Learning to Rank : Using Bayesian Networks

by

Parth Gupta

200911017

A thesis submitted in the partial fulfillment of the requirements for the degree of

Master of Technology

in

Information and Communication Technology

to

Dhirubhai Ambani Institute of Information and Communication Technology

Gandhinagar, India

April 2011

Declaration

This is to certify that

i) the thesis comprises my original work towards the degree of Master of Technology in Information and Communication Technology at DA-IICT and has not been submitted elsewhere for a degree.

ii) due acknowledgment has been made in the text to all other material used.

Signature of Student

Parth Gupta

Certificate

This is to certify that the thesis work entitled *Learning to Rank: Using Bayesian Networks* has been carried out by *Parth Gupta (200911017)* for the degree of Master of Technology in Information and Communication Technology at this Institute under my supervision.

Thesis Supervisors

Dr. Prasenjit Majumder

Dr. Suman Mitra

Acknowledgments

At first, I owe my deepest gratitude to Dr. Prasenjit Majumder, whose guidance and encouragement has given this work a sharp edge. He has been a great source of knowledge and inspiration for the love I possess for the subject. He has been always by my side on every shaky stones to share his experience.

I would also like to thank Dr. Suman Mitra heartily, whose co-guidance and discussions has always enabled me to explore new dimensions of this work. It would have been merely a dream to finish this work without his insightful motivation.

I would sincerely thank Dr. Minal Bhise and Dr. Jaideep Mulherkar, both from DA-IICT, who have been my thesis examiners, for their valuable discussions and suggestions that kept my zeal intact. They have been very supportive throughout this journey.

Besides them, I would like to extend my gratitude to authors of all the books and publications which enhanced me to understand the subject better and groomed my knowledge.

I am indebted to my parents, Dr. Alok Gupta and Mrs. Mithilesh Gupta, who have faith in me and that triggered fire in me to put my best into this work. They have been very supportive and liberal to allow my weird way of working till late night and getting up late in the morning. At the same time I owe my thank to my younger sister, Nirzari who keeps me cheerful.

I would like to thank my friends Sameer, Harish, Anil, Sandeep and Chintan for their discussions, trust and encouragement. Time spent in this institute with them would always be remembered. Last but not the least, a special mention for my dear friend Neha without whom my thesis might be very difficult.

Parth Gupta

Abstract

Ranking is one of the key components of an Information Retrieval system. Recently supervised learning is involved for learning the ranking function and is called 'Learning to Rank' collectively. In this study we present one approach to solve this problem. We intend to test this problem in different stochastic environment and hence we choose to use Bayesian Networks for machine learning. This work also involves experimentation results on standard learning to rank dataset 'Letor4.0'[6]. We call our approach as BayesNetRank. We compare the performance of BayesNetRank with another Support Vector Machine(SVM) based approach called RankSVM [5]. Performance analysis is also involved in the study to identify for which kind of queries, proposed system gives results on either extremes. Evaluation results are shown using two rank based evaluation metrics, Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG).

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

For the given, request and a collection of offerings if we can return the offerings in particular order then it is said they are ranked. Similarly in case of document retrieval for the given query and set of retrieved documents, if we can define the preferential order among them then it is called ranking. In a conventional Information Retrieval system, Text collection i.e. the documents or in case of web search the URLs are indexed. As a user query comes in, the query is matched with the documents indexed with some weighting score and the documents in the decreasing order of their scores are returned as a Rank-List.

Ranking is one of the core modules of Information Retrieval. For many years ranking has been unsupervised like the scores have been calculated on the fly and rank-list is generated. Recently experiments have shown that ranking can be performed with supervised learning which in turn improves the performance and at the same time help to incorporate many features. Many features are time dependent like some documents change rapidly, for example document recently created with buy call for a particular stock may be more important than that of five years back. Supervised Learning allows to incorporate such features for Ranking.

In Learning to Rank, ranking is seen as a model which is trained on the training

data, which comprises of queries and related documents with relevance judgments. When given a new query [i.e. test data], the model should produce correct rank list. Ranking can be a perfect ranking or optimal ranking [Fuhr et. al.. 1989]. In case of former all the relevant documents are ranked above the first irrelevant document and relevant documents can be ranked arbitrarily, while in latter comes the concept of degree of relevance, that all the documents should be ranked in the decreasing order of their degree of relevance.

1.1 Problem Definition

Ranking of Documents is a popular research area in Information Retrieval especially in Web Search where user expects the relevant documents at the top positions in the ranklist. In order to fulfill this the document for a particular query is characterized by a set of features which have influence on the relevance. Given these features for the set of documents the system should produce a ranklist which is perfect in nature i.e. all the relevant documents are ranked above the first irrelevant document. Technically the system should produce a real score for a document from its feature vector so that sorting these documents in decreasing order of the scores yields an perfect ranklist.

1.2 Thesis Objectives

Although there are many algorithms for learning to rank problem, there is no algorithm which incorporates Bayesian Networks. We intend to test performance of Bayesian Networks based algorithm for the document retrieval application of learning to rank. We also want to identify the set of queries where this approach performs really well. At the same time, this work also aims to describe as many existing algorithms in one survey as possible.

1.3 Organization of the Thesis

Chapter 2, tries to describe the task definition of Learning to Rank and more specifically the task of document retrieval using supervised learning. It also contains the details of Data used for the experimentation and features used for learning. In Chapter 3, we describe the evaluation metrics used for evaluating the performance of ranking algorithms. Chapter 4 tries to explain the related work. In general it tries to put many of the existing approaches for solving learning to rank task. There are plenty of algorithms available and being added in large number each year so it is out of scope of this work to include them all but we try to explain many of them. In chapter 6 we describe basic functioning of Bayesian Networks and In chapter 7 we explain specifically our approach. Chapter 8 contains experiments specifications and results, it also contains insightful analysis of the results. In chapter 9, we conclude the work.

Chapter 2

Task and Data

Learning to Rank in context of document retrieval is a task of building a ranking model. Task can be viewed as a machine learning approach where training data is given on which a ranking model is trained and then given new test case, model will correctly rank the documents. In training data some queries are given with list of documents and their relevance label, based on which the model is learnt. So given a new query with set of documents, model should predict the correct rank of the documents. In Letor, unlike conventional unsupervised ranking models, more features can be incorporated and automatically be learnt to combine these features for better rank prediction.

