Semantic Searching and Ranking of Documents using Hybrid Learning System and WordNet

Pooja Arora
Research Scholar, Banasthali Vidyapith
Assistant Professor, MCA Department AKGEC
Ghaziabad, India

Prof. Om Vikas
Senior Member, IEEE
India

***Abstract*— Semantic searching seeks to improve search accuracy of the search engine by understanding searcher's intent and the contextual meaning of the terms present in the query to retrieve more relevant results. To find out the semantic similarity between the query terms, WordNet is used as the underlying reference database. Various approaches of Learning to Rank are compared. A new hybrid learning system is introduced which combines learning using Neural Network and Support Vector Machine. As the size of the training set highly affects the performance of the Neural Network, we have used Support Vector Machine to reduce the size of the data set by extracting support vectors that are critical for the learning. The data set containing support vectors is then used for learning a ranking function using Neural Network. The proposed system is compared with RankNet. The experimental results demonstrated very promising performance improvements. For experiments, we have used English-Hindi parallel corpus, Gyannidhi from CDAC. F-measure and Average Interpolated Precision are used for evaluation.**

Keywords- Learning to Rank; English-Hindi Parallel Corpus; Hybrid Learning; Support Vector Machine (SVM); Neural Network (NN); Semantic Searching; WordNet; Search Engine.

I. INTRODUCTION

Information retrieval is the method of searching documents, and information within documents and metadata about documents in databases and on the World Wide Web. The main idea is to locate terms that the user specify in his query. Many documents contain the desired semantic information, even though they do not contain the user specified query terms. For retrieving those documents semantic searching is required. So, semantic similarity of the terms must also be considered while calculating the score of the particular document. For that, firstly the user query is expanded by replacing all the query terms by their synonyms and then searching is performed according to the changed query and score of all the retrieved relevant documents is calculated using the ranking function. We have used WordNet for query expansion. Finally, all the retrieved documents are ranked according to their relevance score.

First we discuss learning to rank and compare its various approaches. Then we have proposed our approach of learning a ranking function using Support Vector Machine (SVM) and Neural Network (NN). [3]

II. LEARNING TO RANK

The problem of ranking has recently gained much attention in information retrieval. The task of Learning to Rank has emerged as an active and growing area of research both in information retrieval and machine learning. The goal is to design and apply methods to automatically learn a function from training data, such that the function can sort objects (e.g. documents) according to their degrees of relevance, preference, or importance as defined in a specific application. In information retrieval, Learning to Rank is used to generate an effective ranking function that is used to rank documents according to their relevance score.

The learning process, formalized as follows, consists of two steps: **training and test**. Given a query collection Q = { q_1,…….,q_m } and a document collection , D = { d_1, ... , d_n } , the training corpus is created as a set of query-document pairs , each (q_i , d_j) ϵ Q × D, upon which a relevance judgment indicating the relationship between q_i and d_j is assigned by a labeler . The relevance judgment can be a score e.q. sim(q_i , d_j) specifying the degree of relevance between q_i and d_j .[18]

The inputs to the learning algorithm comprise training instances, their feature vectors and the corresponding relevance judgments. The output is a ranking function R_f, where $R_f(q_i,d_j)$ is supposed to give the true relevance score for q_i and d_j. During the training process, the learning algorithm attempts to learn a ranking function such that a performance measure (e.g. Mean Average Interpolated Precision (MAIP), F-measure, etc.) with respect to the output relevance judgment can be optimized.

In the test phase, the learned ranking function is applied to determine the relevance between each document d_i in D and a new query q_{m+1}. The learning is greatly affected by various factors such as performance measure used for evaluation, the form of training instance, etc.

Various Approaches of Learning to Rank

Learning to Rank approach is classified into Pointwise approach, Pairwise approach and Listwise approach depending upon the type of instance used for learning. Their comparison is shown in Table 1.

Pointwise Approach: The Pointwise approach solves the problem of ranking by means of regression or classification

on single documents. It takes features of a single document with respect to query as input : $\varphi(q_k,d_i^k)$. In the training phase, learner learns to classify each instance of the document as relevant or irrelevant. In test phase, model assigns a unique score to each instance according to its relevance to queries. After operation, it produces ordered categories as output: $\{R_f(q_k,d_1^k)$, $R_f(q_k,d_2^k)$ . . ., $R_f(q_k,d_n^k)\}$ where n is the number of documents retrieved and $R_f(q_k,d_n^k)$ is the score of the n^{th} document. Various algorithms were proposed using this approach. For instance, Crammer & Singer propose a ranker Prank based on the Perceptron which maps a feature vector x to the real with a learned weight vector *w* such that the output of the mapping function is just *w*.x. Prank regards a query/document pair as an instance for the input, and each instance is corresponding to a rank level.[9] Harrington has proposed a simple but very effective extension of Prank, which approximates finding the Bayes point by averaging over Prank models.[6] RankProp is also a neural net ranking model proposed by Caruana.[2] RankProp alternates between two phases: an MSE regression on the current target values, and an adjustment of the target values themselves to reflect the current ranking given by the net.

