Chapter 4

Modeling

- BM25
- Language Models
- Divergence from Randomness
- Belief Network Models
- Other Models

Alternative Probabilistic Models

Alternative Probabilistic Models

- We discuss four alternative probabilistic models:
  - **BM25**
  - **Language Models**
  - **Divergence from Randomness**
  - **Belief Network Models**

Alternative Probabilistic Models

BM25 (Best Match 25)

BM25 (Best Match 25)

- BM25 was created as the result of a series of experiments on variations of the probabilistic model
- 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 with the Okapi system, which led to 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 first idea for improving the ranking was to introduce a **term-frequency** in BM1 ranking formula
- This factor, after some changes, evolved to become

$$\mathcal{F}_{i,j} = S_1 \times \frac{f_{i,j}}{K_1 + f_{i,j}}$$

  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 $K_1 = 0$, 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

$$\mathcal{F}'_{i,j} = S_1 \times \frac{f_{i,j}}{\frac{K_1 \times len(d_j)}{avg\_doclen} + f_{i,j}}$$

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

- Next, a correction factor $G_{j,q}$ dependent on the document and query lengths was added

$$\mathcal{G}_{j,q} = K_2 \times len(q) \times \frac{avg\_doclen - len(d_j)}{avg\_doclen + len(d_j)}$$

  where

  - $len(q)$ is the query length (number of terms in the query)
  - K_2 is a constant

BM1, BM11 and BM15 Formulas

- An additional factor was applied to term frequencies within queries

$$\mathcal{F}_{i,q} = S_3 \times \frac{f_{i,q}}{K_3 + f_{i,q}}$$

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 leads to various BM (Best Matching) formulas

$$sim_{BM1}(d_j, q) \sim \sum_{k_i[q,d_j]} \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right)$$

$$sim_{BM15}(d_j, q) \sim \mathcal{G}_{j,q} + \sum_{k_i[q,d_j]} \mathcal{F}_{i,j} \times \mathcal{F}_{i,q} \times \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right)$$

$$sim_{BM11}(d_j, q) \sim \mathcal{G}_{j,q} + \sum_{k_i[q,d_j]} \mathcal{F}'_{i,j} \times \mathcal{F}_{i,q} \times \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right)$$

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, and then $\mathcal{F}_{i,q}$ factor is reduced to $f_{i,q}$ simply
  - 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

$$sim_{BM1}(d_j, q) \sim \sum_{k_i[q,d_j]} \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right)$$

$$sim_{BM15}(d_j, q) \sim \sum_{k_i[q,d_j]} \frac{(K_1 + 1) f_{i,j}}{(K_1 + f_{i,j})} \times \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right)$$

$$sim_{BM11}(d_j, q) \sim \sum_{k_i[q,d_j]} \frac{(K_1 + 1) f_{i,j}}{\frac{K_1 \; len(d_j)}{avg\_doclen} + f_{i,j}} \times \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right)$$

BM25 Ranking Formula

- BM25: combination of the BM11 and BM15
- The motivation was to combine the term frequency factors as follows

$$\mathcal{B}_{i,j} = \frac{(K_1 + 1) f_{i,j}}{K_1 \left[(1 - b) + b \frac{len(d_j)}{avg\_doclen} \right] + f_{i,j}}$$

where b is a constant with values in the interval $[0, 1]$

  - If $b = 0$, it 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

- The ranking equation for the BM25 model can then be written as

$$sim_{BM25}(d_j, q) \sim \sum_{k_i[q,d_j]} \mathcal{B}_{i,j} \times \log \left(\frac{N - n_i + 0.5}{n_i + 0.5} \right)$$

  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

Language Models

Language Models

- Language models are used in many natural language processing applications
  - Ex: part-of-speech tagging, speech recognition, machine translation, and information retrieval
- To illustrate, the regularities in spoken language can be modeled by probability distributions
- These distributions can be used to predict the likelihood that the next token in the sequence is a given word
- These probability distributions are called **language models**

Language Models

- The key idea
  - To define language models for documents and use them to predict the likelihood of observing the query terms
  - By ordering these probabilities, a ranking of the documents is produced