Letor 4.0 is the benchmark collection for research on learning to rank for information retrieval. It was released in July 2009 by Tao et al. [6] . It used Gov2 web page collection and two query sets of Million Query track of TREC 2007 and TREC 2008. Dataset is in the form of query-document pair feature vector for four ranking settings, supervised, semi-supervised, rank aggregation and listwise ranking. Dataset looks as shown below where each line is a query document pair.

2 qid:10032 1:0.056537 2:0.000000 3:0.666667 4:1.000000 ... 45:0.000000 46:0.076923 #docid = GX029-35-5894638

0 qid:10032 1:0.279152 2:0.000000 3:0.000000 4:0.000000 ... 45:0.250000 46:1.000000 #docid = GX030-77-6315042

0 qid:10032 1:0.130742 2:0.000000 3:0.333333 4:0.000000 ... 45:0.750000 46:1.000000 #docid = GX140-98-13566007

1 qid:10032 1:0.593640 2:1.000000 3:0.000000 4:0.000000 ... 45:0.500000 46:0.000000 #docid = GX256-43-0740276

here first column give the relevance label for that particular document for that query which can take values 0, 1 or 2 mapping to not relevant, partially relevant and definitely relevant. In this work we use term ground truth for actual relevance labels. Second column is query id and third column specifies the value of feature-1 for that particular query document pair. Such 46 features are given and then '#' denotes the comment which contains document id for that pair. Here all the feature values are normalized by the maximum value of that feature for that query. List of these 46 features is given below in Table 1.

In case of semi-supervised some of the unjudged documents are also included for which relevance label will be -1, while in rank aggregation one has to output the optimal list from given many input lists. In case of listwise setting ground truth permutation of the query instead of relevance label is given.

This benchmark collection also contains the baseline results of present learning to rank algorithms and hence provide a neutral medium to compare them for which it includes the evaluation script [6].

Index	Feature
1	TF(Term frequency) of body
2	TF of anchor
3	TF of title
4	TF of URL
5	TF of whole document
6	IDF(Inverse document frequency) of body
7	IDF of anchor
8	IDF of title
9	IDF of URL
10	IDF of whole document
11	TF*IDF of body
12	TF*IDF of anchor
13	TF*IDF of title
14	TF*IDF of URL
15	TF*IDF of whole document
16	DL(Document length) of body
17	DL of anchor
18	DL of title
19	DL of URL
20	DL of whole document
21	BM25 of body
22	LMIR.ABS of body
23	LMIR.DIR of body
24	LMIR.JM of body
25	BM25 of anchor
26	LMIR.ABS of anchor
27	LMIR.DIR of anchor
28	LMIR.JM of anchor
29	BM25 of title
30	LMIR.ABS of title
31	LMIR.DIR of title
32	LMIR.JM of title
33	BM25 of URL
34	LMIR.ABS of URL
35	LMIR.DIR of URL
36	LMIR.JM of URL
37	BM25 of whole document
38	LMIR.ABS of whole document
39	LMIR.DIR of whole document
40	LMIR.JM of whole document
41	PageRank
42	Inlink number
43	Outlink number
44	Number of slash in URL
45	Length of URL
46	Number of child page

Table 2.1: List of Features in Letor 4.0

Chapter 3

Evaluation Metrics

Evaluation is a very critical part of an information retrieval system to check the performance of the proposed algorithm. In this section we will focus on some of the rank based evaluation metrics.

1. **WTA (Winners take all) [25]**: For a particular query q, if the top ranked document is relevant then WTA(q)=1 otherwise WTA(q)=0, which is averaged over all the queries.Here the first document in the rank list play a crucial role, and hence its called winners take all.

2. **P@n : [24]** P@1 is precision at n^{th} position, precision is calculated by following formula

$$Precision = \frac{\{Retrieved\ Documents\} \bigcap \{Relevant\ Documents\}}{Retrieved\ Documents}$$

For example if P N P N N P P P N N is the rank list where P denotes positive (relevant) and N denotes negative (irrelevant) document then $P@10 = \frac{5}{10} = 0.5$ and $P@5 = \frac{2}{5} = 0.4$

3. **MRR (Mean Reciprocal Rank) [25]** : For query q, the rank position of the first relevant document is considered as R(q) and documents below that are nor considered. So the measure is 1/R(q) and averaged over all the queries.

4. **MAP (Mean Average Precision) [24]** : $P@n$ is a evaluation measure for unranked documents where no rank position based information is involved. MAP is an evaluation metric for ranked documents. In MAP we have binary relevance like either the document is relevant or irrelevant. For example, if P N P N P is the ranked list where P denotes the positive (Relevant) document and N denotes negative (irrelevant) document then, AP (avg. precision) becomes

$$\text{AP}=\frac{1}{3} * (\frac{1}{1} + \frac{2}{3} + \frac{3}{5}) \simeq 0.76$$

MAP is averaged over all the queries.

5. **NDCG (normalized discounted cumulative gain)**[10]:

NDCG is again an evaluation measure for ranked documents. Main advantage of NDCG over MAP is, its capability of handling multi-level graded relevance. Currently many evaluation forums have datasets where relevance is graded and not binary. for example relevance categories can be 4, 3, 2, 1 and 0 with 4 being extremely relevant and 0 being completely irrelevant. $NDCG@n$ can be calculated by the following formula.

$$\text{NDCG @ n} = Z_n * \sum_{j=1}^{n} \left[\frac{2^{c(j)} - 1}{\log (1 + j)} \right]$$

We describe it with the example of perfect ranking,

$$List-(3,3,2,2,1,1,1)- \qquad Relevance labels$$

$$Gain-(7,7,3,3,1,1,1)- \qquad 2^{r(j)}-1$$

$$position discount-(1,0.63,0.5,0.43,0.39,0.36,0.33)- \qquad \frac{1}{\log 1+j}$$

$$DCG-(7,18.11,24.11,...)- \qquad \sum_{j=1}^{n}\left[\frac{2^{c(j)}-1}{\log(1+j)}\right]$$

$$Normalizing factor-(1/7,1/18.11,1/24.11,...)- \qquad n_j$$

$$NDCG-(1,1,1,1,1,1,1)- \qquad For perfect ranking$$

Chapter 4

Related Work

For the given task there are broadly four approaches. We would brief them one by one in further sections and would discuss advantages and shortcomings of each of them.

i. Pointwise approach

ii. Pairwise approach

iii. Listwise approach

4.1 Pointwise approach:

Here regression or classification methods are directly applied for ranking task. In case of regression given a feature vector x, we need to map it to real value y' such that the loss $y - y'$ is minimum, where y' is computed relevance label and y is ground truth relevance label. One such approach is least squares retrieval function[1] which uses least square error (LSE) method to learn regression function.