Pairwise Approach: The Pairwise approach transforms ranking to classification on document pairs. It takes document pairs as input: $\{\varphi(q_k,d_i^k),\varphi(q_k,d_j^k)\}$ such that one document is more relevant than another. On these instances system performs pairwise preference learning. The label pair $\{label_i^k, label_j^k)$ is used to indicate the order of i^{th} and j^{th} document. $label_i^k < label_j^k$ means i^{th} document is more relevant than j^{th} document. After operation, model gives a binary value to an indicator variable y_{ij} of +1 or -1, depending upon the order of the documents in the instance pair. $y_{ij} = +1$ if $R_f(q_k,d_i^k) \geq R_f(q_k,d_j^k)$ and $y_{ij} = -1$ if $R_f(q_k,d_i^k) < R_f(q_k,d_j^k)$. Many algorithms were trained using this approach. For instance, Herbrich cast the problem of learning to rank as ordinal regression – learning the mapping of an input vector to a member of an ordered set of numerical ranks. [7]

They model ranks as intervals on the real line, and consider loss functions that depend on pairs of examples and their target ranks. RankBoost is another ranking algorithm that is trained on pairs. [21] In this algorithm, results are given using decision stumps as the weak learners. It attempts to solve preference learning problem directly, rather than solving an ordinal regression problem.

Dekel has provided a very general framework for ranking using directed graphs, where an arc from A to B means that A is to be ranked higher than B.[4] Joachims proposed RankSVM algorithm, which uses Support Vector machine for optimizing search performance using click-through data. It aims to minimize the number of discordant pairs, which is similar to RankBoost, and to maximize the margin of pair.[8] RankNet is another algorithm that employs relative entropy as a loss function and gradient descent as an algorithm to train a neural network model for document retrieval.[1]

Listwise Approach: In Listwise approach, there is no classification of instances or instance pairs. It tackles the ranking problem directly by optimizing the ordering of the whole list. It treats the list of documents associated with the same query as learning instance to obtain rank and query level information. It takes document collection with respect to query as input : $\{ \varphi(q_k,d_1^k),\varphi(q_k,d_2^k), . . , \varphi(q_k,d_n^k)\}$ and produces permutation of these documents as output : Π_n^k where $\varphi(q_k,d_i^k)$ is the feature vector of the i^{th} document w.r.t. k^{th} query. ListNet was one of the first listwise method.

It this, the listwise loss function is defined as cross entropy between two parameterized probability distributions of permutations; one is obtained from the predicted result and the other is from the ground truth.[22] RankCosine was another method. In this, the listwise loss function is defined on the basis of cosine similarity between two score vectors from the predicted result and the ground truth.[16] ListMLE is another listwise method that employ the likelihood loss as the surrogate loss function. They maximize the sum of the likelihood function with respect to all the training queries.[5]

TABLE I. COMPARISON OF DIFFERENT APPROACHES

	Pointwise Approach	**Pairwise Approach**	**Listwise Approach**
Number of instances	Equal to number of training elements	Equal to half of the training elements	Takes whole list as a training instance
Implementation Complexity	O(n)	$O(n^2)$	More complex
Training Time	More	Less	Less
Characteristic	More suitable to ordinal regression	More suitable to learning to rank	More suitable to learning to rank
Technique	Transforms ranking to regression, classification or ordinal regression	Transforms ranking to pairwise classification	Simply represents learning to rank problem
Document Dependence	No dependence between training documents is considered	Document dependence is considered	More dependence between documents is considered
Existing Theories	Easy to use existing theories and algorithms	Easy to use existing theories and algorithms	New theory needed
Flexibility	Flexible to ensure people expected precision	More flexible than Pointwise approach	Less flexible

III. OUR APPROACH

Machine learning provides wide range of algorithms for learning a ranking function. Some of the algorithms are more suitable than others. Two potential learning techniques used are Neural Network (NN) and Support Vector Machine (SVM). NNs are partly inspired on biological learning system and are the most effective learning methods currently known. SVM is based on margin maximization and is the most elegant of all kernel-learning methods. Each has its own advantages and disadvantages. The comparison of both the learning approaches is mentioned in Table 2. In our approach, we have used SVM for pruning to documents that are more critical for learning a ranking function and thus reduced the size of the learning data set. And then NN is used to train the system.

