An immune programming-based ranking function discovery approach for effective information retrieval

Shuaiqiang Wang, Jun Ma*, Qiang He

School of Computer Science and Technology, Software College Campus, Shandong University, Shunhua Road, 250101 Jinan, China

ARTICLE INFO

Keywords:
Information retrieval
Learning to rank
Immune programming
Evolutionary computation
Machine learning

ABSTRACT

In this paper, we propose RankIP, the first immune programming (IP) based ranking function discovery approach. IP is a novel evolution based machine learning algorithm with the principles of immune systems, which is verified to be superior to Genetic Programming (GP) on the convergence of algorithm according to their experimental results in Musilek et al. (2006).

However, such superiority of IP is mainly demonstrated for optimization problems. RankIP adapts IP to the learning to rank problem, a typical classification problem. In doing this, the solution representation, affinity function, and high-affinity antibody selection require completely different treatments. Besides, two formulae focusing on selecting best antibody for test are designed for learning to rank.

Experimental results demonstrate that the proposed RankIP outperforms the state-of-the-art learning-based ranking methods significantly in terms of $P_{10}$, $MAP$ and $NDCG@10$.

© 2010 Elsevier Ltd. All rights reserved.

1. Introduction

In an information retrieval (IR) system, a ranked list of documents is returned as a response for each query. Thus the ranking issue is critical to the effectiveness of such systems.

Several methods have been proposed to solve this problem, such as the boolean model, vector space model, probabilistic model, and language model, which can be regarded as empirical IR methods (Tsai, Liu, Qin, Chen, & Ma, 2007). In addition to these traditional IR approaches, machine learning techniques are becoming more widely used for the ranking problem of IR, referred to “learning to rank”. It aims 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 (Joachims, Li, Liu, & Zhai, 2007). Actually this area has become an active and growing research area both in information retrieval and machine learning communities, and lots of traditional classification methods have been adopted for it, e.g., (Cao et al., 2006; Freund, Iyer, Schapire, & Singer, 2003; Joachims, 2002; Xu & Li, 2007), etc.

Meanwhile, recently evolutionary computation (EC) based methods, especially Genetic Programming (GP) based technologies, have been successfully applied into this problem and gained some promising results, e.g., (Fan, Gordon, & Pathak, 2000, 2004a, 2004b, 2005; Trotman, 2005). Nowadays it becomes an important branch in the “learning to rank” area. EC is a kind of effective search or optimization techniques by mimicking the process of natural evolution in biology. In the theoretical and application research area of EC, there has recently been growing interest in the use of methods inspired by the immune systems or their principles and mechanisms (de Castro & Timmis, 2003). These systems have already been applied to numerous types of problems such as computer security, data analysis, clustering, pattern matching and parametric optimization (Dasgupta, Jr., & Gonzalez, 2003). Immune programming (IP) (Musilek, Lau, Reformat, & Wyard-Scott, 2006), is an extension of immune algorithms, particularly the clonal selection algorithm in AIS. Musilek et al. (2006) demonstrate that for optimization problem the convergence of IP is superior to GP, that is, IP can find an ideal antibody/individual in fewer generations with the most dramatic improvement evidently.

Thus we propose RankIP, by adapting IP into learning to rank, a classification problem. To validate our approach we performed experiments on the OHSUMED, TREC 2003 and 2004 data collections. Results indicate that the use of our framework leads to effective ranking functions that significantly outperform the baselines, include RankSVM (Joachims, 2002), RankBoost (Freund et al., 2003) and BM25 (Robertson, 1997) in terms of $MAP$, $NDCG@10$ and $P_{10}$.

In order to adapt IP, which is proposed for optimization problems, into the learning to rank problem, many adaption such as solution representation, affinity function, and high-affinity antibody
selection need to be considered. Besides, formulae focusing on selecting best antibody for test should be designed for learning to rank.

This paper is organized as follows. In Section 2, the related work are summarized. In Section 3, the background information on immune programming is provided, and RankIP, a novel immune programming-based approach for optimizing the performance measures with respect to the training and validation data, is presented in Section 4. Experimental results and discussions are described in Section 5. Finally, Section 6 concludes the paper.

2. Related work

2.1. Learning to rank using traditional classification methods

Opposite to the traditional IR methods, such as BM25 (Robertson, 1997) and LMIIR (Zhai & Lafferty, 2001), recently methods of “learning to rank” have been applied to ranking model construction and some promising results have been obtained. Joachims (2002) develops RankSVM, a support vector machine (SVM) based approach that utilizes click-through data for training, namely the query-log of the search engine in connection with the log of links the users clicked on in the presented ranking. Cao et al. (2006) introduce a Hinge Loss function for RankSVM, which aims to heavily penalize errors on the tops of ranking lists and errors from queries with fewer retrieved documents. Freund et al. (2003) describe and analyze an efficient algorithm called RankBoost for combining preferences by the boosting approach. Burges et al. develop a method called RankNet, which employs cross entropy as loss function and gradient descent as algorithm to train a neural network model. Xu and Li (2007) proposed AdaRank, which optimizes a loss function that is directly defined on the performance measures. It employs a boosting technique in ranking model learning. It offers several advantages: ease of implementation, theoretical soundness, efficiency in training, and high accuracy in ranking.