Statistical Foundation

- Let S be a sequence of r consecutive terms that occur in a document of the collection:

$$S = k_1, k_2, \ldots, k_r$$

- An n-gram language model uses a Markov process to assign a probability of occurrence to S:

$$P_n(S) = \prod_{i=1}^{r} P(k_i | k_{i-1}, k_{i-2}, \ldots, k_{i-(n-1)})$$

  where n is the order of the Markov process

- The occurrence of a term depends on observing the $n - 1$ terms that precede it in the text

Statistical Foundation

- **Bigram language model** ($n = 2$): the estimatives are based on the co-occurrence of pairs of words
- **Unigram language model** ($n = 1$): the estimatives are based on the occurrence of individual words
- Higher order models such as *trigram* language models ($n = 3$) are usually adopted for speech recognition
- **Term independence assumption**: in the case of IR, the impact of word order is less clear
- As a result, unigram models have been used extensively

Bernoulli process

- Given a document d_j, let M_j be a reference to a language model for that document
- M_j should allow estimating the probability of generating a user query q from the model: $P(q|M_j)$
- If we assume independence of index terms, we can compute $P(q|M_j)$ using a multivariate Bernoulli process:

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

  where $P(k_i|M_j)$ are term probabilities
- This is analogous to the expression for ranking computation in the classic probabilistic model

Bernoulli process

- A simple estimate of the term probabilities is

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

  which computes the probability that term k_i will be produced by a random draw (taken from d_j)

- However, the probability will become zero if k_i does not occur in the document
- Thus, we assume that a non-occurring term is related to d_j with the probability $P(k_i|C)$ of observing k_i in the whole collection C

Bernoulli process

- $P(k_i|C)$ can be estimated in different ways
- For instance, Hiemstra suggests an idf-like estimative:

$$P(k_i|C) = \frac{n_i}{\sum_i n_i}$$

where n_i is the number of docs in which k_i occurs

- Miller, Leek, and Schwartz suggest

$$P(k_i|C) = \frac{F_i}{\sum_i F_i} \quad \text{where} \quad F_i = \sum_j f_{i,j}$$

- This last equation for $P(k_i|C)$ is adopted here

Bernoulli process

- As a result, we redefine $P(k_i|M_j)$ as follows:

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

- In this expression, $P(k_i|M_j)$ estimation is based only on the document d_j when $f_{i,j} > 0$
- This is clearly undesirable because it leads to instability in the model

Bernoulli process

- This drawback 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
- 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 M_j involves a risk

Bernoulli process

- To fix this, let us define the average frequency $\overline{f}_{i,j}$ of term k_i in document d_j as

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

Bernoulli process

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

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

- For terms that occur very frequently in the collection, $\overline{f}_{i,j} \gg 0$ and $R_{i,j} \sim 0$
- For terms that are rare both in the document and in the collection, $f_{i,j} \sim 1$, $\overline{f}_{i,j} \sim 1$, and $R_{i,j} \sim 0.25$

Bernoulli process

- Let us refer the probability of observing term k_i according to the language model M_j as $P_R(k_i|M_j)$
- We then use the risk factor $R_{i,j}$ to compute $P_R(k_i|M_j)$, as follows

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

- In this formulation, if $R_{i,j} \sim 0$ then $P_R(k_i|M_j)$ is basically a function of $P(k_i|M_j)$
- Otherwise, it is a mix of $P(k_i)$ and $P(k_i|M_j)$

Bernoulli process

- Substituting into original $P(q|M_j)$ Equation, we obtain

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

  which computes the probability of generating the query from the language (document) model

- This is the basic formula for ranking computation in a language model

Multinomial Process

- Ranking in a language model is provided by estimating $P(q|M_j)$
- Several researchs employed a multinomial process to generate the query
- According to this process, if we assume that terms are independent among themselves (unigram model):

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

Multinomial Process

- By taking logs on both sides

$$\begin{aligned}
\log P(q|M_j) &= \sum_{k_i \in q} \log P(k_i|M_j) \\
&= \sum_{k_i \in q \wedge d_j} \log P_{\in}(k_i|M_j) + \sum_{k_i \in q \wedge \neg d_j} \log P_{\notin}(k_i|M_j) \\
&= \sum_{k_i \in q \wedge d_j} \log \left(\frac{P_{\in}(k_i|M_j)}{P_{\notin}(k_i|M_j)} \right) + \sum_{k_i \in q} \log P_{\notin}(k_i|M_j)
\end{aligned}$$

where $P_{\in}$ and $P_{\notin}$ are two distinct probability distributions:

