|d_j) \times \prod_{\forall i | g_i(\vec{k})=0} P(\overline{k}_i|d_j) \right) \times P(d_j) \\ P(\overline{q \wedge d_j}) &= 1 - P(q \wedge d_j) \end{aligned}$$

# Inference Network Model

- The prior probability $P(d_j)$ reflects the probability associated to the event of observing a given document $d_j$
  - Uniformly for $N$ documents
    - $P(d_j) = \frac{1}{N}$
    - $P(\overline{d}_j) = 1 - \frac{1}{N}$
  - Based on norm of the vector $d_j$
    - $P(d_j) = \frac{1}{|\vec{d_j}|}$
    - $P(\overline{d}_j) = 1 - P(d_j)$

# Inference Network Model

- For the Boolean Model

$$
\begin{aligned}
P(d_j) &= \frac{1}{N} \\
P(\overline{d}_j) &= 1 - P(d_j)
\end{aligned}
$$

$$
\begin{aligned}
P(k_i|d_j) &= \begin{cases} 1 & \text{if } \; g_i(d_j) = 1 \\ 0 & \text{otherwise} \end{cases} \\
P(\overline{k}_i|d_j) &= 1 - P(k_i|d_j)
\end{aligned}
$$

$\Rightarrow$ only nodes associated with the index terms of the document $d_j$ are activated

# Inference Network Model

- For the Boolean Model

$$P(q|\vec{k}) = \begin{cases} 1 & \text{if } \exists \vec{q}_{cc} \mid (\vec{q}_{cc} \in \vec{q}_{dnf}) \wedge (\forall_{k_i},\ g_i(\vec{k}) = g_i(\vec{q}_{cc})) \\ 0 & \text{otherwise} \end{cases}$$

$$P(\overline{q}|\vec{k}) = 1 - P(q|\vec{k})$$

$\Rightarrow$ one of the conjunctive components of the query **must be** matched by the active index terms in **k**

# Inference Network Model

- For a *tf-idf* ranking strategy

$$
\begin{aligned}
P(d_j) &= \frac{1}{|\vec{d_j}|} \\
P(\overline{d}_j) &= 1 - P(d_j)
\end{aligned}
$$

$\Rightarrow$ prior probability reflects the importance of document **normalization**

# Inference Network Model

- For a *tf-idf* ranking strategy

$$
\begin{aligned}
P(k_i|d_j) &= f_{i,j} \\
P(\overline{k}_i|d_j) &= 1 - P(k_i|d_j)
\end{aligned}
$$

$\Rightarrow$ the relevance of the a index term $k_i$ is determined by its normalized term-frequency factor $f_{i,j} = \frac{freq_{i,j}}{max freq_{l,j}}$

# Inference Network Model

- For a *tf-idf* ranking strategy

  Define a vector $k_i$ given by

  $$\vec{k}_i = \vec{k} \ | \ (g_i(\vec{k}) = 1 \ \wedge \ \forall_{j \neq i} \ g_j(\vec{k}) = 0)$$

  $\Rightarrow$ in the state $k_i$ only the node $k_i$ is active and all the others are inactive

# Inference Network Model

- For a *tf-idf* ranking strategy

$$
\begin{aligned}
P(q|\vec{k}) &= \begin{cases} idf_i & \text{if } \vec{k} = \vec{k}_i \wedge g_i(\vec{q}) = 1 \\ 0 & \text{if } \vec{k} \neq \vec{k}_i \vee g_i(\vec{q}) = 0 \end{cases} \\
P(\overline{q}|\vec{k}) &= 1 - P(q|\vec{k})
\end{aligned}
$$

$\Rightarrow$ we can sum up the individual contributions of each index term by its normalized **idf**

# Inference Network Model

- For a *tf-idf* ranking strategy

  As $P(q|\vec{k}) = 0$ if $\vec{k} \neq \vec{k}_i$, we can rewrite $P(q \wedge d_j)$ as

  $$P(q \wedge d_j) =$$
  $$\sum_{\forall \vec{k}_i} P(q|\vec{k}_i) \times P(k_i|d_j) \times \left(\prod_{\forall\ l \neq i} P(\overline{k}_l|d_j)\right) \times P(d_j)$$

  $$= \left(\prod_{\forall\ i} P(\overline{k}_i|d_j)\right) \times P(d_j) \times \sum_{\forall \vec{k}_i} P(k_i|d_j) \times P(q|\vec{k}_i) \times \frac{1}{P(\overline{k}_i|d_j)}$$