2.2. Learning to rank using evolutionary computation technologies

Different approaches to discover ranking functions based on GA and GP have been proposed in the literatures. Fan et al. (2000) propose a GP-based approach to automatically generate term weighting strategies for different contexts. They argue that each specific context demands a different term weighting strategy. In their work (Fan et al., 2004a, 2004b, Fan, Gordon, & Pathak, 2005) they demonstrate that GP has been effective at improving the performance of information retrieval tasks. In Fan, Fox, Pathak, and Wu (2004), they also study the effects of affinity functions on genetic programming-based ranking discovery for web search. In Trotman (2005), Trotman adds the baseline functions, such as those in Robertson, Walker, Jones, Hancock-Beaumont, and Gatford (1995), Salton and Buckley (1988) and Singhal et al. (1996) as individuals in the initial population. In de Almeida, Gonçalves, Cristo, and Calado (2007), Almeida et al. propose CCA, which uses parts of well-known, significant, and proven effective ranking formulas as terminals, for representing term weighting components, instead of simple statistical information. They believe that providing richer components for GP to work with will allow the discovery of better final ranking formulae.

3. Background: immune programming

In this section we will introduce immune programming (IP), an novel evolutionary computation (EC) approach for machine learning. Actually, IP is an extension of immune algorithms, particularly the clonal selection algorithm, inspired by the biological immune systems or their principles and mechanisms.

3.1. Biological immune systems

The immune system of vertebrates is composed of a great variety of molecules, cells, and organs spread throughout the body. There is no central organ controlling the immune system: various distributed elements perform complementary tasks. The main role of the immune system is to protect the organism against pathogens and to eliminate malfunctioning cells.

All elements recognizable by the immune system are called antigens: pathogens, malfunctioning cells, and healthy cells. The native cells, which originally belong to the organism and are harmless to its functions, are termed self or self antigens, while the disease-causing elements are named non-self or non-self antigens.

A immune system should be capable of distinguishing between what is self from what is non-self by B-cells and T-cells. These lymphocytes, thus, has to be capable of pattern recognition. However, that is not enough. In order to be effective in reacting to new pathogens and to improve response to pathogens already encountered, the immune system is endowed with memory and the ability to learn. The process of pathogen deactivation is described in de Castro and Von Zuben (2002). After successful recognition, cells capable of binding with pathogen are cloned, thus the lymphocytes with high affinities become memory cells with long life spans. The elements of this subpopulation also undergo mutations resulting in a subpopulation of cells. The variation provides the ability in that the subpopulation is able to recognize both the pathogen itself and similar ones, moreover, some of the mutated clones may have higher affinity. This whole process of selection and mutation is known as the maturation of the immune response (Nossal, 1993).

3.2. Immune programming

In immune programming, an antigen is used to represent the programming problem to be addressed and may be provided in closed form or as an input/output mapping. An antibody set (a repertoire/population), wherein each member represents a candidate solution, is generated at random from a gene library representing computer instructions. Affinity, the fit of an antibody (a solution candidate) to the antigen (the problem), is analogous to shape-complementarity evident in biological systems. This measure is used to determine both the fate of individual antibodies, and whether or not the algorithm has successfully completed.

### Algorithm 1. Immune programming operations

```
\textbf{Input:} the replacement, cloning and mutation probability: \(p_r, p_c,\) and \(p_m.\)
\textbf{Output:} the population \(\mathcal{P}\) with affinity set \(\mathcal{F}.
\begin{align*}
1 & \mathcal{N} \leftarrow \emptyset \\
2 & \text{While } \text{size}(\mathcal{N}) < \text{PopulationSize} \text{ do } \\
3 & \quad r \leftarrow \text{Replacement}(\mathcal{P}) \\
4 & \quad \text{if } r \neq \text{null then } \\
5 & \quad \quad \mathcal{N} \leftarrow \mathcal{N} \cup \{r\} \\
6 & \quad \text{else } \\
7 & \quad \quad i \leftarrow \text{GetGoodAntibody}(\mathcal{F}) \\
8 & \quad \quad c \leftarrow \text{Cloning}(\mathcal{P}, i, \mathcal{F}(i)) \\
9 & \quad \quad \text{if } c \neq \text{null then } \\
10 & \quad \quad \quad \mathcal{N} \leftarrow \mathcal{N} \cup \{c\} \\
11 & \quad \quad \text{else } \\
12 & \quad \quad \quad m \leftarrow \text{Mutation}(\mathcal{P}, i, \mathcal{F}(c)) \\
13 & \quad \quad \quad \mathcal{N} \leftarrow \mathcal{N} \cup \{m\} \\
14 & \text{end} \\
15 & \text{end} \\
16 & \text{end}
\end{align*}
```
As Algorithm 1 expressed, firstly the initialization executes by setting the next population as empty (Line 1). Then a new population is generated and compared to the parameter $g$, a new population is generated and compared according to the current population. In the next population, the function $P$ is generated which can estimate the relevance of the antibodies in the current population. Firstly an antibody $i$ with a high-affinity attribute string $m = (m_1, m_2, ..., m_l) \in S$, and replaces each attribute $m_i \in m$ with a new randomly generated value $v \in S$. This mutation probability is $\min(P_m(F(i), 1))$, so only a portion of the attributes is actually replaced.

4. Our method – RankIP

4.1. Formal definitions

The problem of information retrieval can be formalized as follows. For a query $q$ and a document collection $D$, the optimal retrieval system should return a ranking that orders the documents in $D$ according to their relevance to the query $q$.

Let $\mathcal{A}$ be the query set, for a given query, in the training data the relevance of the certain document is labeled as an integer number, formally, it is defined as a function $rel(q) : \mathcal{A} \rightarrow \mathbb{R}$. For example, for OHSUMED data collection, $rel(q) \in \mathbb{R}$ for stands for that the document is "not relevant" to the query $q$, $rel(q) = 1$ for "possibly relevant", and $rel(q) = 2$ for "definitely relevant". However, for some data collections, e.g. TREC, each document is given a binary judgment: "relevant" or "irrelevant". In this way, the boolean set $\mathcal{B}$ is set empty (Line 1) instead of $\mathbb{R}$ is enough.