Here least squares loss function will look like below where y is the actual relevance label and y' is estimated relevance.

$$\sum_{i=1}^{n}[y - y']^2$$

and rank function can be modeled as below,

$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_{46} x_{i46}$

$= \beta_0 + \sum_{j=1}^{46} \beta_j x_{ij}$

which in matrix form can be written as

$y = X\beta$

where,

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix} \qquad and \qquad X = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,k} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & a_{n,1} & a_{n,2} & \cdots & x_{n,k} \end{bmatrix}$$

Thus from training data least squares estimator $\hat{\beta}$ can be found by

$$\hat{\beta} = (X'X)^{-1} * X'y$$

Another approach used Discriminative models [2] over generative models. In latter case the class conditional probability $P(x \mid C)$ and prior probability P(C) is modeled and then Posterior probability $P(C \mid x)$ is estimated while in former case directly the posterior probabilities $P(C \mid x)$ is estimated where the C is the

class in which the input x is to be classified. Examples of discriminative models are Support Vector machines, Maximum Entropy model etc while Binary Retrieval Model(BIR), Language Models are types of generative models. Vapnik [26] argues that "one should solve the classification problem directly and never solve a more general problem class-conditional as an intermediate step." [2]

$$P\left(R \mid D, Q\right) = \frac{1}{Z\left(Q, D\right)} \exp(\textstyle\sum_{i=1}^{n} \lambda_i * x_i)$$

where λ_i's are estimated using gradient descent algorithm.

Although there are some disadvantages of regression based methods as they do not take information of pairs or total orders in learning. Also they consider relevance as absolute and independent. It is not the case especially in case of ranking in IR. Like term frequency of irrelevant document for popular query may be much higher than the term frequency of relevant document for a rare query. Also in Ranking the correct order is important and not the absolute score or value of f(x). So other approaches beyond regression methods are discussed in further sub-sections.

In pointwise approaches ordinal regression[21] is also used rather than simple regression or classification. In ordinal regression, given input as feature vector $x \epsilon R^t$ we get the ordered category $Y \epsilon (c1 \prec c2 \prec .... \prec ck)$ where c_i is an ordered category, in contrast to that of regression in which we get real value as output and in that of classification where we get unordered categories.

One of the approaches of applying ordinal regression in ranking task is Pranking with Ranking[3]. In Pranking we project $w^{T} \cdot x$ in to the ordered thresholds $b1 \prec b2 \prec \ \prec bk$ where w is the weight vector and bi is an ordered category. In the training phase we initialize w and b's and then the category is predicted which is compared to the actual

category, if the category is correct then next input is taken otherwise w and b's are modified in such a way that the predicted category is as closest to the actual category. Iteratively converged w and b's are found from the training data so that given a test data, correct categories are found.

In another approach by Shashua and Levin[4], they try to fit ordinal regression with help of SVM and propose two strategies, one is 'fixed margin' strategy and another is 'sum of margins' strategy. Figure below gives physical idea of two strategies. We use the same notations used in [4]. Here w and b_j are SVM parameters defining the hyperplane and ϵ_i^j is the slack variable for data points which are not correctly classified. Here $j = 1, \ldots, k-1$ where k is total number of classes and $i = 1, \ldots, i_j$ where i_j is number of training examples of class j.

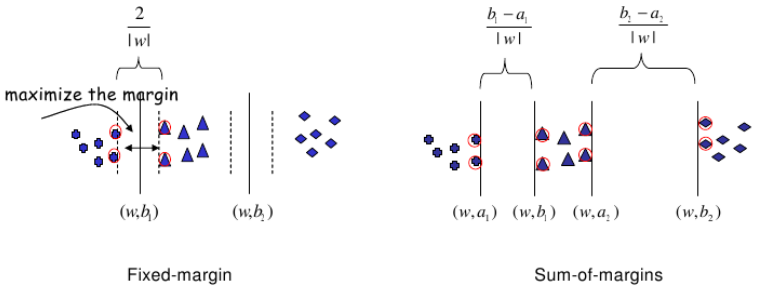


Figure 4.1: Fixed margin and Sum-of-margins strategy

Here Fixed margin strategy can be seen as an optimization problem shown below,

$$min_{w,b_j,\epsilon_i^j,\epsilon_i^{*j+1}} \frac{1}{2} w \cdot w + C \sum_i \sum_j \left(\epsilon_i^j + \epsilon_i^{*j+1}\right)$$

$$subject\ to$$

$$w \cdot x_i^j - b_j \leq -1 + \epsilon_i^j$$

$$w \cdot x_i^{j+1} - b_j \geq 1 - \epsilon_i^{*j+1}$$

$$\epsilon_i^j \geq 0, \epsilon_i^{*j+1} \geq 0$$

And sum-of-margin strategy can be seen as below,

$$min_{w,a_j,b_j} \sum_{j=1}^{k-1} + C \sum_i \sum_j \left(\epsilon_i^j + \epsilon_i^{*j+1}\right)$$

$$subject\ to$$

$$a_j \leq b_j,$$

$$b_j \leq a_{j+1}, j = 1, 2, \ldots, k-2,$$

$$w \cdot x_i^j \leq \epsilon_i^j,$$

$$b_j - \epsilon_i^{*j+1} \leq w \cdot x_i^{j+1}$$

$$w \cdot w \leq 1, \epsilon_i^j \geq 0, \epsilon_i^{*j+1} \geq 0$$

Pointwise approaches are slightly different from conventional approaches and again can not handle ground truth of pairwise preferences or partial/total order information.

Nallapati et. al.[2] also used discriminative models using Support Vector Machine for classification of documents in two classes, relevant and irrelevant and ranking

function was considered as how far is the document from the classification hyperplane and on which side. The document close to the hyperplane on the relevant side is less relevant than the one which is far from the hyperplane.