# Inference Network Model

- For a *tf-idf* ranking strategy

  Applying the previous probabilities we have

$$P(q \wedge d_j) = C_j \times \frac{1}{|\vec{d_j}|} \times \sum_{\forall i | g_i(\vec{d_j})=1 \wedge g_i(\vec{q})=1} f_{i,j} \times idf_i \times \frac{1}{1-f_{i,j}}$$

  $\Rightarrow$ $C_j$ vary from document to document
  $\Rightarrow$ the ranking is distinct of the one provided by the vector model

# Inference Network Model

- Combining evidential source

  Let $I = q \vee q_1$

$$P(I \wedge d_j) = \sum_{\vec{k}} P(I|\vec{k}) \times P(\vec{k}|d_j) \times P(d_j)$$
$$= \sum_{\vec{k}} (1 - P(\overline{q}|\vec{k})\ P(\overline{q}_1|\vec{k})) \times P(\vec{k}|d_j) \times P(d_j)$$

  $\Rightarrow$ it might yield a retrieval performance which surpasses the retrieval performance of the query nodes in isolation (Turtle & Croft)

# Belief Network Model

# Belief Network Model

- As the Inference Network Model
  - Epistemological view of the IR problem
  - Random variables associated with documents, index terms and queries
- Contrary to the Inference Network Model
  - Clearly defined sample space
  - Set-theoretic view
  - Different network topology

# Belief Network Model

- The Probability Space
  - Define:
    - $K = \{k_1, \ldots, k_t\}$ the sample space (a concept space)
    - $u \subset K$ a subset of $K$ (a concept)
    - $k_i$ an index term (an elementary concept)
    - $\vec{k}$ = {$\vec{k}_1$, $\vec{k}_2$, . . ., $\vec{k}_t$} a vector associated to each $u$ such that $g_i(\vec{k}) = 1 \Longleftrightarrow k_i \in u$
    - $k_i$ a binary random variable associated with the index term $k_i$, ($k_i = 1 \Longleftrightarrow g_i(\vec{k}) = 1 \Longleftrightarrow k_i \in u$)

# Belief Network Model

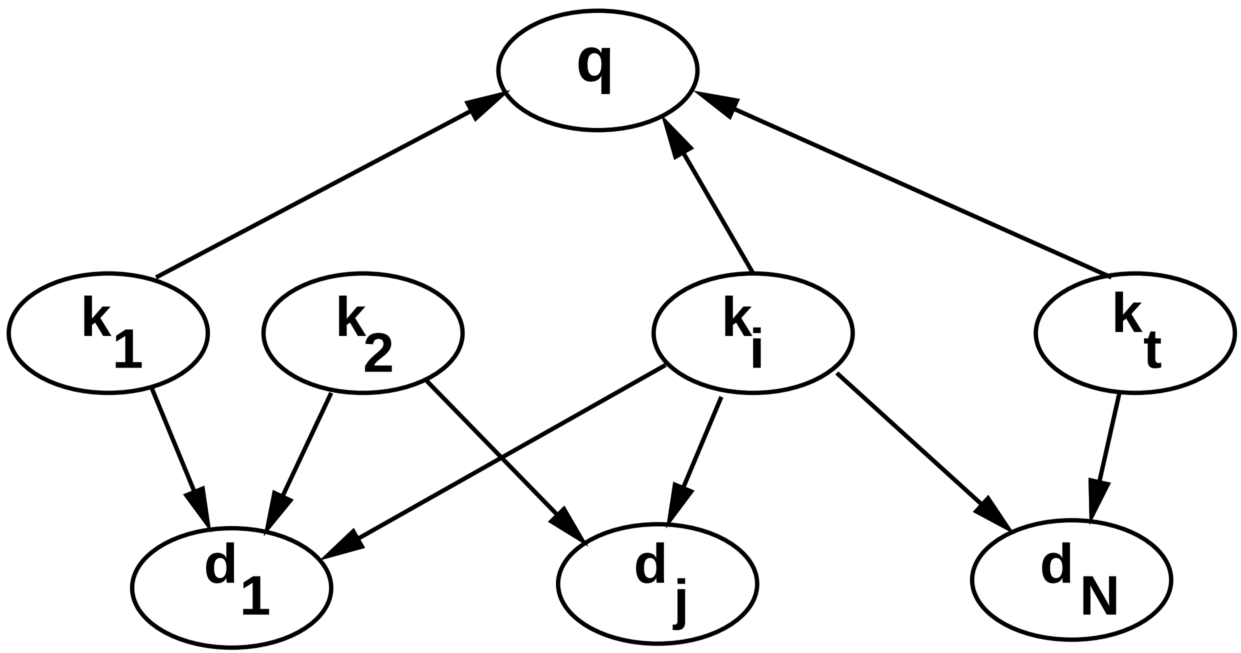
- A Set-Theoretic View
  - Define:
    - a document $d_j$ and query $q$ as concepts in $K$
    - a generic concept $c$ in $K$
    - a probability distribution $P$ over $K$, as