A training example is expressed as a triple $t = (q, \mathcal{A}, \lambda(q))$ where $q \in \mathcal{Q}$, $\mathcal{A} \subseteq \mathcal{D}$, and the document order is $\lambda = (d_1, d_2, ..., d_n)$. Thus given a training data set $\mathcal{D}$ including the query set $\mathcal{A}$, we use tree structures to represent solutions in RankIP.

In RankIP, an antibody is actually a ranking function. Let $\mathcal{F}$ be the antibody set including all antibodies. Given a data set $\mathcal{D}$, the affinity of a certain antibody can be expressed as a real number. Thus the affinity function for the antibodies is defined as $\mathcal{F} : \mathcal{D} \rightarrow \mathbb{R}$.

Let $P_g \subseteq \mathcal{F}$ denote the set of a population at generation $g$. From a random subpopulation of antibodies $P_g \subseteq \mathcal{F}$, a selection operator $s : \mathcal{F} \rightarrow \mathcal{F}$ selects one antibody for variation that show a better affinity than others. Depending on the immune operations the population $P_{g+1}$ of the next generation $g + 1$ are generated.

An immune operator $v : \mathcal{F} \rightarrow \mathcal{F}$ creates one offspring out of the selected parent from the population $P_g$. This new antibody becomes a member of the population $P_{g+1}$. All immune operators must guarantee that no syntactically incorrect programs are generated during evolution.

4.2. RankIP algorithm

In learning to rank three types of the data collections are employed: training, validation and test. The training data set $\mathcal{D}$ is used to generate a series of candidate solutions $\mathcal{F}$ using IP algorithm, the validation set $\mathcal{V}$ helps in choosing good solutions that are not over-specialized for the training queries, and the test set $\mathcal{E}$ is the estimation data set for the ranking functions generated by RankIP.

Algorithm 2. RankIP algorithm

Input: the training, validation and test sets $\mathcal{D}$, $\mathcal{V}$ and $\mathcal{E}$; the affinity function $\mathcal{F}$; the maximum generation $\mathbb{G}$
Output: the best antibody $\mathcal{B}$

\begin{enumerate}
  \item $P_1 \leftarrow \text{RandInitPopulation}()$
  \item $\mathcal{A} \leftarrow \mathcal{D}$
  \item For $g \leftarrow 1 \text{ KwiTo } \mathbb{G}$ do
  \item \hspace{1em} $\mathcal{F} = \emptyset$
  \item \hspace{1em} $\mathcal{F} \leftarrow \mathcal{F} \cup \text{GetBestAntibody}(\mathcal{F}(\mathcal{D}))$
  \item \hspace{1em} $\mathcal{B} \leftarrow \text{GetBestAntibody}(\mathcal{F}(\mathcal{D}))$
  \item \hspace{1em} $\mathcal{B} \leftarrow \text{GetBestAntibody}(\mathcal{F}(\mathcal{D})(\mathcal{V}))$
  \item $P_{g+1} \leftarrow \text{GenerateNextGenerationByIP}(P_g)$
  \item $\mathcal{F} \leftarrow \emptyset$
  \item $\mathcal{B} \leftarrow \text{Select}(\mathcal{F}(\mathcal{D})(\mathcal{V})(\mathcal{E}))$
\end{enumerate}

The RankIP algorithm is shown as Algorithm 2. Initially a population of antibodies are generated randomly, and the candidate set is set empty (Line 1–2). Then the population evolves iteratively, where a candidate antibody, the individual with the best affinity value according to the training data, is selected from each generation, and form a candidate set $\mathcal{A}$ (Line 3–7). Finally, according to the affinity values of each candidate calculated with the training data and validation data respectively, the final antibody, i.e., the ranking function, is selected from $\mathcal{A}$.

Since the duplicates of certain antibody in $\mathcal{A}$ make no sense at all, the function $\text{GetBestAntibody}(\mathcal{F}(\mathcal{D}))$ will always return the best antibody per generation which has not recorded in the previous generations. That is to say, if the best antibody in the current generation has always existed in $\mathcal{A}$, it will return the second best one instead, and the rest may be deduced by analogy.

4.2.1. Solution representation

In Musilek et al. (2006) the authors adopt a linear representation for antibodies. However, although (Musilek et al., 2006) demonstrates that the stack-based model can handle some problems with three variables, there exists difficulty in dealing with many problems with more than three variables indeed. Since the "learning to rank" problems usually have dozens of terminals and their solutions are some functions, in order to learn non-linear ranking function, we use tree structures to represent solutions in RankIP.
In RankIP, two types of terminals are adopted: “Features” and “Constants”.

Features: The features are abstracted in certain data set, including some basic statistical information of the collection, documents and queries, such as term frequency \( tf \), etc. In addition, the conclusions of some classic approach, such as BM25 (Roberson, 1997) and LIMR (Zhai & Lafferty, 2001), are considered as the high-level features in OHSUMED and TREC data sets abstracted by the Microsoft research (Liu, Xu, Qin, Xiong, & Li, 2007). Although there are some common features in almost every data set, each data collection also keeps its own features, thus they may be quite different. Users can disassemble some features which they think them useless.

Constants: The constant set used in RankIP includes 0.1, 0.2, ..., 0.9, 1, 2, ..., 10. In addition to that, document number \( N \) is also counted in this type of terminals. Thus there are 20 constants in total.

RankIP adapts 10 functions in total: six two-variables functions (including +, -, \( \times \), \( \div \), min and max) and four single-variables functions (including \( \sin \), cos, log and sqrt). Since these functions are ones with no more than three input parameters, thus the functions can be defined as a triple \( \langle \Sigma, x, y \rangle \), where \( \Sigma \) is its function symbol, \( x \) and \( y \) are its parameters. If the function is a single-variable function, the second parameter \( y \) is expressed as “\( \cdot \lbrack \cdot \rbrack \)” standing for nothing.