4.2 Pairwise Approach:

In Pairwise approaches we take input as two feature vectors $x_i, x_j \epsilon R^t$ where $x_i \succ x_j$ and give output as $Y \epsilon \{+1, -1\}$, where '+1' denotes that $x_i \succ x_j$ is correct order and '-1' otherwise. Ultimately we derive the pairwise preferences from the training data. Ex. if given data is like (x1,2), (x2,1), (x3,0) where 0,1 and 2 are the relevance labels then pairwise information will be (x1,x2,+1), (x2,x1,-1), (x1,x3,+1), (x3,x1,-1), (x2,x3,+1) and (x3,x2,-1).

One such approach is Ranking SVM proposed by Herbrich et al.[5]. Here as described above we create our new training dataset S' in pairwise form where we will have entries like (x1-x2,+1) where x1-x2 is a new vector and +1 is label. In case of Conventional Support Vector Machine binary classifier, we try to find weight vector w and offset b, which can be found by solving the optimization problem shown below,

$$minimize \frac{1}{2} \parallel w \parallel^2 + C \sum_i \zeta_i$$
$$subject\ to\ y_i \left(w^T \cdot x_i + b\right) \geq 1 - \zeta_i, \zeta_i \geq 1$$

In case of RankSVM the SVM model is as shown below,

$$minimize \frac{1}{2} \parallel w \parallel^2 + C \sum_i \zeta_i$$
$$subject\ to\ y_i \left\langle w, x_i^1 - x_i^2 \right\rangle \geq 1 - \zeta_i, \zeta_i \geq 1, i = 1, 2, \ldots, l$$

which further can be written as below when $\lambda = \frac{1}{2C}$,

$$min\ \left[1 - y_i \left\langle w, x_i^1 - x_i^2 \right\rangle\right] + \lambda \parallel w \parallel^2$$

here first term is so called Hinge Loss and second term is regularizer. Tao et. Al investigate that there are still concerns with biased nature of RankSVM where hyperplane is more inclined towards queries with many relevant documents compared to those with fewer relevant documents. Also it treats all the pairs with equal weight instead it should give more weight to definitely relevant-and-not relevant document pair.

In pairwise approaches, algorithms take advantage of ordered pairs of documents but still they do not handle any information of total order which is handled by listwise approaches.

4.3 Listwise Approaches

In listwise approach a ranklist itself is an entity in learning in contrast to pairwise approaches where two document pairs of a query are learning entities and pointwise approaches where single query-document pairs are learning entities. Cao et. al. defined listwise loss function of two types [17]. The loss function

$$\sum_{i=1}^{N} L(y_i, z_i)$$

where z_i is the generated ranklist for the query i and y_i is the ground truth ranklist for the query i, ground truth ranklist is the actual ranklist for the query given its documents. Two probability models are proposed to calculate the loss function. One is Permutation Probability and the second one is Top k Probability. For n objects, π

is a permutation of n objects and $\pi(j)$ denotes the object at j^{th} position. $s_{\pi(j)}$ is the score of the j^{th} object in permutationπ. $\phi(.)$ is the strictly positive and increasing function. Then Permutation probability can be defined as below

$$P_s(\pi) = \prod_{j=1}^{n} \frac{\phi(s_{\pi(j)})}{\sum_{k=j}^{n} \phi(s_{\pi(k)})}$$

Here, the permutation probability for all the possible permutations for n objects is calculated, which is of $O(n!)$ and may be sometimes computationally intractable. So the provision of Top k Probability is included. Where calculating the Permutation Probability is limited to a subset of all the permutations. This subset comprises those permutations which have top k elements exactly as $(j_1, j_2, .., j_k)$ which is denoted as $\varsigma(j_1, j_2, ..., j_k)$. and Top k Probability is defined as below

$$P_s(\varsigma(j_1, j_2, ..., j_k)) = \prod_{t=1}^{k} \frac{\phi(s_{j_t})}{\sum_{l=t}^{n} \phi(s_{j_t})}$$

After getting this probability the loss function is defined by Cross Entropy as a metric as shown below. Here Nurel Network is is used for Learning the model. That model will assign a score to the document.

$$L(y_i, z_i) = -\sum_{\forall g \in \varsigma_k} P_{y_i}(g) \log(P_{z_i}(g))$$

Chapter 5

Bayesian Networks

Bayesian Networks have two major components: Structure Learning and Parameter Learning. In structure learning, structure of the Network that best fits the data is chosen. After determining the structure Probability of Particular node given its parents is determined. There are two broad categories of structure learning: Constraint-Based Approach and Search-and-Score approach. In Constraint-Based Approach we start with fully connected graph and edges are removed if certain in-dependencies are found in the data. CIS is such type of algorithm. While in Search-and-Score we search through the space of possible Directed Acyclic Graphs (DAG) and return the best one found. K2 and Hill Climber are Search-and-Score based algorithms. We would like to make it clear that here we learn Bayesian Networks with full observability where there are no missing values.

5.1 Structure Learning

In Structure Learning, we have to define basically a set of edges for the network. Hence we define a structure of the network as a set of edges E and selecting the best

network can be seen as

$$P(E|X) = \frac{P(X|E) * P(E)}{P(X)}$$

Here $P(E)$ is prior and can be set as $\frac{1}{|A|}$where $|A|$ is total number of networks possible and $P(X)$ is constant so can be omitted. $P(X|E)$ is the likelihood term which can be defined as below

$$P(X|E) = \prod_{j=1}^{d} \prod_{l=1}^{q_j} \prod_{i=1}^{k_j} \Theta_{jil}^{n_k(x_j^{(i)}|\pi_j^{(l)})}$$

Here d is the total number of variables, in our case its total number of nodes. k_j is the total number of states that a variable x_j can take, in case of binary variable x_j value of k_j will be 2. q_j is total number of possible parent configurations for variable x_j. Function Θ_{jil} is defined below. n_k is the function which says number of times $X_j = x_j^{(i)}$ and $\Pi_j = \pi_j^{(l)}$ configuration occurs together divided by number of times $\Pi_j = \pi_j^{(l)}$ occurs.

$$\Theta_{jil} = P(X_j = x_j^{(i)}|\Pi_j = \pi_j^{(l)})$$

This likelihood function is called Cooper-Herskovitz Likelihood function [22]. This score is calculated for all the possible networks and the network which has the maximum score is selected. But calculating the score for all possible networks is not possible because possible number of DAGs(Directed Acyclic Graphs) grows super exponentially with number of nodes. e.g. total number of possible of DAGs for N=10 nodes is 4.2 x 10^{18}. So there are different algorithms available, which tries to select the subset of possible DAGs to compute the score and select the Network from them which maximizes the score. We describe some of the algorithms below.