- The first is a distribution for the query terms in the document
- The second is a distribution for the query terms not in the document

Multinomial Process

- For the second distribution, statistics are derived from all the document collection
- Thus, we can write

$$P_{\not\in}(k_i|M_j) = \alpha_j P(k_i|C)$$

where α_j is a parameter associated with document d_j and $P(k_i|C)$ is a collection C language model

Multinomial Process

- Thus, we obtain

$$
\begin{aligned}
\log P(q|M_j) &= \sum_{k_i \in q \wedge d_j} \log \left(\frac{P_{\in}(k_i|M_j)}{\alpha_j P(k_i|C)} \right) + n_q \log \alpha_j + \sum_{k_i \in q} \log P(k_i|C) \\
&\sim \sum_{k_i \in q \wedge d_j} \log \left(\frac{P_{\in}(k_i|M_j)}{\alpha_j P(k_i|C)} \right) + n_q \log \alpha_j
\end{aligned}
$$

where n_q stands for the query length and the last sum was dropped because it is constant for all documents

Multinomial Process

- The ranking function is now composed of two separate parts
- The **first part** assigns weights to each query term that appears in the document, according to the expression

$$\log\left(\frac{P_{\in}(k_i|M_j)}{\alpha_j P(k_i|C)}\right)$$

- This term weight plays a role analogous to the idf weight in the vector model
- Further, the parameter α_j can be used for document length normalization

Multinomial Process

- The **second part** assigns a fraction of probability mass to the query terms that are not in the document
- The combination of a multinomial process with smoothing leads to a ranking formula that naturally includes tf, idf, and document length normalization
- That is, smoothing plays a key role in modern language modeling, as we now discuss

Smoothing

- In our discussion, we estimated $P_{\not\in}(k_i|M_j)$ using $P(k_i|C)$
- This approach avoids assigning zero probability to terms that are not in the document
- It is called **smoothing** and is important for fine tuning the ranking function and improving results
- One popular technique for smoothing is to move some mass probability from the (query) terms in the document to the terms not in the document

$$P(k_i|M_j) = \begin{cases} P_{\in}(k_i|M_j) & \text{if } k_i \in d_j \\ \alpha_j P(k_i|C) & \text{otherwise} \end{cases}$$

Smoothing

- Since $\sum_i P(k_i|M_j) = 1$, we can write

$$\sum_{k_i \in d_j} P_{\in}(k_i|M_j) + \sum_{k_i \not\in d_j} \alpha_j P(k_i|C) = 1$$

- That is,

$$\alpha_j = \frac{1 - \sum_{k_i \in d_j} P_{\in}(k_i|M_j)}{1 - \sum_{k_i \in d_j} P(k_i|C)}$$

Smoothing

- Under the above assumptions, the smoothing parameter α_j is also a function of $P_{\in}(k_i|M_j)$
- As a result, distinct smoothing methods can be obtained through distinct specifications of $P_{\in}(k_i|M_j)$
- Examples of smoothing methods:
  - Jelinek-Mercer Method
  - Bayesian Smoothing using Dirichlet Priors

Jelinek-Mercer Method

- The idea is to do a linear interpolation between the document frequency and the collection frequency distributions:

$$P_{\in}(k_i|M_j,\lambda) = (1-\lambda)\frac{f_{i,j}}{\sum_i f_{i,j}} + \lambda\frac{F_i}{\sum_i F_i}$$

  where $0 \leq \lambda \leq 1$

- Closer is λ to 0, higher is the influence of the term document frequency
- As λ moves towards 1, higher is the influence of the term collection frequency
- The larger the values of λ, the larger is the effect of smoothing

Dirichlet smoothing

- In this method, the language model is a multinomial distribution
- In this distribution, the conjugate prior probabilities are given by the Dirichlet distribution
- This leads to

$$P_{\in}(k_i|M_j, \lambda) = \frac{f_{i,j} + \lambda \frac{F_i}{\sum_i F_i}}{\sum_i f_{i,j} + \lambda}$$

- As before, closer is λ to 0, higher is the influence of the term document frequency
- As λ moves towards 1, the influence of the term collection frequency increases