Some functions need the protected parameters, e.g., in the logarithm function \( \log(x) \), the parameter \( x \) should be greater than zero. Thus we design two protected mechanisms for the protected parameter \( x \):

- \( \text{PM}(1) \) if \( x < 0 \), then \( x := \left\lfloor x \right\rfloor \).
- \( \text{PM}(2) \) if \( x = 0 \), then \( x := \varepsilon \), where \( \varepsilon \) is a real number close to zero.

In our experiments \( \varepsilon = 0.000001 \).

We call the protected mechanism \( x\text{-PM}(i) \) if variable \( x \) needs the \( i \)th protected mechanism, where \( i = 1 \) or 2. Table 1 presents these protected mechanisms for the functions in RankIP.

A tree can be parsed into a function. Fig. 1 shows the tree representation for an example ranking function \( f_1 - f_2 + 0.5 \times f_3 \) where \( f_1, f_2 \) and \( f_3 \) are basic features of learning to rank.

### 4.2.2. Affinity functions

Due to the flexibility of the tree-based architecture, the traditional affinity measure, Minkowski distance, will not be appropriate any more.

Some GP-based “learning to rank” approaches usually regard the evaluation measures, e.g., mean average precision (MAP), as the fitness (affinity) function directly. However, there are some remarkable differences between IP and GP-based approaches, and the most significant one is the manner how to choose an alternative genetic operator to execute in the evolutionary process of the population. Specifically, in a GP-based system an alternative genetic operator is chosen to execute only according to certain probabilities. On the contrary, such decision in RankIP is made with respect to not only some probabilities, but also the affinity function value of each antibody ready to be operated, i.e., the antibody with high-affinity value will have more probability to be cloned and less chance to be mutated. Since the probability is defined over the range of the number set \( \mathcal{A} = \{0.1\} \), the affinity function value had better be well-distributed over \( \mathcal{A} = \{0.1\} \) as well. Actually if the affinity values are much lower than 1, IP tends to behavior like a Monte Carlo process.

Unfortunately, evaluation measures are usually less than \( iv \) (the ideal value), which is far lower than 1. Thus a mapping process is necessary in RankIP. Firstly we estimate a constant \( iv \), an ideal affinity value of the output antibody. Thus the domain of the mapping function \( \mathcal{A} = \{0.1\} \) can be decided. The following step is to design a mapping function \( f : \mathcal{A} \rightarrow \mathcal{A} \) to transform the affinity value of antibodies onto the range set \( \mathcal{A} \). A challenge lies in that at first we do not know the proper value of \( iv \) for a given data set. Since we try to make RankIP close to a heuristic strategy rather than a Monte Carlo process, convex functions which can output fairly high scores despite of lower input values should be adopt as the mapping function. In our scheme, \( f(x) \) is a logarithm function, expressed formally as follows:

\[
f(x) = \log_{10}\left(1 + \frac{9 \times x}{10} \right), \quad x \in \mathcal{A}
\]

where \( x \) is the evaluation measure of certain antibody.

### 4.2.3. Antibody selection for clone and mutation

As mentioned in Algorithm 1, once a high-affinity antibody selected in the current generation, clone and mutation operators can get ready to execute. In Musilek et al. (2006) antibodies are selected in a sequential manner, and it is opposite to the principles of immune system. So we introduce the conception “Deme” into our approach, which originates from an important technology of GA and GP (Collins, 1992).

Particularly, in RankIP an antibody \( I \) is chosen randomly at first, then itself together with its \( \text{DemeSize} - 1 \) neighbors comprise a deme, where \( \text{DemeSize} \) is the number of the candidates, called deme size. As soon as a deme \( \mathcal{D} = \{I_1, I_2, \ldots, I_{\text{DemeSize}}\} \) is decided, a selection can be processed as follows: the probability of certain antibody which can be chosen is proportional to its affinity. Formally,

\[
\Pr(I_k | I_k \in \mathcal{D}) = \frac{\mathcal{F}(I_k)}{\sum_{I_k \in \mathcal{D}} \mathcal{F}(I_k)}
\]

### 4.2.4. Selection of the best antibody

As mentioned before, we use a validation set to help in choosing good solutions that are not over-specialized for the training queries, i.e., that are able to generalize for unseen queries. In Lacerda et al. (2008), the choice of the best antibody is accomplished by considering the average performance of an antibody in both the training and validation sets minus the standard

![Fig. 1. A sample tree-based antibody.](image-url)
deviation value. In de Almeida et al. (2007) this method is called AVG$_e$, formally, if the candidate antibody is $I_i \in \mathcal{A}$, the best antibody is selected by:

$$AVG_e : \arg\max_i \left( \frac{\mathcal{F}(I_i) + \mathcal{F}(V)(I_i)}{2} - \sigma_i \right)$$

The antibody with the highest value of AVG$_e$ will be selected as the best. In de Almeida et al. (2007) authors argue that this method may not lead to the best performance in some runs. They propose a similar method, which also considers the dispersal between training and validation values, but uses the sum of these values in place of the average. They call this method SUM$_e$:

$$SUM_e : \arg\max_i \left( (\mathcal{F}(I_i) + \mathcal{F}(V)(I_i)) - \sigma_i \right)$$

Since the training data collection is far greater than the validation data set, we do not believe it appropriate to assume that the training and validation results have the same weights, as shown in Eqs. (3) and (4). Furthermore, we do not admit that different datasets should apply the same selecting formula. According to our experiments, we propose two new similar methods: $\omega -$SUM$_e$ and $\omega -$SUM$_{e2}$:

$$\omega -$SUM$_e : \arg\max_i \left( (\mathcal{F}(I_i) + \mathcal{F}(V)(I_i)) - \sigma_i \right)$$

$$\omega -$SUM$_{e2} : \arg\max_i \left( (\mathcal{F}(I_i) + \mathcal{F}(V)(I_i)) - \gamma \times \sigma_i \right)$$

The values of $\alpha$ and $\beta$ are according to the size of training dataset $\mathcal{F}$ and validation dataset $V$, respectively, that is,

$$\alpha = \frac{\text{size}(\mathcal{F})}{\text{size}(\mathcal{F}) + k \times \text{size}(\mathcal{V})}$$

$$\beta = \frac{k \times \text{size}(\mathcal{V})}{\text{size}(\mathcal{F}) + k \times \text{size}(\mathcal{V})}$$

$$\gamma = m$$

where both $k$ and $m$ are constants.