$$
\begin{aligned}
P(c) &= \sum_u P(c|u) \times P(u) \\
P(u) &= \left(\frac{1}{2}\right)^t
\end{aligned}
$$

    - $P(c)$ is the *degree of coverage* of the space $K$ by $c$

# Belief Network Model

- Network topology



# Belief Network Model

- Assumption
  - $P(d_j|q)$ is adopted as the rank of the document $d_j$ with respect to the query $q$. It reflects the degree of coverage provided to the concept $d_j$ by the concept $q$

# Belief Network Model

- The rank of $d_j$

$$P(d_j|q) = P(d_j \wedge q)/P(q)$$

$$P(d_j|q) \sim P(d_j \wedge q)$$

$$P(d_j|q) \sim \sum_{\forall u} P(d_j \wedge q|u) \times P(u)$$

$$P(d_j|q) \sim \sum_{\forall u} P(d_j|u) \times P(q|u) \times P(u)$$

$$P(d_j|q) \sim \sum_{\forall \vec{k}} P(d_j|\vec{k}) \times P(q|\vec{k}) \times P(\vec{k})$$

# Belief Network Model

- For the vector model
  - Define a vector $k_i$ given by

$$\vec{k}_i = \vec{k} \ | \ (g_i(\vec{k}) = 1 \ \wedge \ \forall_{j \neq i} \ g_j(\vec{k}) = 0)$$

$\Rightarrow$ in the state $k_i$ only the node $k_i$ is active and all the others are inactive

# Belief Network Model

- For the vector model
  - Define

$$P(q|\vec{k}) = \begin{cases} \frac{w_{i,q}}{\sqrt{\sum_{i=1}^{t} \; w_{i,q}^2}} & \text{if } \vec{k} = \vec{k}_i \; \wedge \; g_i(q) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$P(\overline{q}|\vec{k}) = 1 - P(q|\vec{k})$$

$\Rightarrow \frac{w_{i,q}}{\sqrt{\sum_{i=1}^{t} \; w_{i,q}^2}}$ is a normalized version of weight of the index term $k_i$ in the query $q$

# Belief Network Model

- For the vector model
  - Define

$$
\begin{aligned}
P(d_j|\vec{k}) &= \begin{cases} \frac{w_{i,j}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2}} & \text{if } \vec{k} = \vec{k_i} \ \wedge \ g_i(\vec{d_j}) = 1 \\ 0 & \text{otherwise} \end{cases} \\
P(\overline{d_j}|\vec{k}) &= 1 - P(d_j|\vec{k})
\end{aligned}
$$

$\Rightarrow \frac{w_{i,j}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2}}$ is a normalized version of the weight of the index term $k_i$ in the document $d_j$

# Bayesian Network Models

- Comparison
  - Inference Network Model is the first and well known
  - Belief Network adopts a set-theoretic view
  - Belief Network adopts a clearly define sample space
  - Belief Network provides a separation between query and document portions
  - Belief Network is able to reproduce any ranking produced by the Inference Network while the converse is not true (for example: the ranking of the standard vector model)