Dirichlet smoothing

- Contrary to the Jelinek-Mercer method, this influence is always partially mixed with the document frequency
- As before, the larger the values of λ, the larger is the effect of smoothing

Smoothing Computation

- In both smoothing methods above, computation can be carried out efficiently
- All frequency counts can be obtained directly from the index
- The values of α_j can be precomputed for each document
- Thus, the complexity is analogous to the computation of a vector space ranking using tf-idf weights

Applying Smoothing to Ranking

- The IR ranking in a multinomial language model is computed using Equation below as follows:

$$\log P(q|M_j) = \sum_{k_i \in q \wedge d_j} \log \left(\frac{P_{\in}(k_i|M_j)}{\alpha_j P(k_i|C)} \right) + n_q \log \alpha_j$$

  - compute $P_{\in}(k_i|M_j)$ using a smoothing method
  - compute $P(k_i|C)$ using $\frac{n_i}{\sum_i n_i}$ or $\frac{F_i}{\sum_i F_i}$ Equation
  - compute α_j from the Equation $\alpha_j = \frac{1 - \sum_{k_i \in d_j} P_{\in}(k_i|M_j)}{1 - \sum_{k_i \in d_j} P(k_i|C)}$
  - compute the ranking

Alternative Probabilistic Models

Divergence from Randomness

Divergence from Randomness

- A distinct probabilistic model has been proposed by Amati and Rijsbergen
- The idea is to compute term weights by measuring the divergence between a term distribution produced by a random process and the actual term distribution
- Thus, the name **divergence from randomness**
- The model is based on two fundamental assumptions, as follows

Divergence from Randomness

- First assumption:
  - Not all words are equally important for describing the content of the documents
  - Words that carry little information are assumed to be **randomly distributed** over the whole document collection C
  - Given a term k_i, its probability distribution over the whole collection is referred to as $P(k_i|C)$
  - The amount of information associated with this distribution is given by $-\log\ P(k_i|C)$
  - By specifying this distribution in different ways, we can implement distinct **notions of randomness** of the term in the collection

Divergence from Randomness

- Second assumption:
  - A complementary term distribution can be obtained by considering just the subset of documents that contain term k_i
  - This subset is referred to as the **elite set**
  - The corresponding probability distribution, computed with regard to document d_j, is referred to as $P(k_i|d_j)$
  - Smaller the probability of observing a term k_i in a document d_j, more rare and important is the term considered to be
  - Thus, the amount of information associated with the term in the elite set is defined as $1 - P(k_i|d_j)$

Divergence from Randomness

- Given these assumptions, the weight $w_{i,j}$ of a term k_i in a document d_j is defined as

$$w_{i,j} = (-\log \; P(k_i|C)) \; \times \; (1 - P(k_i|d_j))$$

- Two distribution of the term are considered: in the collection and in the subset of docs in which it occurs
- The rank $R(d_j, q)$ of a document d_j with regard to a query q is then computed as

$$R(d_j, q) = \sum_{k_i \in q} f_{i,q} \times w_{i,j}$$

where $f_{i,q}$ is the frequency of term k_i in the query

Random Distribution

- To compute the distribution of terms in the collection, distinct probability models can be considered
- For instance, consider that Bernoulli trials are used to model the occurrences of a term in the collection
- To illustrate, consider a collection with 1,000 documents and a term k_i that occurs 10 times in the collection
- Then, the probability of observing 4 occurrences of term k_i in a document is given by

$$P(k_i|C) = \binom{10}{4} \left(\frac{1}{1000}\right)^4 \left(1 - \frac{1}{1000}\right)^6$$

which is a standard binomial distribution

Random Distribution

- In general, let $p = 1/N$ be the probability of observing a term in a document, where N is the number of docs
- The probability of observing $f_{i,j}$ occurrences of term k_i in document d_j is described by a binomial distribution:

$$P(k_i|C) = \binom{F_i}{f_{i,j}} p^{f_{i,j}} \times (1-p)^{F_i - f_{i,j}}$$

- Define

$$\lambda_i = p \times F_i$$

and assume that $p \to 0$ when $N \to \infty$, but that $\lambda_i = p \times F_i$ remains constant