The only difference between Eqs. (5) and (6) is that the former adopts the standard deviation $\sigma$ while the latter employs variance $\sigma^2$.

Obviously, when $\text{size}(\mathcal{V}) = k \times \text{size}(\mathcal{F})$ and $\gamma = 1.0$, Eq. (5) is equivalent to the Eq. (3); when $\text{size}(\mathcal{V}) = k \times \text{size}(\mathcal{F})$ and $\gamma = 0.5$, Eq. (5) is equivalent to Eq. (4).

In our experiments, we let $k = 2$ and $m = 0.5$. Since for the OHSUMED and TREC data set, $\text{size}(\mathcal{F}) = 3 \times \text{size}(\mathcal{V})$, according to Eq. (7), $\alpha = 0.6$ and $\beta = 0.4$ in the experiments.

5. Experiments

We use three data sets in the experiments, i.e., OHSUMED, a benchmark data set for document retrieval and TREC, a data set obtained from web track of TREC 2003 and 2004. These data collections are all provided by Microsoft research web site.

We compared the ranking accuracies of RankIP with those of three baseline methods: Ranking SVM, RankBoost and BM25. The ranking performances of both Ranking SVM and RankBoost are evaluated and reported in Liu et al. (2007). Table 2 shows the control parameters in RankIP.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>OHSUMED</th>
<th>TREC 2003</th>
<th>TREC 2004</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antibody depth</td>
<td>6</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Cloning probability</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Random seed</td>
<td>1234567890</td>
<td>1234567890</td>
<td>1234567890</td>
</tr>
</tbody>
</table>

Table 2

Control parameters for OHSUMED and TREC data collections.

Table 3

Features adopted on OHSUMED in RankIP.

<table>
<thead>
<tr>
<th>Id</th>
<th>Formulations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1-02</td>
<td>$\sum_{w,d} c(w,d)$</td>
<td>Raw term frequency (tf) (Baeza-Yates &amp; Ribeiro-Neto, 1999)</td>
</tr>
<tr>
<td>F1-04</td>
<td>$\sum_{w,d} \log(c(w,d) + 1)$</td>
<td>Feature in SIGIR paper (Cao et al., 2006)</td>
</tr>
<tr>
<td>F1-06</td>
<td>$\sum_{w} \log(c(w) + 1)$</td>
<td>Normalized term frequency (tf$/$idf) (Baeza-Yates &amp; Ribeiro-Neto, 1999)</td>
</tr>
<tr>
<td>F1-08</td>
<td>$\sum_{w,d} \log(d_{w,d} + 1)$</td>
<td>Feature in SIGIR paper (Cao et al., 2006)</td>
</tr>
<tr>
<td>F1-10</td>
<td>$\sum_{w,d} \log(d_{w,d} \log(d_{w,d} + 1)) + 1$</td>
<td>Feature in SIGIR paper (Baeza-Yates &amp; Ribeiro-Neto, 1999)</td>
</tr>
<tr>
<td>F1-12</td>
<td>$\sum_{w} \log(c(w) \log(c(w) + 1))$</td>
<td>Combinations of term frequency (tf) and inverse document frequency (idf) (Baeza-Yates &amp; Ribeiro-Neto, 1999)</td>
</tr>
<tr>
<td>F1-14</td>
<td>$\sum_{w,d} \log(d_{w,d} \log(d_{w,d} + 1)) + 1$</td>
<td>Feature in SIGIR paper (Baeza-Yates &amp; Ribeiro-Neto, 1999)</td>
</tr>
<tr>
<td>F1-16</td>
<td>BM25: $\log(BM25)$</td>
<td>BM25 score and its logarithm value (Robertson, 1997)</td>
</tr>
<tr>
<td>F1-18</td>
<td>LMR$\text{abs}$, LMR$\text{dir}$, LMR$\text{jm}$</td>
<td>LMR with DIR, JM and ABS smoothing respectively (Zhai &amp; Lafferty, 2001)</td>
</tr>
</tbody>
</table>

Note that $c(w,d)$ represents the count of the word $w$ in the document $d$; $C$ represents the entire collection; $||$ denotes the size of function.
Microsoft research, including low-level content features, high-level content features, hyperlink features and hybrid features.

Each data set was partitioned into five parts in order to conduct five-fold cross validation. For each fold, three parts are used for training, one part for validation, and the remaining part for testing. The training set is used to learn the ranking model. The validation set is used to tune the parameters of the ranking model, and the test set is used to report the ranking performance of the model. Note that since we conduct five-fold cross validation, the reported performance in this paper is actually the average over different folds.

5.2. Evaluation measures

We use three standard rank-aware accuracy measures, precision at n \( P@n \), mean average precision (MAP), and normalized discount cumulative gain (NDCG).