5.1.1 K2 Algorithm [22]

This algorithm takes the nodes in order and considers all the nodes before the particular node as a possible candidate to be considered as its parent. It has a parameter to define the max number of parents for a node. It starts with a node and calculate the score of the network by adding a node from its possible parent set i.e. all the nodes before it. After this the node is added as a parent which maximizes the score. This process is continued until maximum number of parent bound is reached or addition doesn't improve the score.

5.1.2 Hill Climber(HC) Algorithm [23]

This algorithm does not require nodes in specific order. It has four possible operations: Add an Arc, Remove an Arc, Reverse an Arc or Don't do anything. Here we start with an empty network and perform any of the mentioned operations and select the one which retains the maximum score.

5.1.3 Conditional Independence Search(CIS) Algorithm [27]

In this algorithm we start with fully connected network and start removing edges which by removing improves the score of the network. The implicit idea behind removing edge and checking the score is to see that respective nodes are conditional independent or not.

5.2 Parameter Learning

Now from structure learning algorithms we have selected the best fitting network and we know explicitly the parent set of each node. Maximum Likelihood Estimation

is used for calculating the class probabilities where the likelihood function is same Cooper-Herskovitz likelihood function as described in the previous section.

Chapter 6

Our Approach : BayesNetRank

We want to solve the ranking of documents problem by considering it as a multi-class classification problem. We have tried to solve the classification problem using Bayesian Networks. Ranking problem can be solved by classification but is quite different than classification, because if all the instances are classified one class below then the classification error is 100% while ranking is correct. So as suggested by Burges et. al.[11] There has to be ranking function on top of the classification which links the Ranking and Classification.

On of the major limitation in using Bayesian Networks with Learning to Rank data is, Bayesian Networks work only with discrete integer data while Letor data is continuous. So we first discretize the data. We have thought of a very straight forward scheme to discretize the continuous data which we call KDisc. Here all the features have values between and including 0 and 1. So we define a parameter interval. We take the value of interval=0.025 so we get 40 classes between 0 and 1.

Alg. 1 KDisc discretization scheme

1: repeat step 2 to step 7 for $i = 0$ *to* n, where n=no. of features
2: take all the possible values of feature i and get the histogram
3: count the frequency for each class and average value of that feature
4: **for** $j = 1 \rightarrow TotalClasses - 1$ **do**
5: **if** $(count[j] > count[j-1]$ && $count[j] > count[j+1]$ && $count[j] > average)$ **then**
6: consider it as local maxima and add it in initial means list
7: **else if** there is no local maxima **then**
8: consider one class and add 0.5 in initial means list
9: **else**
10: continue
11: **end if**
12: add 0 and 1 in the initial means list
13: run the K-Means clustering to find the actual means of the clusters
14: replace the value of the feature by the index of the cluster it belongs
15: **end for**

We also use discretization scheme proposed by Fayyad and Irani [12]. Which implicitly uses class entropy to decide the cut-point for the discretization.

We have tried three different learning modules, K2, Hill Climber (HC) and Conditional Independence Search (CIS). Former two are of type Search-and-Score and last one is of type Constraint-Based. Bayesian Network based learning will give us probabilities of the instance belonging to a particular class, in our case three classes of Relevance 0,1,2. While our final aim being ranking and not classification we use a ranking function on top of the probabilities provided by the Bayesian Network. This ranking function is called expected Relevance and proposed by Burges et. el. [11]

$$Score(X_i) = \sum_i i * P(R = i)$$

This ranking function gives a real score to the instance and in the end all the instances are sorted according to their scores in decreasing order to generate the ranklists. We call this ranking method collectively as BayesNetRank.

We have also tried to fit in the loss function in bayesian networks structure learning. So far we have used the Cooper-Herskovitz likelihood function to score a network and we choose the network which maximizes it. We use the following two loss functions to see the performance of our approach. Here Y is the set of actual Relevance Score and Y' is the set of predicted Relevance Score. In loss function L_1 we try to minimize the Mean-Squared-Error loss between the two while in loss function L_2 we choose the network structure which minimizes the Entropy.

$$L_1(Y, Y') = \sum_{i}^{N} (y_i - {y'}_i)^2$$

$$L_2(Y, Y') = -\sum_{i}^{N} \log(P(R = y)$$

Chapter 7

Experiments and Results

We have tried our discretization approach and MDL based discretization to learn the bayesian network. For learning purpose, we have tested K2, Hill Climber and Conditional Independence Search algorithms. Results of linear combination of the three are also obtained. A novel combination scheme 'Zscore' proposed by [13] is also used to combine the ranklists generated by the three learning algorithms. Calculation of the scores in both: Linear Combination and ZScore is described below, where d_i is i^{th} document which score is to be computed. $LC(d_i)$ basically computes the average score of the document given by three different structures: K2, HC and CIS. There can be a weighted average with unequal weights but experiments showed that equal weights are more robust in terms of MAP. In $ZScore(d_i)$ α_j is the weight of the j^{th} ranklist, $Score_j(d_i)$ is score given to i^{th} document by j^{th} ranklist, $Mean_j$, Min_j and $StdDev_j$ are the mean, minimum and standard deviation of j^{th} ranklist respectively. In our case α_j is also 1 with the same reason given for LC. We report the results on MQ2007 and MQ2008 DataSet of Letor 4.0 which is basically Queryset of Million Query Track of TREC 2007 and 2008 respectively. Reported results are averaged over 5 cross-folds of dataset. Results suggest that MDL based discretization improves the performance than the very handy KDisc discretization scheme. One of the possible reasons is, in MDL based discretization scheme, the classes are selected based on the

Dataset: MQ2007	**MAP**		**NDCG**	
Algorithm	**KDisc**	**MDL Based**	**KDisc**	**MDL Based**
RankSVM	**0.4645**		**0.4966**	
K2	0.3849	0.4333	0.4109	0.4695
HC	0.4081	0.4189	0.4385	0.4507
CIS	0.4193	0.4400	0.4515	0.4773
LC(K2,HC,CIS)	0.4117	**0.4544**	0.4441	**0.4919**
ZScore(K2,HC,CIS)	0.4224	0.4500	0.4543	0.4877