# Bayesian Network Models

- Computational costs
  - Inference Network Model one document node at a time then is linear on number of documents
  - Belief Network only the states that activate each query term are considered
  - The networks do not impose additional costs because the networks do not include cycles

# Bayesian Network Models

- Impact
  - The major strength is net combination of distinct evidential sources to support the rank of a given document

# Language Models

# Language Models

- We can represent each document of a collection by a smaller pre-selected subset of its terms
- To produce this shorter representation of the documents, an **indexing procedure** has to be designed
- This procedure is based on an **indexing model** for the document collection
  - For instance, a simplistic indexing model is to restrict the representation of a document to the nouns it contains

# Language Models

- *Ponte and Croft* propose a method of using indexing models to improve the quality of search results
- Given an indexing model for a set of documents, we can inquire about the likelihood that this **indexing model generates a given user query**

# Language Models

- In speech recognition, probability distributions of terms can be used to predict the likelihood that the next token in the spoken sequence is a given word
- These probability distributions are called **language models**
- Notice the analogy with the idea of using a model of the documents to predict the likelihood of observing a given user query
- Because of this parallel, *Ponte and Croft* called the IR model presented here of **language model**

# Language Models

- Given a document $d_j$, let $M_j$ be a reference to a language model for that document
- Also, let $P(q|M_j)$ the probability of generating a user query $q$ from the language model of the document $d_j$
- Assuming independence of index terms, we can compute $P(q|M_j)$ from the term probabilities $P(k_i|M_j)$
- A simple estimate of term probabilities is

$$P(k_i|M_j) = \frac{f_{i,j}}{\sum_i f_{i,j}}$$

# Language Models

- The probability of producing the user query $q$ (from document $d_j$) could then be computed as

$$P(q|M_j) = \prod_{k_i \in q} P(k_i|M_j)$$

- However, this formulation doesn't allow partial matches
- To overcome this restriction, we assume that a non-occuring term is related to the document with the probability of observing that term in the whole collection

$$P(k_i|M_j) = \begin{cases} \frac{f_{i,j}}{\sum_i f_{i,j}} & \text{if } f_{i,j} > 0 \\ \frac{F_i}{\sum_i F_i} & \text{if } f_{i,j} = 0 \end{cases}$$

where $F_i$ is given by $\sum_j f_{i,j}$, as before

# Language Models

- However, this term probability estimation is based on a sample composed of a single document
- To make the model more resilient, we need an estimate that is based on a larger sample of documents
- This can be accomplished through an average computation as follows

$$P(k_i) = \frac{\sum_{j|k_i \in d_j} P(k_i|M_j)}{n_i}$$

- That is, $P(k_i)$ is an estimate based on the language models of all documents that contain term $k_i$

# Language Models

- $P(k_i)$ is a much more robust statistics
- However, it is the same for all documents that contain term $k_i$
- That is, using $P(k_i)$ to predict the generation of term $k_i$ by the language model of a particular document $d_j$ involves a risk
- To fix this, let us define the average frequency $\overline{f}_{i,j}$ of term $k_i$ in document $d_j$, that is given by

$$\overline{f}_{i,j} = P(k_i) * \sum_i f_{i,j}$$

# Language Models

- The risk $R_{i,j}$ associated with using $\overline{f}_{i,j}$ can be quantified by a geometric distribution as follows

$$R_{i,j} = \left(\frac{1}{1+\overline{f}_{i,j}}\right) \times \left(\frac{\overline{f}_{i,j}}{1+\overline{f}_{i,j}}\right)^{f_{i,j}}$$

- As $f_{i,j}$ becomes further away from the average frequency $\overline{f}_{i,j}$, the average probability $P(k_i)$ becomes riskier to use as an estimate

# Language Models

- The notion of risk associated with $P(k_i)$ is used as a mixing parameter to allow better estimation of $\overline{P}(k_i|M_j)$
- We use the risk factor as follows

$$\overline{P}(k_i|M_j) = \begin{cases} P(k_i|M_j)^{(1-R_{i,j})} \times P(k_i)^{R_{i,j}} & \text{if } f_{i,j} > 0 \\ \frac{F_i}{\sum_i F_i} & \text{otherwise} \end{cases}$$