Random Distribution

- Under these conditions, we can aproximate the binomial distribution by a Poisson process, which yields

$$P(k_i|C) = \frac{e^{-\lambda_i} \; \lambda_i^{f_{i,j}}}{f_{i,j}!}$$

Random Distribution

- The amount of information associated with term k_i in the collection can then be computed as

$$
\begin{aligned}
-\log\ P(k_i|C) &= -\log\left(\frac{e^{-\lambda_i}\ \lambda_i^{f_{i,j}}}{f_{i,j}!}\right) \\
&\approx -f_{i,j}\log\lambda_i + \lambda_i\log e + \log(f_{i,j}!) \\
&\approx f_{i,j}\log\left(\frac{f_{i,j}}{\lambda_i}\right) + \left(\lambda_i + \frac{1}{12 f_{i,j}+1} - f_{i,j}\right)\log e \\
&\quad + \frac{1}{2}\ \log(2\pi f_{i,j})
\end{aligned}
$$

in which the logarithms are in base 2 and the factorial term $f_{i,j}!$ was approximated by the **Stirling's formula**

$$
f_{i,j}! \approx \sqrt{2\pi}\ f_{i,j}^{(f_{i,j}+0.5)}\ e^{-f_{i,j}}\ e^{(12 f_{i,j}+1)^{-1}}
$$

Random Distribution

- Another approach is to use a Bose-Einstein distribution and approximate it by a geometric distribution:

$$P(k_i|C) \approx p \times p^{f_{i,j}}$$

where $p = 1/(1 + \lambda_i)$

- The amount of information associated with term k_i in the collection can then be computed as

$$-\log P(k_i|C) \approx -\log \left(\frac{1}{1+\lambda_i} \right) - f_{i,j} \times \log \left(\frac{\lambda_i}{1+\lambda_i} \right)$$

which provides a second form of computing the term distribution over the whole collection

Distribution over the Elite Set

- The amount of information associated with term distribution in elite docs can be computed by using Laplace's law of succession

$$1 - P(k_i|d_j) = \frac{1}{f_{i,j} + 1}$$

- Another possibility is to adopt the ratio of two Bernoulli processes, which yields

$$1 - P(k_i|d_j) = \frac{F_i + 1}{n_i \times (f_{i,j} + 1)}$$

where n_i is the number of documents in which the term occurs, as before

Normalization

- These formulations do not take into account the length of the document d_j
- Distinct normalizations can be used, such as

$$f'_{i,j} = f_{i,j} \times \frac{avg\_doclen}{len(d_j)}$$

or

$$f'_{i,j} = f_{i,j} \times \log\left(1 + \frac{avg\_doclen}{len(d_j)}\right)$$

where avg_doclen is the average document length in the collection and $len(d_j)$ is the length of document d_j

Normalization

- To compute $w_{i,j}$ weights using normalized term frequencies, just substitute the factor $f_{i,j}$ by $f'_{i,j}$
- In here we consider that a same normalization is applied for computing $P(k_i|C)$ and $P(k_i|d_j)$
- By combining different forms of computing $P(k_i|C)$ and $P(k_i|d_j)$ with different normalizations, various ranking formulas can be produced

Bayesian Network Models

Bayesian Inference

- One approach for developing a probabilistic model of IR is to use **Bayesian belief networks**
- Belief networks provide a clean formalism for combining distinct sources of evidence
  - Ex: past queries, past feedback cycles, and distinct query formulations
- In here we discuss two models:
  - **Inference network**, proposed by Turtle and Croft
  - **Belief network model**, proposed by Ribeiro-Neto and Muntz
- Before proceeding, we briefly introduce Bayesian networks

Bayesian Networks

- Bayesian networks are **directed acyclic graphs (DAGs)** in which
  - **the nodes** represent random variables
  - **the arcs** portray causal relationships between these variables
  - **the strengths** of these causal influences are expressed by conditional probabilities
- The **parents** of a node are those judged to be direct causes for it
- This causal **relationship** is represented by a link directed from each parent node to the child node
- The **roots** of the network are the nodes without parents