TABLE II. COMPARISON OF SVM AND NN LEARNING

SVM	**NN**
Deterministic Algorithm	Non-Deterministic Algorithm
SVM takes into account learning examples as well as structural behavior. It achieves better generalization due to structural risk management.	Use data empirical risk minimization which stops training once learning error is within a specified margin. This leads to non-optimal model & the solution is often plagued by local minimum problem.
An objective function is convex, so that unlike in the cases of many NN models, any local minimum of an SVM model is also a global minimum.	Suffers from local convergence problem
SVM is comparatively faster than NN. SVM solution is sparse; it only involves the support vectors.	NN suffer from long learning times, which become worse as the volume of data grows

A. Architecture of Hybrid Learning System

Our hybrid learning system works on user feedback as shown in figure 1. Initially, we gave query to the Information Retrieval System which is then internally expanded by the synonyms using WordNet. Initially, term frequency is used to calculate the score of the document and ranked accordingly. Then we select relevant documents out of the result set returned and generate training set for learning the ranking function. This training set is then used by SVM to extract support vectors which are critical for learning. These support vectors are then used to learn a ranking function using back-propagation algorithm using NN. Then the whole result set is ranked again according to the new ranking function. This ranking is compared with the previous ranking using Kendall's Rank correlation coefficient. If the correlation coefficient value is less than the threshold value then the new training set is generated based on user feedback and the whole process is repeated again. We have taken threshold value as 0.6.

B. Implementation

Outliers or meaningless vectors are identified by SVM and can therefore be easily eliminated. Support vectors that are critical in classification are extracted from SVM. Then these support vectors along with their corresponding actual output labels are used to train NN to perform final ranking.

In this way NN training examples are pruned using SVM so that the training examples that are closest to decision boundary are left for final training of the learner.

C. Pruning to Documents Using Support Vector Machine (SVM)

SVM can be characterized as an algorithm that affects a nonlinear mapping of input vectors into a higher dimensional feature space. It involves a dual formulation of governing equations and constraints.

SVM will construct a hyperplane in a high-dimensional features space as the decision surface between positive and negative patterns. SVM formulation approximates SRM (Structural Risk Minimization) principle by maximizing the margin of separation. Basic SVM is linear but it can also be used for non-linear data by using kernel function to first indirectly map non-linear data into linear feature space.

We have used RankSVM algorithm which is based on pairwise approach for extracting critical data patterns. [15], [20]

a) Margin Maximization

Let,

d_i^k represents i^{th} document retrieved from the document collection w.r.t query q_k.

$label_i^k$ and $label_j^k$ are the integer labels of documents d_i^k and d_j^k respectively w.r.t query q_k.

r be the number of features of the document used for learning a ranking function.

A feature vector of r features $\varphi(q_k,d_i^k) = (x_{i1}^k, x_{i2}^k, \ldots x_{is}^k, \ldots, x_{ir}^k)$ represented by x_i^k is created from each query-document pair (q_k,d_i^k) k= 1,2,. .,m and i=1,2,. .,n , where x_{is}^k is the value of the s^{th} feature of the i^{th} document w.r.t query q_k.

Then, mathematically ranking function R_f is represented by a weight vector *w* of size r that satisfies

$\forall \{(d_i^k,d_j^k): label_i^k < label_j^k \in I\}$ w.r.t q_k : $R_f(d_i^k) > R_f(d_j^k) \Leftrightarrow w.\varphi(q_k, d_i^k) > w.\varphi(q_k, d_j^k)$

SVM has to learn the values of the parameter *w* on a training sample. Here, learning is done pairwise i.e. it takes