- By combining the separate factors $\overline{P}(k_i|M_j)$, we write

$$\overline{P}(q|M_j) = \prod_{k_i \in q} \overline{P}(k_i|M_j) \times \prod_{k_i \notin q} [1 - \overline{P}(k_i|M_j)]$$

which combines the probability of generating the query terms from the language (document) model with the probability of not generating the terms not in the query

# Structured Text Models

# Introduction

- Keyword-based query answering considers that the documents are flat, i.e., a word in the title has the same weight as a word in the body of the document
- But, the document structure is one additional piece of information which can be taken advantage of
- For instance, words appearing in the title or in sub-titles within the document could receive higher

# Introduction

- Consider the following information need:
  - Retrieve all documents which contain a page in which the string "atomic holocaust" appears in italic in the text surrounding a Figure whose label contains the word *earth*
- The corresponding query could be:
  - same-page( near( "atomic holocaust", Figure( label( "earth" ))))

# Introduction

- Advanced interfaces that facilitate the specification of the structure are also highly desirable
- Models which allow combining information on text content with information on document structure are called *structured text models*
- Structured text models include no ranking (open research problem)

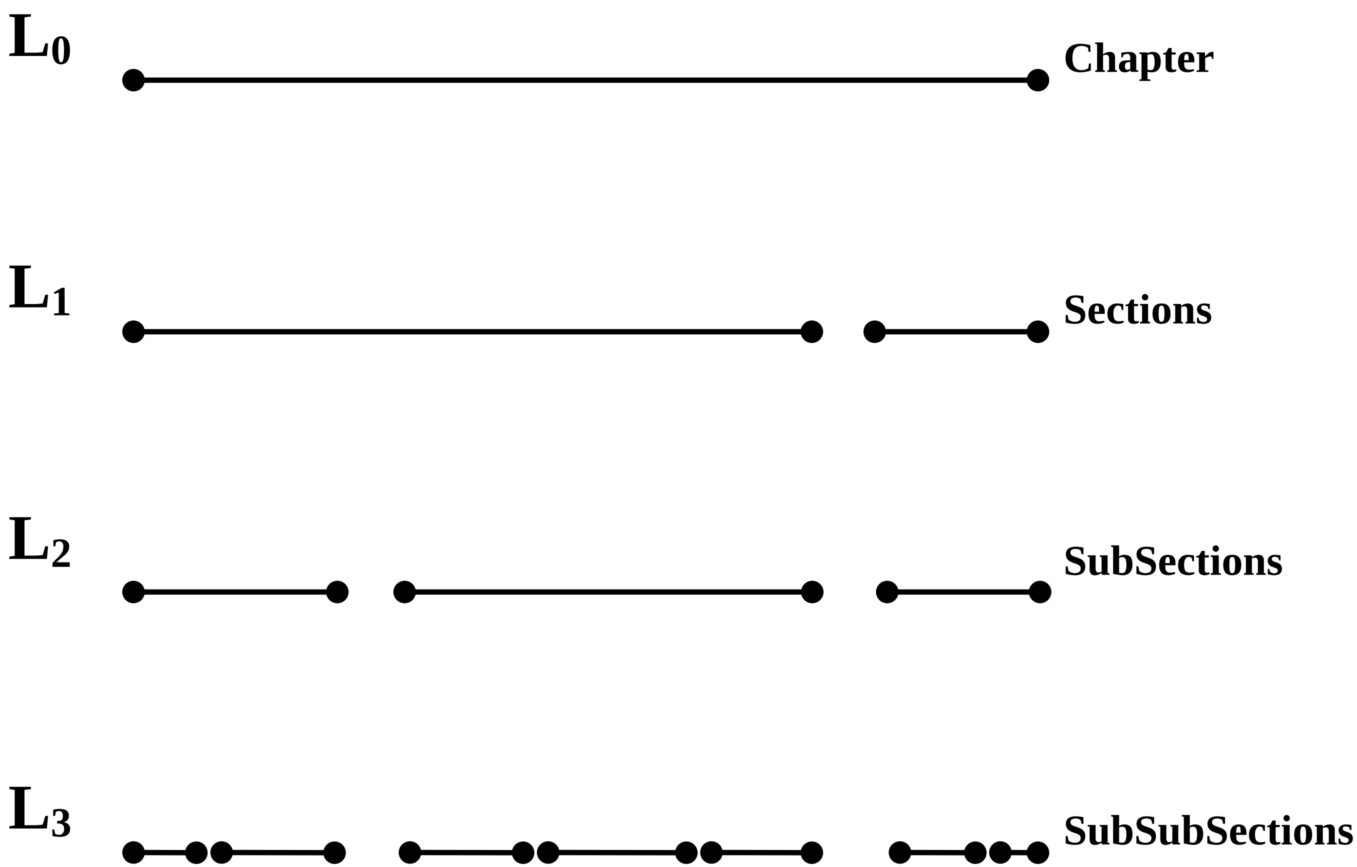
# Basic Definitions

- *Match point*: the position in the text of a sequence of words that match the query