Bayesian Networks

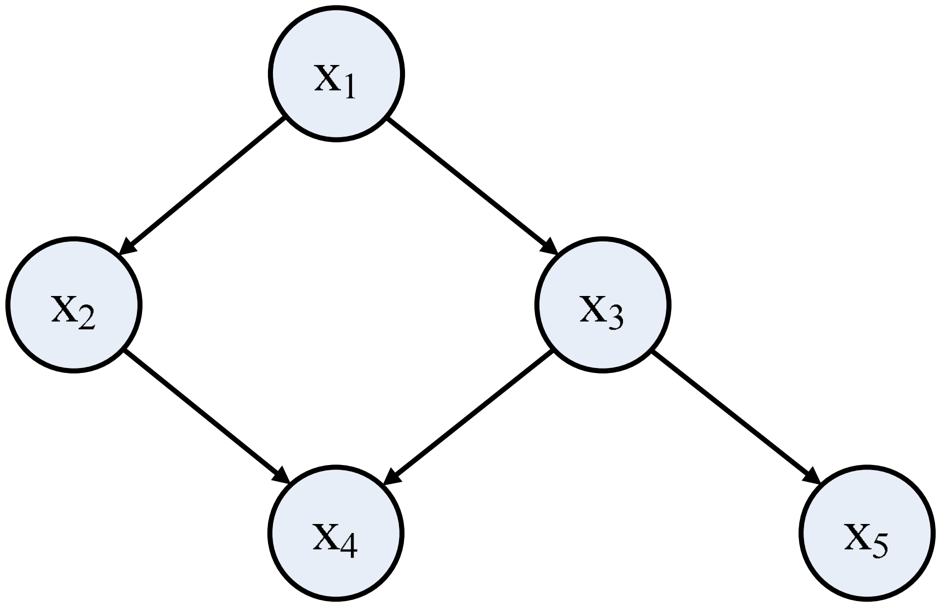
- Let
  - x_i be a node in a Bayesian network G
  - Γ_{x_i} be the set of parent nodes of x_i
- The influence of Γ_{x_i} on x_i can be specified by any set of functions $F_i(x_i, \Gamma_{x_i})$ that satisfy

$$\begin{aligned} \sum_{\forall x_i} F_i(x_i, \Gamma_{x_i}) &= 1 \\ 0 \leq F_i(x_i, \Gamma_{x_i}) &\leq 1 \end{aligned}$$

where x_i also refers to the states of the random variable associated to the node x_i

Bayesian Networks

- Figure below illustrates a Bayesian network for a joint probability distribution $P(x_1, x_2, x_3, x_4, x_5)$

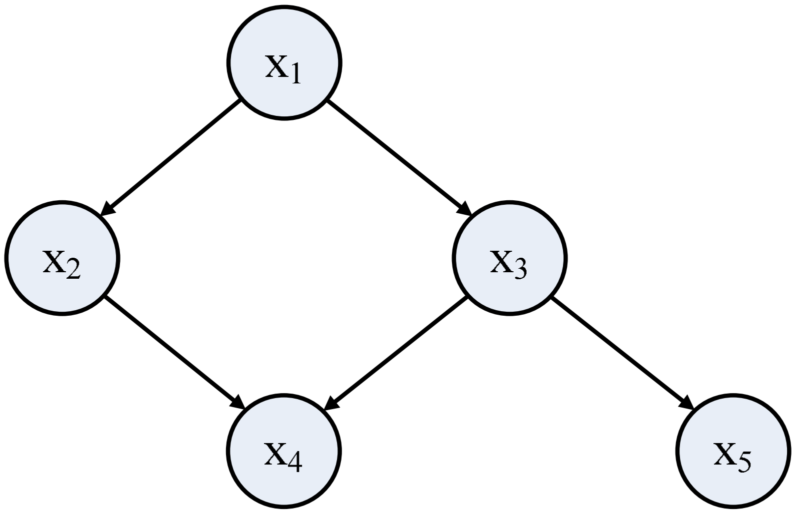


Bayesian Networks

- The dependencies declared in the network allow the natural expression of the joint probability distribution

$$P(x_1, x_2, x_3, x_4, x_5) = P(x_1)P(x_2|x_1)P(x_3|x_1)P(x_4|x_2, x_3)P(x_5|x_3)$$

- The probability $P(x_1)$ is called the **prior** probability for the network
- It can be used to model previous knowledge about the semantics of the application