Table 7.1: Results for MQ2007 Dataset

Dataset: MQ2008	**MAP**		**NDCG**	
Algorithm	**KDisc**	**MDL Based**	**KDisc**	**MDL Based**
RankSVM	**0.4695**		**0.4832**	
K2	0.4283	0.4537	0.4343	0.4580
HC	0.4224	0.4264	0.4272	0.4365
CIS	0.4555	0.4672	0.4671	0.4786
LC(K2,HC,CIS)	0.4566	**0.4683**	0.4660	**0.4796**
ZScore(K2,HC,CIS)	0.4475	0.4622	0.4565	0.4665

Table 7.2: Results for MQ2008 Dataset

measured classification entropy while KDisc does not involve such classification based knowledge in deciding the discretization levels. Now onwards we would report results based on the MDL based discretization scheme.

$$LC(d_i) = \frac{K2Score(d_i) + HCScore(d_i) + CIScore(d_i)}{3}$$

$$Zscore(d_i) = \sum_{j}^{3} \alpha_j \left[\left(\frac{Score_j(d_i) - Mean_j}{Std.Dev._j} \right) + \left(\frac{Mean_j - Min_j}{Std.Dev._j} \right) \right]$$

Both the datasets in Letor 4.0 contains 5 cross-fold validation. *Figure*7.1 and *Figure*7.2 below show the Fold wise MAP results of all the algorithms.

We also compute the results after incorporating loss functions described in previous chapter: Mean-Squared-Error based loss function and Entropy based loss function. Following table contains the results of K2 and Hill Climber algorithms which tries

Algorithm	Loss/Score Function	MAP	NDCG
K2	Cooper-Herskovitz	0.4333	0.4694
K2	MSE	**0.4477**	**0.4850**
K2	Entropy	0.4370	0.4767
HC	Cooper-Herskovitz	0.4189	0.4505
HC	MSE	0.4397	0.4765
HC	Entropy	**0.4400**	**0.4773**

Table 7.3: Results for MQ2007 Dataset with Different Score/Loss Function

Algorithm	Loss/Score Function	MAP	NDCG
K2	Cooper-Herskovitz	0.4537	0.4580
K2	MSE	**0.4658**	**0.4774**
K2	Entropy	0.4580	0.4689
HC	Cooper-Herskovitz	0.4264	0.4365
HC	MSE	0.4584	0.4690
HC	Entropy	**0.4672**	**0.4786**

Table 7.4: Results for MQ2008 Dataset with Different Score/Loss Function

to select the network which minimizes the loss function. From the table we can see modifying the score type in structure learning to that for pointwise loss function suitable for Learning to Rank certainly improves the performance. We observe that K2 algorithm works better with MSE based loss function and HC algorithm works better with Entropy based loss function. Both : K2 and HC algorithms outperforms with described loss functions compared to that with Cooper-Herskovitz likelihood score function.

We also do an analysis on querywise performance of the BayesNetRank Algorithm where basically we want to identify the set of queries for which the algorithm performs well. We also intend to do failure analysis where we want to check the set of queries for which our system left behind RankSVM. Following table contains the data in terms of number of queries for which our system performed well compared to RankSVM as well as number of queries for which RankSVM performed well and for remainder queries both the system performed comparable to each other. In order to compare the performance we consider *NDCG Score* as a measure, if difference between NDCG

Dataset	**Total Queries**	**# of queries for BayesNetRank performed well**	**# of queries for RankSVM performed well**	**# of queries for which both performed comparable**
MQ2007	1692	434 (25.65%)	464	794
MQ2008	784	134 (17.09%)	157	493

Table 7.5: Querywise performance statistics

Score of both the systems for a particular query is greater than 0.05 than we consider system with higher NDCG Score is performing well. Similarly if the difference between the NDCG Score of both the systems for a particular query is less than or equal to 0.05 then we say both of them performs comparable for that query.

As Letor4.0 data and most of the Learning to Rank data are statistical and does not give any sense of semantics, it becomes very difficult to identify the nature of the queries e.g. the query is focused or open-ended or navigational. We have also done an analysis by incorporating the actual query sets of Million Query Track of TREC 2007 and TREC 2008. This allowed us to have a look at the nature of the queries at the same time performance of our system for those queries. We categorize the queries in three categories: Focused Queries - Where user is very precise for the information need , Open-Ended Queries - Where user give abstract keywords and seeks information in very abstract and Navigational Queries - Where user has some virtual information in mind and want to reach to that page using the query words [24]. In 7.6 we give some examples of such queries from the same query sets. We have done the analysis of set of top 100 queries for both the ranking methods: BayesNetRank and RankSVM. For example set S_1 contains those queries for which BayesNetRank performs well with NDCG score difference of 0.05 with that of RankSVM while set S_2 contains those queries for which RankSVM performs well compared to BayesNetRank. After that we did manual classification of queries of these sets in aforementioned categories.

From the tables 7.7 and 7.8 we notice an interesting fact that our system usually performs well for focused queries and RankSVM performs better in case of open-ended

Type of Query	Examples
Focused Queries	• what is the purpose of icd codes • water well treatments for salt removal • guidelines for handwashing vs. alcohol based rubs
Open-Ended Queries	• peregrine falcon • about the white house • girls health
Navigational Queries	• farmington savings bank • albert gore jr • nicholas donofrio ibm

Table 7.6: Types of Queries

Type of Query	BayesNetRank	RankSVM
Focused	**52**	38
Open-Ended	38	**50**
Navigational	10	12

Table 7.7: Distribution of top 100 better performing queries for MQ2007 Dataset

Type of Query	BayesNetRank	RankSVM
Focused	**66**	**56**
Open-Ended	24	33
Navigational	9	11