  - Query: “atomic holocaust in Hiroshima”
  - Doc $d_j$: contains 3 lines with this string
  - Then, doc $d_j$ contains 3 match points
- *Region*: a contiguous portion of the text
- Node: a structural component of the text such as a chapter, a section, etc.

# Non-Overlapping Lists

- Due to Burkowski, 1992
- Idea: divide the text in *non-overlapping* regions which are collected in a *list*
- Multiple ways to divide the text in non-overlapping parts yield multiple lists:
  - a list for chapters
  - a list for sections
  - a list for subsections
- Text regions from distinct lists might overlap

# Non-Overlapping Lists



# Non-Overlapping Lists

- Implementation:
  - single inverted file that combines keywords and text regions
  - to each entry in this inverted file is associated a list of text regions
  - lists of text regions can be merged with lists of keywords

# Non-Overlapping Lists

- Regions are non-overlapping which limits the queries that can be asked
- Types of queries:
  - select a region that contains a given word
  - select a region A that does not contain a region B (regions A and B belong to distinct lists)
  - select a region not contained within any other region

# Conclusions

- The non-overlapping lists model is simple and allows efficient implementation
- But, types of queries that can be asked are limited
- Also, model does not include any provision for ranking the documents by degree of similarity to the query
- What does structural similarity mean?