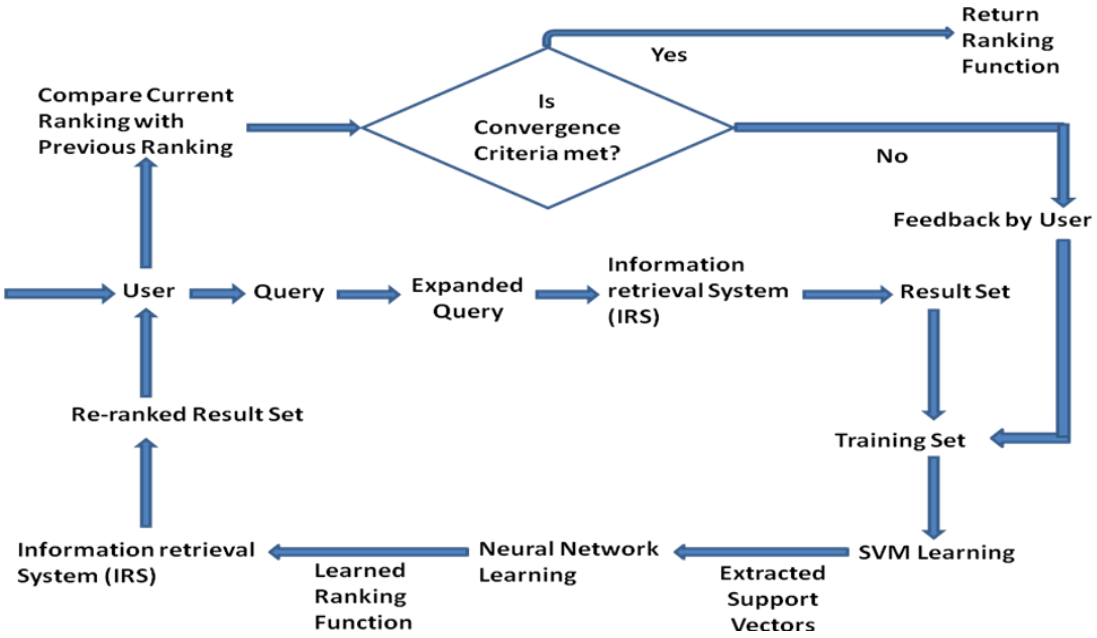

Figure 1. Architecture of Hybrid Learning System

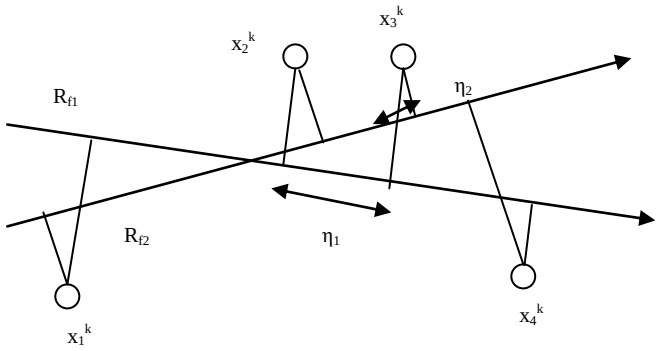

Figure 2. Ranking function as projecting data points

document pairs as input for training such that one document is more relevant than another. [7],[8],[22] A weight vector *w* is adjusted by a learning algorithm. The training set is denoted as triplet $T=\{(q_k,d_j^k), \varphi(q_k,d_j^k), label_i^k\}$ k=1,2,. . .,m and i=1,2,. .,n where, $label_i^k$ is the label of the i^{th} document w.r.t query k. $label_i^k < label_j^k$ means i^{th} document is more relevant than j^{th} document. Herein strict ordering is assumed.

The ranking model is a real valued function of features:

$R_f(q,d) = w.\varphi(q,d)$ where *w* denotes a weight vector

To rank all the documents retrieved w.r.t. query q_k, ranking model $R_f(q_k,d_j^k)$ gives score to each document d_j^k as their degree of relevance with respect to query q_k and sort the documents based on that score.

Let, two ranking vectors R_{f1} and R_{f2} and four feature vectors x_1^k ,x_2^k ,x_3^k and x_4^k of the four documents to be ranked. Ranking function can be viewed as the projecting data points of the four documents onto the separating hyperplane as shown in the figure 2. Let, η_1 and η_2 be the distance between the closest points on the hyperplane. From a geometric point of view, calculating the value of the parameters *w* means looking for a hyperplane called ranking vector that maximizes the distance between various data points of the documents to be ranked according to some criteria. The criterion is based on margin maximization i.e. to maximize the distance between closest points. The distance between these points is calculated as

$$\frac{w \| x_i^k - x_j^k \|}{\| w \|}$$

Given a training set of instance-label pairs, the SVM requires the solution of the following optimization problem:

$$\min_{w,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \sum_{j=1}^{n} \xi_{ij}$$

subject to : $\forall\{(x_i^k, x_j^k) : y_{ij}(w^T(x_i^k - x_j^k)) \geq 1 - \xi_{ij}$,

$\forall(i, j) : \xi_{ij} \geq 0.$

where, i,j = 1,2,3 . . .n. and y_{ij} is an indicator variable to indicate order of i^{th} and j^{th} document and C = "capacity" is a tuning parameter for controlling the generalization ability of an SVM.

In the first part of the equation, it maximizes the margin or distance which is inversely proportional to $\|w\|$ by minimizing $\|w\|^2/2$ and second term is the sum of in-sample ranking errors ξ_{ij} times the parameter C. In most cases, ξ_{ij} =0, that means i^{th} and j^{th} documents are ranked correctly. Thus, SVM maximizes the margin width while minimizing errors. Thus, this problem is quadratic i.e. convex.

C weights in-sample errors and thus controls the generalization ability of an SVM. Higher is C, higher is the weight given to in-sample errors, and lower is the generalization of the learning model. By choosing a low C, the risk of overfitting an SVM on the training sample is reduced. So, the constant C is the soft margin parameter that controls the trade-off between the margin size and training error.