Bayesian Network Models

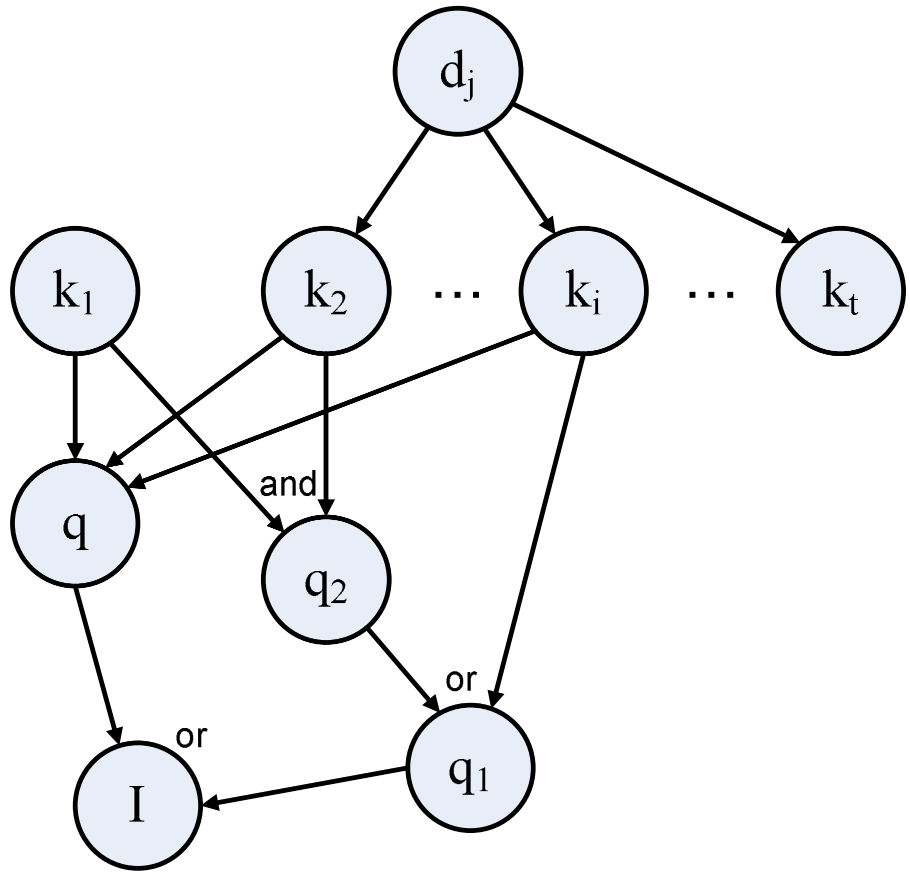
Inference Network Model

Inference Network Model

- An epistemological view of the information retrieval 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