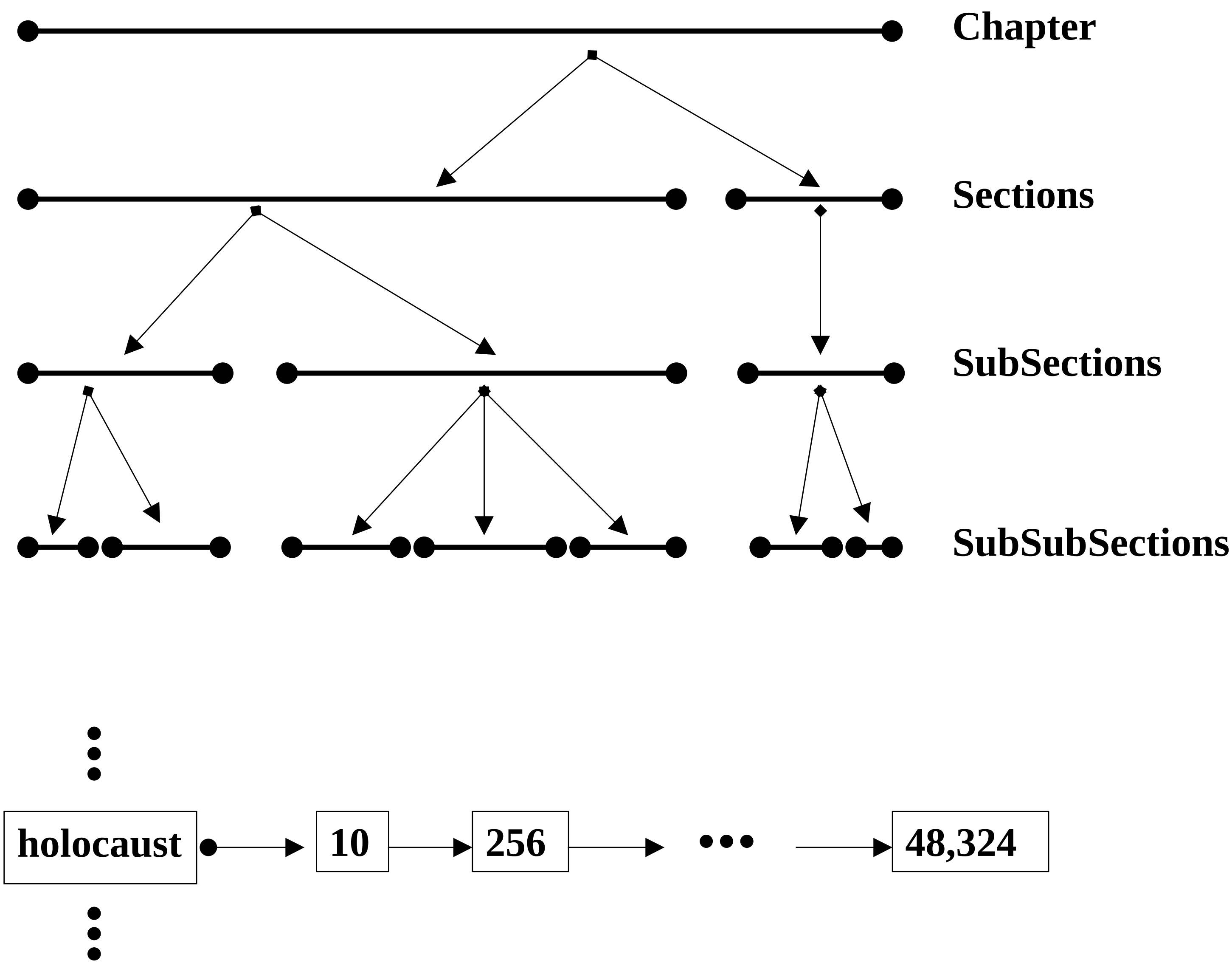
# Proximal Nodes

- Due to Navarro and Baeza-Yates, 1997
- Idea: define a strict hierarchical index over the text. This enriches the previous model that used flat lists
- Multiple index hierarchies might be defined
- Two distinct index hierarchies might refer to text regions that overlap

# Definitions

- Each indexing structure is a strict hierarchy composed of
  - chapters
  - sections
  - subsections
  - paragraphs
  - lines
- Each of these components is called a *node*
- To each node is associated a text region

# Proximal Nodes



# Proximal Nodes

- Key points:
  - In the hierarchical index, one node might be contained within another node
  - But, two nodes of a same hierarchy cannot overlap
  - The inverted list for keywords complements the hierarchical index
  - The implementation here is more complex than that for non-overlapping lists

# Proximal Nodes

- Queries are now regular expressions:
  - search for strings
  - references to structural components
  - combination of these
- Model is a compromise between expressiveness and efficiency
- Queries are simple but can be processed efficiently
- Further, model is more expressive than non-overlapping lists

# Proximal Nodes

- Query: find the sections, the subsections, and the subsubsections that contain the word “holocaust”
  - .[(*section) with (“holocaust”)]
- Simple query processing:
  - traverse the inverted list for “holocaust” and determine all match points
  - use the match points to search in the hierarchical index for the structural components

# Proximal Nodes

- Query: [(*section) with (“holocaust”)]
- Sophisticated query processing:
  - get the first entry in the inverted list for “holocaust”
  - use this match point to search in the hierarchical index for the structural components
  - Innermost matching component: smaller one
  - Check if innermost matching component includes the second entry in the inverted list for “holocaust”
  - If it does, check the third entry and so on
  - This allows matching efficiently the nearby (or proximal) nodes

# Conclusions

- Model allows formulating queries that are more sophisticated than those allowed by non-overlapping lists
- To speed up query processing, nearby nodes are inspected
- Types of queries that can be asked are somewhat limited (all nodes in the answer must come from a same index hierarchy!)
- Model is a compromise between efficiency and expressiveness