Using the method of Lagrange multipliers, we can obtain the dual formulation which is expressed in terms of variables α_i:

$$\underset{\alpha}{\text{maximize}} \sum_{ij} \alpha_{ij} - \frac{1}{2} \sum_{ij} \sum_{uv} \alpha_{ij}\, \alpha_{uv} y_{ij} y_{uv} ((x_i - x_j).(x_u - x_v))$$

subject to : $C \geq 0$ α_{ij} is a coefficient for a pairwise difference vectors $(x_i - x_j)$.[19]

We perform a non-linear mapping of the feature vector x onto a high-dimensional space that is hidden from the inputs or the outputs using Kernel function. As SVM solution depends only on the dot product of $(x_i - x_j)$ and $(x_u - x_v)$, operations in high dimensional space $\Phi(x)$ do not have to be performed explicitly if we find a function $K(x_i, x_j)$ such that $K(x_i, x_j)$ is equal to dot product of x_i and x_j. This function is called Kernel function.

Using Kernel function, the dual formulation can be expressed as:

$$\underset{\alpha}{\text{maximize}} \sum_{ij} \alpha_{ij} - \frac{1}{2} \sum_{ij} \sum_{uv} \alpha_{ij}\, \alpha_{uv} y_{ij} y_{uv} K(x_i - x_j, x_u - x_v)$$

subject to : $0 \leq \alpha \leq C$

where K(.) is a kernel function.

We have used the same optimization problem, except the dot product of $(x_i - x_j)$ and $(x_u - x_v)$ is replaced by the kernel $K(x_i - x_j, x_u - x_v)$. We have used linear Kernel function $K(x_i, x_j) = (x_i^T x_j)^2$ for mapping feature vector x onto a high-dimensional space.

The values of α's are obtained between 0 and C. Data points with non-zero α's are called support vectors which are critical for learning a ranking function. So, all those data point pairs with non-zero α's are extracted and then used to train NN.

b) Format of Training and Test file

The format of training file and test file is same. Details of all the documents are stored in a file in which each row represents one document in a LINE in the following format:

LINE —> L qid : QID F:FV F:FV . . . . F:FV # COMMENT

L —> <float>	// label
QID —> <integer>	// query identifier
F —> <integer>	// feature
FV—> <float>	// feature value
COMMENT —> <string>	// comment as line identifier

Each line contains the target label of the document, query identifier and each of the feature/value pairs are separated by a space character. Feature/value pairs must be ordered by increasing feature number. Features with value zero can be skipped. The target label defines the order of the examples for each query and is used to generate pairwise preference constraints. Two examples are considered for a pairwise preference constraint only if the value of "qid" is same. [14]

D. Learning a Ranking function Using NN

RankNet algorithm is used for learning a ranking function.[1] This algorithm is based on error back-propagation of the NN. Here also, the learning is done pairwise. All the relevant document w.r.t. a query in the training set are paired in such a way that odd numbered document is more relevant than the next even numbered document. E.g. Document positioned at row 1 is more relevant than document at row 2, and document at row 3 is more relevant than document at row 4 and so on. So, if we have 100 documents in the training set it will give 50 such pairs. Thus, in each pair first document is more relevant than second document.

Let the training set is denoted as triplet T= $\{(q_k, d_i^k)$, $(\varphi(q_k, d_i^k), \varphi(q_k, d_j^k))$, $(label_i^k, label_j^k)\}$ k=1,2,. . .,m and i,j=1,2,. .,n where, $label_i^k$ is the label of the i^{th} document w.r.t query k. $label_i^k < label_j^k$ means i^{th} document is more relevant than j^{th} document. Herein strict ordering is assumed.

In this algorithm, learning consists of two passes through the different layers of the network: a forward pass and a backward pass. In the forward pass, learning algorithm is given a set of pairs of documents (d_i^k, d_j^k), together with their labels indicating the order of the documents, and its effect propagates through the network layer by layer. A set of outputs is produced as the actual response of the network. During the backward pass, the synaptic weights are all adjusted in accordance with an error-correction rule to

minimize the cost function. The cross entropy cost function is used as a loss function for training.[13]

The target probability P_{ij}^* of d_i being ranked higher than d_j is 1 for each document pair (d_i^k, d_j^k) in the training set as we have arranged all the document pairs accordingly. A logistic function used for mapping the output of the NN as a ranking function to a probability is as follows:

$$P_{ij} = \frac{e^{o_{ij}}}{1+e^{o_{ij}}}$$

where $o_{ij} = y_i - y_j$ where y_i is the output of the network for the i^{th} document and $P_{ij} = P(d_i > d_j)$ i.e. the probability that the i^{th} document is more relevant than j^{th} document. The cross entropy is adapted as a loss function for training, which can be represented as:

$CE_{ij} = CE(o_{ij}) = -P_{ij}^* \log P_{ij} - (1-P_{ij}^*) \log (1-P_{ij})$

$= -P_{ij}^* o_{ij} + \log(1+ e^{o_{ij}})$

where CE_{ij} is the cross entropy loss of the pair (d_i^k, d_j^k), P_{ij}^* is the desired probability, and P_{ij} is the probability modeled by the NN.