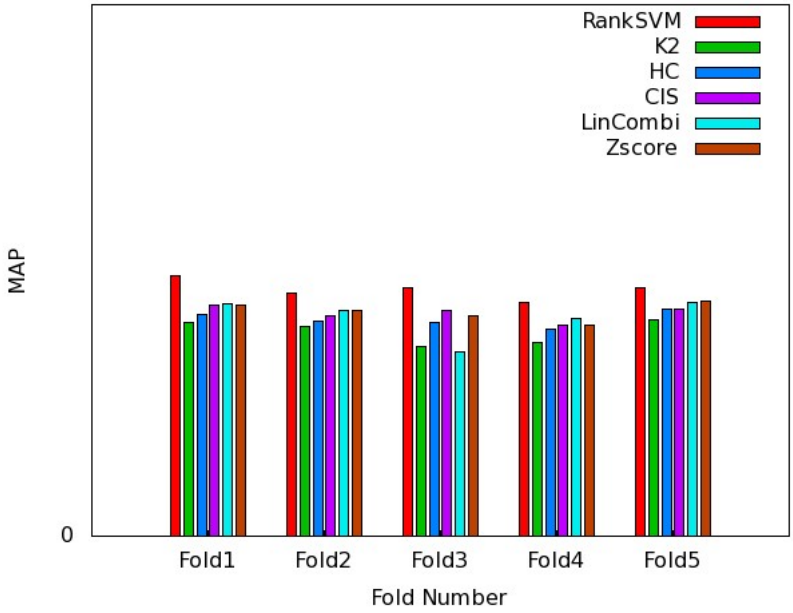
Table 7.8: Distribution of top 100 better performing queries for MQ2008 Dataset

Dataset	RankSVM	BayesNetRank(LC)	p-value
MQ2007	0.4966	0.4919	0.10981
MQ2008	0.4832	0.4796	0.23996

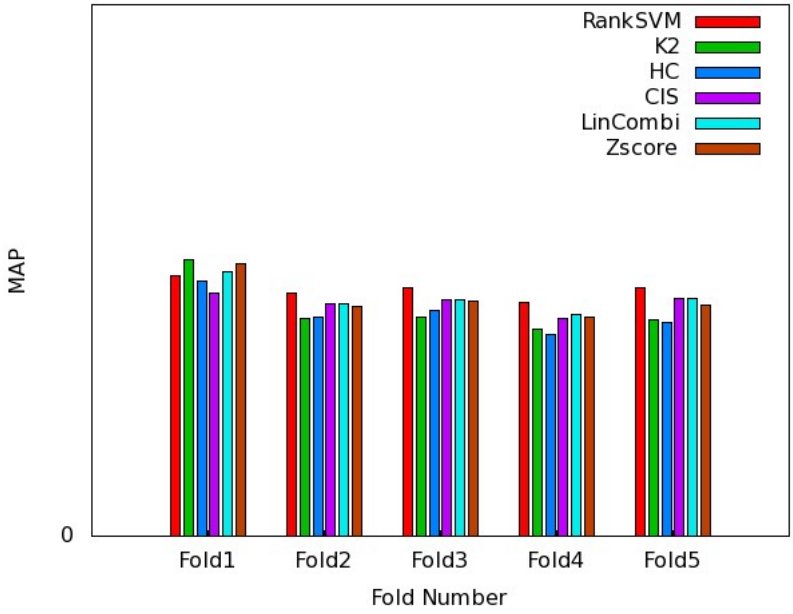
Table 7.9: Significance TTest Results for RankSVM and BayesNetRank(LC)

queries. For Navigational queries both the systems perform comparable. Generally Focused queries are those where user seeks very specific information and so the relevant documents are comparatively less in initial retrieval which is given as input to the Learning to Rank system. On the other hand open-ended queries have generally large number of relevant documents in the initial retrieval.

From the tables 7.1 & 7.2 and figures 7 & 7 it is visible that in some folds of MQ2007 and in all folds of MQ2008 Our approach performs comparable to the RankSVM. From the results it is also noticeable that combination of ranklists generated from different learning algorithms certainly improves ranking in comparison to individuals. We consider NDCG as a measure to evaluate ranking function because of its ability to handle multi-level relevance labels. In contrast to Letor 4.0 dataset, MAP considers only binary relevance judgements. To compute MAP for Letor4.0 dataset all the relevance labels with 0 and 1 value are considered as 0 and 2 is considered as 1. To support our statement of comparable performance we present the p-value of the Significance TTest for MQ2007 and MQ2008 DataSet between our best performing run and RankSVM in Table 7.9. Generally if p-values are greater than 0.05 then we consider that both the results are coming from the same distribution.