\( P@n \) measures the accuracy within the top \( n \) results of the returned ranked list for a query

\[
P@n = \frac{\# \text{ of relevant docs in top } n \text{ results}}{n}
\]

MAP takes the average of the average precision values over all queries, where the average precision for each query is defined as the average of the \( P@n \) values for all relevant documents, formally,

\[
\text{average precision} = \frac{\sum_{n=1}^{N} (P@n \times \text{rel}(n))}{\# \text{ relevant docs for this query}}
\]

where \( \text{rel}(n) \) is a binary function on the relevance of the \( n \)th document.

Note that \( P@n \) and MAP can only handle cases with binary judgment, relevant or irrelevant. Recently, a new evaluation measure NDCG (Järvelin & Kekäläinen, 2002) has been proposed, which can handle multiple levels of relevance judgments.

\[
\text{NDCG} = \frac{\sum_{j=1}^{n} \left( \frac{2^{r_j} - 1}{\log_2(j)} \right)}{\sum_{j=1}^{n} \left( \frac{2^{r_j} - 1}{\log_2(j)} \right)_\text{perfect list}}
\]

where \( r_j \) is the rating of the \( j \)th document in the ranking list, and the normalization constant \( Z_n \) is chosen so that the perfect list gets a NDCG score of 1.

5.3. Performance

As Table 3 shown, there are 19 features adopted on OHSUMED data collection in RankIP. Counting the 20 constants mentioned above, there are 39 terminals in this experiment. Accordingly, features abstracted in Liu et al. (2007) are classified into low-level and high-level features for the OHSUMED. There are seven low-level features adopted in RankIP from the fields of title and abstract respectively, including term frequency, normalized term frequency, combinations of term frequency and inverse document frequency, and some other features in SIGIR paper (Cao et al., 2006). High-level features include the outputs of BM25 (Robertson, 1997) and LMIR (Zhai & Lafferty, 2001) algorithms. In particular, for LMIR, different smoothing methods (DIR, JM, ABS) (Zhai & Lafferty, 2001) and LMIR (Zhai & Lafferty, 2001) algorithms. In particular, for LMIR, different smoothing methods (DIR, JM, ABS) (Zhai & Lafferty, 2001) algorithms. In particular, for LMIR, different smoothing methods (DIR, JM, ABS) (Zhai & Lafferty, 2001) algorithms.

### Table 3

<table>
<thead>
<tr>
<th>Id</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F01–04</td>
<td>dl, Document length of body, anchor, title and URL</td>
</tr>
<tr>
<td>F05–08</td>
<td>tf, Term frequency of body, anchor, title and URL</td>
</tr>
<tr>
<td>F09–12</td>
<td>idf, Inverse document frequency of body, anchor, title and URL</td>
</tr>
<tr>
<td>F13–16</td>
<td>tfidf, Combinations of ( t ) and ( idf ) of body, anchor, title and URL</td>
</tr>
<tr>
<td>F17</td>
<td>BM25, BM25 of anchor, title and extracted title (Robertson, 1997)</td>
</tr>
<tr>
<td>F18–22</td>
<td>Sitemap based score propagation and feature propagation (Qin et al., 2005)</td>
</tr>
<tr>
<td>F23–25</td>
<td>LMIR.ABS of anchor, title and extracted title (Zhai &amp; Lafferty, 2001)</td>
</tr>
<tr>
<td>F26–28</td>
<td>LMIR.DIR of anchor, title and extracted title (Zhai &amp; Lafferty, 2001)</td>
</tr>
<tr>
<td>F29–31</td>
<td>LMIR.JM of anchor, title and extracted title (Zhai &amp; Lafferty, 2001)</td>
</tr>
<tr>
<td>F32–34</td>
<td>Hyperlink base feature propagation: weighted in-link, weighted out-link and uniform out-link (Qin et al., 2005; Shakery &amp; Zhai, 2003)</td>
</tr>
</tbody>
</table>

Table 4

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RankIP</th>
<th>RankSVM</th>
<th>RankBoost</th>
<th>BM25</th>
</tr>
</thead>
<tbody>
<tr>
<td>OHSUMED</td>
<td>0.4584</td>
<td>0.4469</td>
<td>0.4403</td>
<td>0.4361</td>
</tr>
<tr>
<td>TREC 2003</td>
<td>0.2404</td>
<td>0.2564</td>
<td>0.2125</td>
<td>0.1254</td>
</tr>
<tr>
<td>TREC 2004</td>
<td>0.4009</td>
<td>0.3505</td>
<td>0.3835</td>
<td>0.3351</td>
</tr>
</tbody>
</table>

We demonstrate the ranking performances of RankIP, RankSVM, RankBoost and BM25 on the OHSUMED subset in Table 4 and Fig. 2. For OHSUMED we evaluate the measure MAP, NDCG@1–10 and P@1–10 for comparison. We can see that RankIP outperforms the three baseline methods of Ranking SVM, RankBoost and BM25 in...
terms of almost all measures with exceptions of P@5. Especially for NDCG@1, RankIP achieves more than 12% relative improvement.

As Table 5 shown, for TREC data set, there are 34 features adopted for TREC data collection in RankIP, including the outputs of BM25 (Robertson, 1997) and LMIR (Zhai & Lafferty, 2001) algorithms, sitemap-based relevance propagation (Qin, Liu, Zhang, Chen, & Ma, 2005) and hyperlink base feature propagation (Shakery & Zhai, 2003), which are abstracted in Liu et al. (2007). Counting the 20 constants mentioned above, there are 54 terminals in this experiment.

For TREC 2003 and 2004 we evaluate MAP and NDCG@1–5 for comparison. Fig. 3 and Table 4 give these results. We can see that the RankIP also outperforms the baseline methods in most measures.

5.4. Sensitivity analysis

5.4.1. Selecting formulae

In Section 4.2 we introduce two selecting formulae $\omega - SUM_r$ and $\omega - SUM_{r^2}$, expressed by Eqs. (5) and (6) respectively. As Table 2 illustrated, $\omega - SUM_r$ is adopted for OHSUMED while $\omega - SUM_{r^2}$ is employed for TREC.