Following steps are followed :

1) i^{th} document and j^{th} document pair along with their labels are presented to the input layer of the network. These inputs are propagated through the network until they reach the output units. This forward pass produces the predicted output in the form of the probability P_{ij}.
2) Because back propagation is a supervised learning algorithm, the desired output in the form of the desired probability P_{ij}^* is given as part of the training vector. The actual network output is then subtracted from the desired output and an error signal in the form of cross entropy loss is calculated.
3) This error signal is then the basis for the back propagation step, whereby the errors are passed back through the neural network by computing the contribution of each hidden processing unit and deriving the corresponding adjustment needed to produce the correct output. The connection weights are then adjusted and the neural network has just "learned" from an experience.

After the network has learned all the new documents can easily be scored by presenting them to the input layer and ranked them according to that score.

E. Features used for Learning

We have used the following features for learning a ranking function:

Term Frequency(TF): $\sum_{q_i \in Q \cap D} \log(c(q_i, D))$ where $c(q_i,D)$ is the frequency of the i^{th} term of the query in document D

Inverse Document Frequency(IDF):

$\sum_{q_i \in Q \cap D} \log(idf(q_i))$ where $idf(q_i)$ is the inverse document frequency of the i^{th} term of the query

Normalized Cumulative Frequency(NCF):

$\sum_{q_i \in Q \cap D} (\frac{log|C|}{c(q_i,C)})$

Normalized Term Frequency weighted by IDF(NTFI):

$\sum_{i=1}^{n} \log(1 + \frac{c(q_i,D)}{|D|} \times idf(q_i)$

F. Efficacy measure: Kendall's tau τ rank correlation coefficient

Let R^* be the optimal ranking of the documents in which all the documents are ranked according to user's preference. A new generated ranking function R_f is typically evaluated by how closely its ordering approximates optimal ordering of R^*. [11]

The ranking order of a set of training instances is optimized according to Kendall's Tau, τ :

$$\tau(R^*,R_f) = \frac{P-Q}{P+Q} = 1 - \frac{2Q}{\binom{n}{2}}$$

where,

P is the number of concordant pairs (two documents are ordered correctly)

Q is the number of discordant pairs (two documents are ordered incorrectly)

On a finite domain D of n documents, the sum of P (concordant pairs) and Q (discordant pairs) is $\binom{n}{2}$

G. Semantic searching using WordNet

The query keyword used for retrieval of documents is the most significant but not always sufficient. We have used synonyms of the keyword also for searching the corpus. We have created a WordNet containing synonyms of various words and thus stored semantic relations between various words. We have used those semantic relations for expanding the given query and to improve the retrieval effectiveness. Red-Black tree data structure is used to store various words along with their synonyms as shown in figure 3. Each node of the tree contains word along with its various synonyms. Each query term is expanded by the various synonyms before searching in the corpus.[17]

For example, If the query contains the term "estimate" then after expansion it will be replaced by terms "estimate, idea, approximate, appraisal".

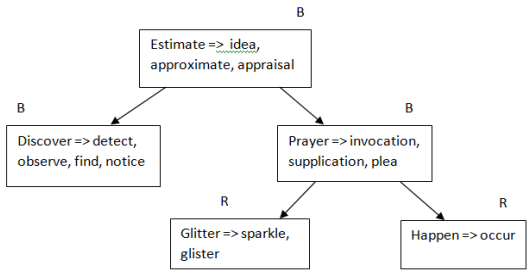

Figure 3. Sample Red-Black tree of Synonyms

IV. EVALUATION MEASURES

A. Recall

It is the measure of the ability of a system to present all relevant items.[3]

$$R = \frac{\text{Number of relevant documents retrieved}}{\text{Number of relevant documents in collection}}$$

B. Precision

It is the measure of the ability of the system to present only relevant items.

$$P = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of documents retrieved}}$$

To evaluate ranked lists, precision can be plotted against recall after each retrieved document. To facilitate computing average performance over set of queries – each with different number of relevant documents – precision values for individual query are interpolated to a set of standard recall levels (0 to 1 in increments of .2). The standard rule to interpolate precision at standard recall level i is to use the maximum precision obtained for the query for any actual recall level greater than or equal to i.