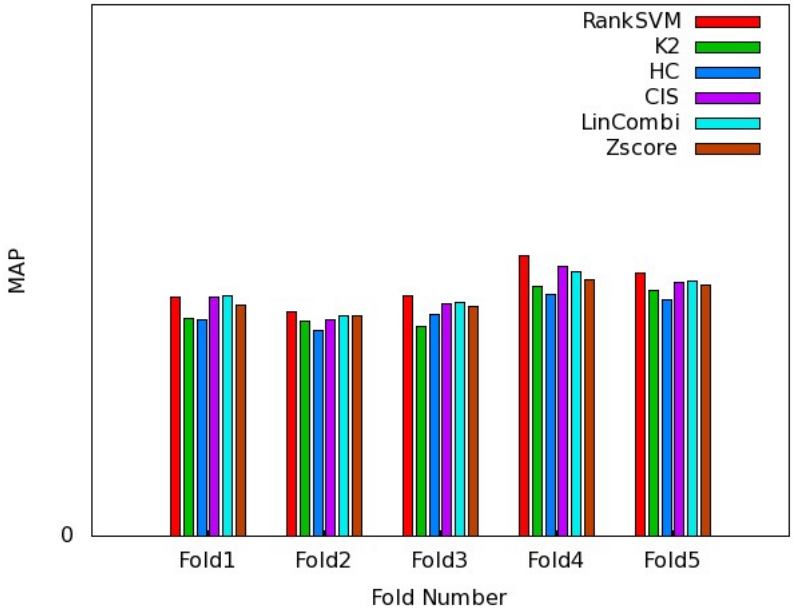


(a) KDisc

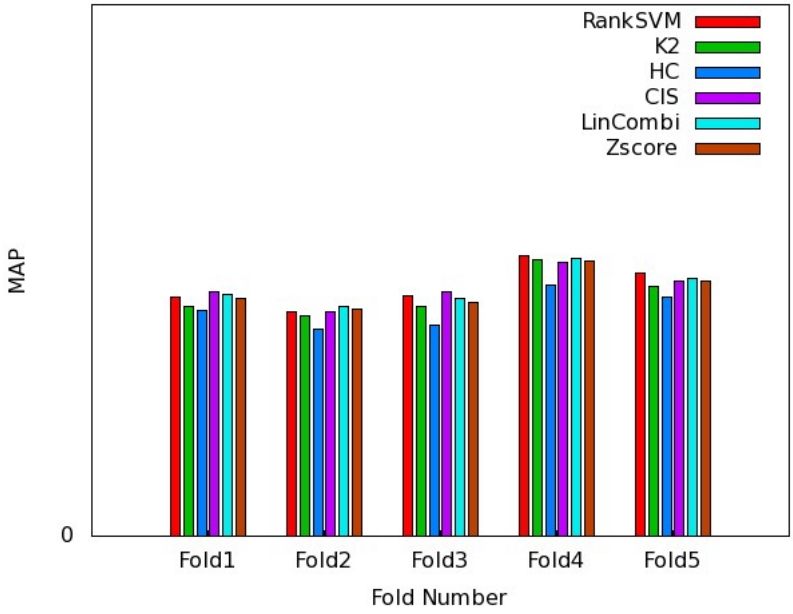


(b) MDL Based

Figure 7.1: Foldwise Results for MQ2007



(a) KDisc



(b) MDL Based

Figure 7.2: Foldwise Results for MQ2008

Chapter 8

Conclusion

In this work, we intended to test the behaviour of learning to rank problem in different stochastic environment like Bayesian Networks. There are many approaches to address the problem using Support Vector Machines, Nueral Networks, Regression, Ordinal Regression etc. To the best of our knowledge, for document retrieval there does not exist a learning to rank algorithm based on Bayesian Networks. We also succeed to fit the pointwise loss functions defined for learning to rank in our approach to select suitable structure of Bayesian Network for the specified task. The obtained results are encouraging and suggest that the system is comparable to RankSVM.

In this work, we do querywise analysis of set of queries for which system either performs really well or performs very poor. Experiments and analysis suggest that the BayesNetRank performs well in case of focused queries and RankSVM performs well for open-ended queries.

Bayesian Networks are very much fragile to over-fitting the data compared to other existing state-of-the-art methods. Bayesian Networks rely on probability of patterns in the input. Currently the learning to rank algorithms consider all the queries equal and hence it becomes very essential to properly normalize the data. The training data

currently available for the training is generated from the manually created relevance judgements which contains number of unjudged documents which may be relevant but marked as irrelevant and hence fragile for learning and testing.

REFERENCES

[1] N. Fuhr. Optimum polynomial retrieval functions based on probability ranking principle. *ACM Transactions on Information Systems (ToIS)*,1989.

[2] Nallapati, R. Discriminative models for information retrieval. *Proceedings of SIGIR 2004* (pp. 64-71).

[3] Crammer K. & Singer Y. Pranking with ranking. *Proceedings of NIPS 2001.*

[4] Shashua A. & Levin A. Taxonomy of large margin principle algorithms for ordinal regression problems. *Proceedings of NIPS 2002.*

[5] R. Herbrich, T. Graepel, and K. Obermayer. Large margin Rank Boundaries for OrdinalRegression. *Advances in Large Margin Classifiers*, pages 115-132,2000.

[6] T-Y Liu, J. Xu, T. Qin, H. Li. LETOR: A benchmark collection for research on learning to rank for information retrieval. *In SIGIR 07 Workshop on learning to rank for information retrieval*, 2007.

[7] T. Qin, T.-Y. Liu, W. Lai, X.-D. Zhang, D.-S.Wang, and H. Li. Ranking with multiple hyperplanes. *In SIGIR 2007*, pages 279-286, 2007.

[8] D. Cossock and T. Zhang. Subset ranking using regression. *In COLT 2006*, pages 605-619, 2006.

[9] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. *In SIGIR 2006*, pages 186-193, 2006.

[10] Jarvelin K., Lekalainen J., Cumulative gain based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4),422-446, 2002.

[11] P. Li, C. Burges, and Q. Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. *In NIPS 2007*, 2007.

[12] Usama M. Fayyad, Keki B. Irani: Multi-interval discretization of continuous valued attributes for classification learning. In: *Thirteenth International Joint Conference on Articial Intelligence*, 1022-1027, 1993.

[13] Savoy, Jacques : Data Fusion for Effective European Monolingual Information Retrieval. *Multilingual Information Access for Text, Speech and Images. LNCS*, pages 233-244, 2005.

[14] T. Koski, J. Noble : Bayesian Networks : An Introduction , John Wiely & Sons, Ltd. 2009.

[15] Neapolitan, R. Learning Bayesian Networks, Prentice Hall Series in Artificial Intelligence, 2004.

[16] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; *SIGKDD Explorations*, Volume 11, Issue 1.

[17] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. *In ICML 2007*, pages 129-136, 2007.

[18] D. Sculley. Combined Regression and Ranking. *In KDD 2010*, pages 979-987, July 25-28,2010, Washington, DC, USA.

[19] Z. Zheng, K. Chen, G. Sun, and H. Zha. A regression framework for learning ranking functions using relative relevance judgments. *In SIGIR 2007*, pages 287-294, 2007.

[20] T. Joachims. Optimizing search engines using clickthrough data. *In KDD 2002*, pages 133-142, 2002.

[21] W.W. Cohen, R.E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research (JAIR)*, 10:243270, 1999.

[22] Cooper, G.F. and Herskovitz, E. (1992) A Bayesian method for the induction of probabilistic networks from data. Machine Learning, 9, 309347.

[23] Tsamardinos, I., Brown, L.E. and Aliferis, C.F. (2006) The max-min hill-Climbing Bayesian network structure learning algorithm, *Machine Learning*, 65, 3178.

[24] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze (2007), Introduction to Information Retrieval, Cambridge university press, cambridge.

[25] Hang Li, Learning to Rank, Tutorial, *ACL-IJCNLP*, 2009, Singapore.

[26] Vapnik V., Statistical learning theory, John Wiley, New-York, 1998.

[27] Kevin Murphy, A Brief Introduction to Graphical Models and Bayesian Networks, *http://www.cs.ubc.ca/ murphyk/Bayes/bnintro.html*

[28] Million Query Track, Text REtrieval Conference, Homepage : *ciir.cs.umass.edu/research/million/*

[29] Text REtrieval Conference, Homepage: *trec.nist.gov/*