We conceive four experiments to validate it: the data sets are Fold1, four of OHSUMED and Fold1, three of TREC 2003. For these two types of data sets, both selecting formulae are used to choose the best antibody. Table 6 shows the evaluated measures (MAP for TREC and NDCG@5 for OHSUMED) of the selected antibody in the training, validation and test processes respectively. Note that

- The inputs of the selecting formulae are not the original data listed in Table 6, but the outputs of the mapping function, evaluated by Eq. (1).
- The function getBestAntibody($F, X$) in Algorithm 2 will always return the best antibody per generation which has not been recorded in the previous generations, as mentioned in the Section 3.2.
- The improvement proportions are computed from the measure values (MAP for TREC or NDCG@5 for OHSUMED) on the test data collections, with the bold font in the table.

From Table 6 we can see that the $\omega - SUM_{r^2}$ improves evidently compared with that gained by $\omega - SUM_r$ for the OHSUMED data set. In addition, more dramatic improvement is demonstrated on TREC, especially for the data collection of Fold3 in TREC, the MAP of the antibody selected by $\omega - SUM_{r^2}$ is only 0.048868, while those values are raised to 0.299279 sharply if $\omega - SUM_r$ is used instead, improved more than 500% !

We also enumerate those corresponding results of the AVGr (Lacerda et al., 2006) and SUMr (de Almeida et al., 2007), expressed by Eqs. (3) and (4) respectively, in Table 6. We can conclude that neither of them can always select the best antibody.

5.4.2. Other parameters

To examine the sensitivity of the RankIP algorithm with respect to its parameters, the OHSUMED data collection is considered.

<table>
<thead>
<tr>
<th>Selecting formulae</th>
<th>Fold 1</th>
<th>Fold 2</th>
<th>Fold 3</th>
<th>Fold 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega - SUM_r$</td>
<td>0.4826</td>
<td>0.4349</td>
<td>0.1754</td>
<td>0.2587</td>
</tr>
<tr>
<td>$\omega - SUM_{r^2}$</td>
<td>0.4780</td>
<td>0.4631</td>
<td>0.4813</td>
<td>0.4826</td>
</tr>
<tr>
<td>$\omega - SUM_{r^3}$</td>
<td>0.4631</td>
<td>0.4813</td>
<td>0.5789</td>
<td>0.4826</td>
</tr>
<tr>
<td>$\omega - SUM_{r^4}$</td>
<td>0.3427</td>
<td>0.3427</td>
<td>0.4939</td>
<td>0.3316</td>
</tr>
<tr>
<td>$\omega - SUM_{r^5}$</td>
<td>0.3427</td>
<td>0.3427</td>
<td>0.4939</td>
<td>0.3316</td>
</tr>
<tr>
<td>Mean average precision (MAP) for TREC 2003</td>
<td>0.2587</td>
<td>0.2587</td>
<td>0.1705</td>
<td>0.1705</td>
</tr>
</tbody>
</table>

5.4. Sensitivity analysis

5.4.1. Selecting formulae

In Section 4.2 we introduce two selecting formulae $\omega - SUM_r$ and $\omega - SUM_{r^2}$, expressed by Eqs. (5) and (6) respectively. As Table 2 illustrated, $\omega - SUM_r$ is adopted for OHSUMED while $\omega - SUM_{r^2}$ is employed for TREC.

We conceive four experiments to validate it: the data sets are Fold1, four of OHSUMED and Fold1, three of TREC 2003. For these two types of data sets, both selecting formulae are used to choose the best antibody. Table 6 shows the evaluated measures (MAP for TREC and NDCG@5 for OHSUMED) of the selected antibody in the training, validation and test processes respectively. Note that

- The inputs of the selecting formulae are not the original data listed in Table 6, but the outputs of the mapping function, evaluated by Eq. (1).
- The function getBestAntibody($F, X$) in Algorithm 2 will always return the best antibody per generation which has not been recorded in the previous generations, as mentioned in the Section 3.2.
- The improvement proportions are computed from the measure values (MAP for TREC or NDCG@5 for OHSUMED) on the test data collections, with the bold font in the table.

From Table 6 we can see that the $\omega - SUM_{r^2}$ improves evidently compared with that gained by $\omega - SUM_r$ for the OHSUMED data set. In addition, more dramatic improvement is demonstrated on TREC, especially for the data collection of Fold3 in TREC, the MAP of the antibody selected by $\omega - SUM_{r^2}$ is only 0.048868, while those values are raised to 0.299279 sharply if $\omega - SUM_r$ is used instead, improved more than 500% !

We also enumerate those corresponding results of the AVGr (Lacerda et al., 2006) and SUMr (de Almeida et al., 2007), expressed by Eqs. (3) and (4) respectively, in Table 6. We can conclude that neither of them can always select the best antibody.

5.4.2. Other parameters

To examine the sensitivity of the RankIP algorithm with respect to its parameters, the OHSUMED data collection is considered.
Varied parameters are listed, below, with their default values indicated in parentheses. Ideal values: $iv_1 = 0.65$, $iv_2 = 0.65$, $iv_3 = 0.64$, $iv_4 = 0.63$, $iv_5 = 0.64$.

### Table 7
Normalized discount cumulative gain (NDCG) at position 1–10 with respect to the parameter $iv$ for the OHSUMED data.