Mathematically, Interpolated precision $P_{interpolated}$ at certain standard recall level i is defined as the highest precision found for any recall level $i' \geq i$:

$$P_{interpolated(i)} = \max P(i') \quad i' \geq i$$

C. Average Interpolated Precision (AIP)

It is the average of the interpolated precision at each standard recall point value for all queries together.

$$AIP_i(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|} P_{interpolated(i)}(j) \quad i= 0.0, 0.1, 0.2, \ldots, 1.0$$

where Q is the set of queries $P_{interpolated(i)}$ is the interpolated precision at i^{th} recall value level,

$AIP_i(Q)$ is the average of the interpolated precision at i^{th} recall level for all the queries in set Q.

D. F-Measure (F)

It is harmonic mean of precision and recall. It is a single measure that trades precision versus recall.

$$F = \frac{2*\text{Precision}*\text{Recall}}{(\text{Precision} + \text{Recall})}$$

E. Mean Average Interpolated Precision (MAIP)

It is the mean of all the average interpolated precisions calculated at all standard recall points.

$$MAIP = \frac{\sum AIP_i(Q)}{\text{No\_recall\_points}} \qquad i = 0.0, 0.1, 0.2, \ldots, 1.0$$

where $AIP_i(Q)$ is the average of the interpolated precision at i^{th} recall level for all the queries in set Q

No_recall_points are the number of standard recall points used.

V. EXPERIMENTAL EVALUATION

We have implemented Information Retrieval System in Java. For experiments, we have created English-Hindi test collection of about 50 documents extracted from gyannidhi corpus from CDAC. Details of English-Hindi parallel corpus are mentioned in table 3. We have also generated training set triplet of about 20 queries that are stored along with their relevant documents for training. Sample queries for both Hindi and English are shown in table 4. All the data are encoded in Unicode text. F-measure and AIP are used for evaluation.

TABLE III. PARALLEL CORPUS DETAIL

	English	Hindi
No. of Documents	50	50
No. of index terms	7896	12687
No. of queries	20	20
Average No. of terms/doc	152	236

TABLE IV. SAMPLE QUERIES

qid	English	Hindi
1	estimate the approximate age	आयु का अनुमान
2	pleasant sound	मधुर ध्वनि
3	hazardous journeys	जोखिम-भरी यात्राएं

VI. TABLE V. AIP VALUES FOR RANKNET AND OUR HYBRID LEARNER

Recall	AIP	
	RankNet	Hybrid Learner
0.2	0.3	0.4
0.4	0.45	0.54
0.56	0.52	0.65
0.7	0.63	0.7
0.83	0.74	0.8

TABLE VI. F-MEASURE VALUES FOR RANKNET AND HYBRID LEARNER

Recall	F-Measure	
	RankNet	Hybrid Learner
0.2	0.36	0.37
0.4	0.49	0.54
0.6	0.61	0.65
0.8	0.77	0.79

VII. RESULTS AND DISCUSSION

We now describe our experimental analysis of our learning system. We evaluated ranking functions for 20 queries by calculating AIP and F-Measure values on different recall values ranging from 0 to 1. Comparison of these values for RankNet and our hybrid is learner shown in table 4 and table 5 respectively. Their comparison is also shown graphically in figure 4 and figure 5 respectively. MAIP of retrieved documents using RankNet and our hybrid learner is 0.528 and 0.618 respectively. There is significant improvement in F-measure and AIP values using proposed hybrid learner.

VIII. CONCLUSION

Learning to Rank is applied to automatically learn a ranking function from the training data. In this paper, a new hybrid learner is introduced based on NN and SVM that gives better performance than learning using NN alone. SVM is used to extract support vectors from the training data which are critical for learning a ranking function. These support vectors contribute better in learning a ranking function. We have used semantic searching to improve search accuracy by using synonyms of the query term to retrieve more relevant results. For experiments, we have constructed English-Hindi IR data collection from Gyannidhi parallel corpus. This system works language independently. F-measure and AIP are used for evaluation. F-measure and AIP values of the retrieved documents are improved using the proposed hybrid learner.