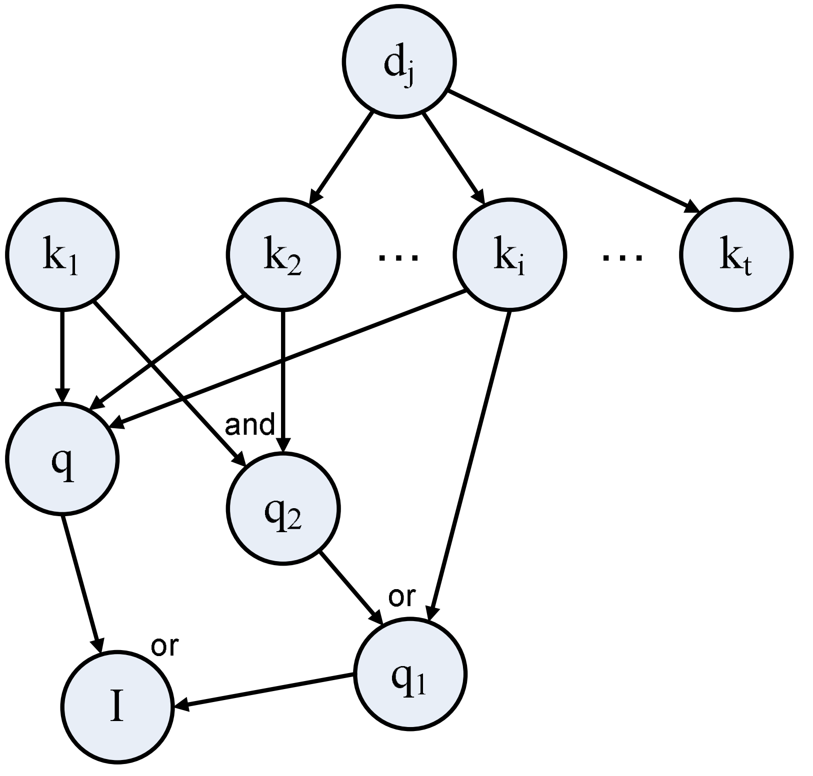
Inference Network Model

- Figure below illustrates an inference network for information retrieval

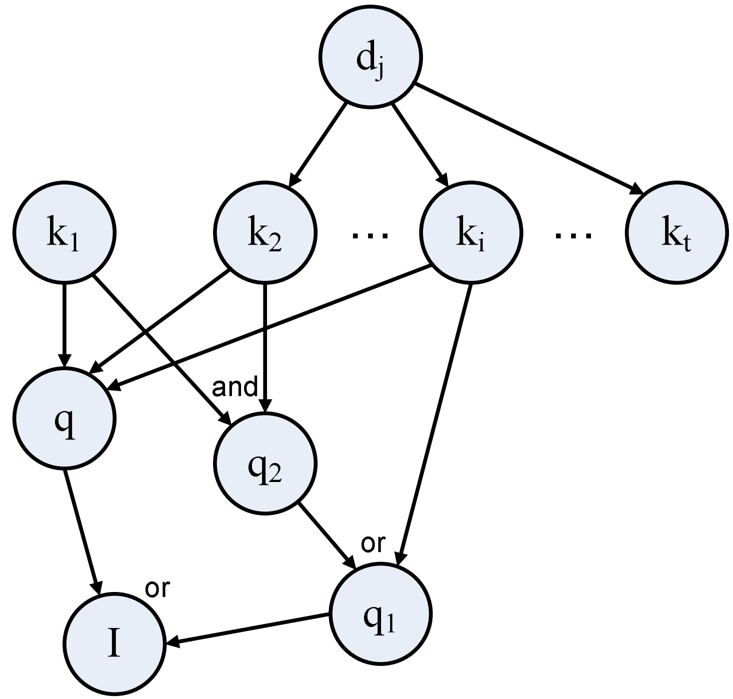


Inference Network Model

- Figure below illustrates an inference network for information retrieval
- Nodes of the network
  - documents (d_j)
  - index terms (k_i)
  - queries $(q, q_1, and\ q_2)$
  - user information need (I)
- 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
- $q_1 = (k_1 \wedge k_2) \vee k_i)$

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

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

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

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

  $\Rightarrow$ 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

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

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

$\Rightarrow$ 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 $I = q \vee q_1$

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

  $\Rightarrow$ 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)
    - $\vec{k}$ = $\{\vec{k}_1, \vec{k}_2, \ldots, \vec{k}_t\}$ a vector associated to each u such that $g_i(\vec{k}) = 1 \iff k_i \in u$
    - k_i a binary random variable associated with the index term k_i, ($k_i = 1 \iff g_i(\vec{k}) = 1 \iff k_i \in u$)

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

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

$\Rightarrow$ 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

Other Models

Introduction

- We now discuss information retrieval models that are not derived directly from the classic IR models
- These models include hypertext, Web ranking, structured text, and multimedia

Other Models

The Hypertext Model

The Hypertext Model

- Hypertexts provided the basis for the design of the hypertext markup language (HTML)
- One fundamental concept related to the task of writing down text is the notion of sequencing
- Written text is usually conceived to be read sequentially
  - When the reader fails to perceive such a structure and abide by it, they frequently are unable to capture the essence of the writer's message
- Sometimes, however, we are looking for information that cannot be easily captured through sequential reading
  - For instance, while glancing at a book about the history of the wars fought by man, we might be interested in the regional wars in Europe
- In such a situation, a different organization of the text is desired

The Hypertext Model

- A hypertext is a high level interactive navigational structure
- It allows browsing text non-sequentially on a computer screen
- It consists basically of nodes that are correlated by directed links in a graph structure
- With each node is associated a text region which might be a chapter in a book, a section in an article, or a Web page
- Two nodes A and B might be connected by a **directed link** l_{AB} which correlates the texts associated with these two nodes
- In this case, the reader might move to the node B while reading the text associated with node A

- In its most conventional form, a hypertext link l_{AB} is