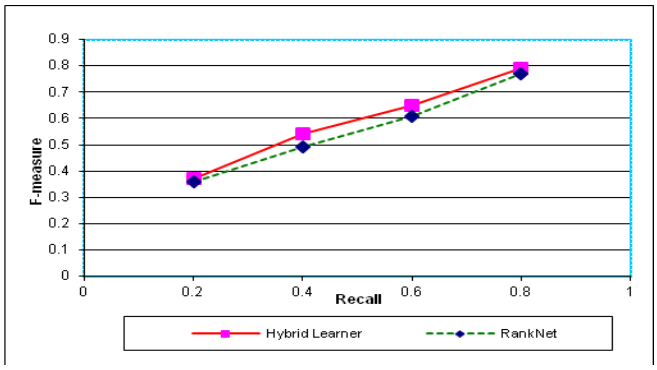

Figure 4. F-Measure using RankNet and Hybrid Learner

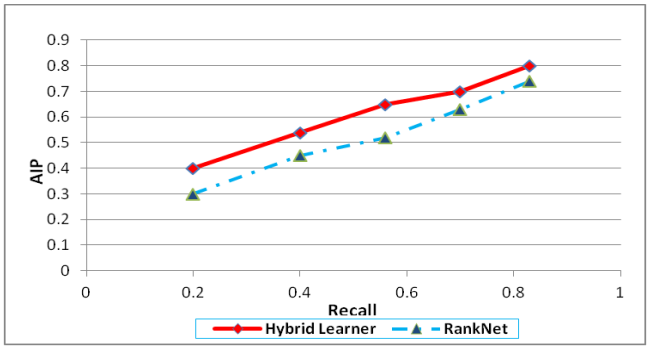

Figure 5. AIP using RankNet and Hybrid Learner

REFERENCES

[1] C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In Proceedings of the International Conference on Machine learning, pages 89–96, 2005.

[2] Caruana, R., Baluja, S., & Mitchell, T. (1996). Using the future to "sort out" the present: Rankprop and multitask learning for medical risk evaluation. NIPS 8 (pp. 959-965).

[3] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze. Introduction to Information Retrieval, Cambridge University Press,2008.

[4] Dekel, O., Manning, C., & Singer, Y. (2004). Log-linear models for label-ranking. NIPS 16.

[5] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, Hang Li: Listwise approach to learning to rank: theory and algorithm. ICML 2008: 1192-1199.

[6] Harrington EF. Online ranking/collaborative filtering using the perceptron algorithm. In: Proceedings of the twentieth international conference on machine learning (ICML-2003), Washington, DC.

[7] Herbrich, R., Graepel, T., & Obermayer, K. Large margin rank boundaries for ordinal regression. Advances in Large Margin Classifiers, MIT Press, pages 115-132, 2000.

[8] Joachims, T.: Optimizing search engines using clickthrough data. In: Proc. ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (SIGKDD '02), pages 133–142, 2002.

[9] K.Crammer and Y. Singer. Pranking with ranking. In Advances in Neural Information Processing Systems (NIPS), 2001.

[10] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, and W.-Y. Ma. Frank: a ranking method with fidelity loss. In SIGIR '07: Proceedings of the30th annual international ACM SIGIR conference on Research and development in information retrieval, pages 383–390, 2007.

[11] Mirella Lapata.Automatic Evaluation of Information Ordering: Kendall's Tau. In Association for Computational Linguistics.Volume 32,Number 4.2006.

[12] O. Chapelle, Y. Chang, and T.-Y. Liu. The Yahoo! Learning to Rank Challenge, 2010.

[13] Simon Haykin. Neural Networks and Learning machines, PHI Learning Private Limited, 2010.

[14] SVM-Light, http://svmlight.joachims.org/

[15] T. Joachims. Training linear svms in linear time. In KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, 2006.

[16] T. Qin, T.-Y. Liu, M. feng Tsai, X.-D. Zhang, and H. Li. Learning to search web pages with query-level loss functions. Technical Report, 2006.

[17] Thomas H. Cormen, Introduction to Algorithms, MIT Press. 2001.

[18] Tie-Yan Liu. Learning to Rank for Information retrieval, Springer.2008

[19] V. Kecman, Support Vector machines : Theory and Applications, Springer.2005

[20] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking SVM to document retrieval. In SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, 2006.

[21] Yoav Freund , Raj Iyer , Robert E. Schapire , Yoram Singer, An efficient boosting algorithm for combining preferences, The Journal of Machine Learning Research, 4, p.933-969, 12/1/2003

[22] Z. Cao, T. Qin, T.-Y. Liu, M-F. Tsai, H. Li.Learning to Rank: from pairwise approach to listwise approach.In Proceedings of 24th Annual International Conference on Machine Learning,pages:129- 136, 2007.

AUTHORS PROFILE

Pooja Arora was born in 1978 in Delhi, India. She took her Bachelor of Computer Science from Delhi University in 1998. Then she did her MCA from Banasthali Vidyapith Rajasthan, India. She has 10 years' experience in teaching MCA students in AKG engineering college, Uttar Pradesh India. Currently she is also pursuing Ph.D. under the guidance of Prof. Om Vikas and her area of research is information retrieval.

Prof. Om Vikas is former director IIITM, Gwalior India. He is also senior member IEEE, India.