<table>
<thead>
<tr>
<th>$iv$</th>
<th>@1</th>
<th>@2</th>
<th>@3</th>
<th>@4</th>
<th>@5</th>
<th>@6</th>
<th>@7</th>
<th>@8</th>
<th>@9</th>
<th>@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.56</td>
<td>0.5637</td>
<td>0.4700</td>
<td>0.4807</td>
<td>0.4751</td>
<td>0.4611</td>
<td>0.4599</td>
<td>0.4602</td>
<td>0.4532</td>
<td>0.4512</td>
<td>0.4501</td>
</tr>
<tr>
<td>0.57</td>
<td>0.5398</td>
<td>0.4825</td>
<td>0.4743</td>
<td>0.4607</td>
<td>0.4538</td>
<td>0.4540</td>
<td>0.4583</td>
<td>0.4528</td>
<td>0.4482</td>
<td>0.4467</td>
</tr>
<tr>
<td>0.58</td>
<td>0.5208</td>
<td>0.4447</td>
<td>0.4515</td>
<td>0.4474</td>
<td>0.4486</td>
<td>0.4518</td>
<td>0.4482</td>
<td>0.4480</td>
<td>0.4460</td>
<td>0.4423</td>
</tr>
<tr>
<td>0.59</td>
<td>0.5639</td>
<td>0.4943</td>
<td>0.4836</td>
<td>0.4710</td>
<td>0.4585</td>
<td>0.4551</td>
<td>0.4612</td>
<td>0.4531</td>
<td>0.4534</td>
<td>0.4490</td>
</tr>
<tr>
<td>0.60</td>
<td>0.5453</td>
<td>0.4879</td>
<td>0.4815</td>
<td>0.4695</td>
<td>0.4598</td>
<td>0.4561</td>
<td>0.4553</td>
<td>0.4571</td>
<td>0.4547</td>
<td>0.4540</td>
</tr>
<tr>
<td>0.61</td>
<td>0.5703</td>
<td>0.4388</td>
<td>0.4398</td>
<td>0.4356</td>
<td>0.4279</td>
<td>0.4254</td>
<td>0.4268</td>
<td>0.4291</td>
<td>0.4306</td>
<td>0.4297</td>
</tr>
<tr>
<td>0.62</td>
<td>0.5081</td>
<td>0.4677</td>
<td>0.4546</td>
<td>0.4444</td>
<td>0.4489</td>
<td>0.4485</td>
<td>0.4484</td>
<td>0.4442</td>
<td>0.4440</td>
<td>0.4438</td>
</tr>
<tr>
<td>0.63</td>
<td>0.5608</td>
<td>0.4742</td>
<td>0.4742</td>
<td>0.4658</td>
<td>0.4527</td>
<td>0.4501</td>
<td>0.4517</td>
<td>0.4477</td>
<td>0.4494</td>
<td>0.4466</td>
</tr>
<tr>
<td>0.64</td>
<td>0.5734</td>
<td>0.4719</td>
<td>0.4686</td>
<td>0.4533</td>
<td>0.4516</td>
<td>0.4472</td>
<td>0.4477</td>
<td>0.4478</td>
<td>0.4468</td>
<td>0.4430</td>
</tr>
<tr>
<td>0.65</td>
<td>0.5734</td>
<td>0.4813</td>
<td>0.4725</td>
<td>0.4590</td>
<td>0.4490</td>
<td>0.4461</td>
<td>0.4497</td>
<td>0.4430</td>
<td>0.4431</td>
<td>0.4414</td>
</tr>
</tbody>
</table>

$\sigma = 0.0229$, $0.0177$, $0.0147$, $0.0120$, $0.0094$, $0.0095$, $0.0098$, $0.0078$, $0.0068$, $0.0066$.

Fig. 4. Sensitivity of IP performance NDCG@5 with respect to the population $PopulationSize$ for the OHSUMED data.

Fig. 5. Sensitivity of IP performances with respect to the max generation $GenerationMax$ for the OHSUMED data.

Fig. 6. Sensitivity of IP performances with respect to the deme size $DemeSize$ for the OHSUMED data.

- Max generation: $GenerationMax = 100$.
- Deme size: $DemeSize = 9$. 

(a) $NDCG@5$ for OHSUMED

(b) $NDCG@10$ for OHSUMED
Effect of the ideal value: To examine the effect of the ideal value, \( iv \), its value is varied from 0.56 to 0.65 in increments of 0.01. The evaluation measures NDCG@5 and NDCG@10, as well as the regression analyses, are plotted against the population size, PopulationSize, in Fig. 4. Overall, the curves show increasing trends and reach the maximum when PopulationSize is 500.

Effect of the max generation: The population size, GenerationMax is varied from 10 to 100 in increments of 10. The evaluation measures NDCG@5 and NDCG@10 are plotted against the max generation, GenerationMax, in Fig. 5. Overall, the curves show increasing trends and reach the top of the graph when GenerationMax < 70.100.

Effect of the demesize: The value of the demesize, DemeSize is varied from 3 to 13 in increments of 2. The evaluation measure NDCG@5 is plotted against the demesize, DemeSize, in Fig. 6. The bar graph demonstrates that there is a maximum of NDCG@5 at a value of DemeSize 9 for the OHSUMED data collection.

6. Conclusions

On the basis of the tree-based representation architecture, in this paper we presented RankIP, an approach for learning to rank with the goal of improving the accuracy of conventional IR and Web searching. In order to adapt IP to the learning to rank problem, we employed the mapping mechanism for RankIP to make sure that the affinity values of the antibodies are well-distributed over the range [0,1]. Besides, we introduced the demes technology to the IP algorithms. Furthermore, two formulae focusing on selecting best antibody for test were designed for learning to rank.

We used OHSUMED, TREC 2003 and 2004 data collections to validate our approach, and the experiments showed that the performance based on our RankIP improved evidently compared with Ranking SVM, RankBoost and BM25.

For OHSUMED collection, our experiment results demonstrated that RankIP led to significant improvements, especially NDCG@1 is improved by 40.14% and 12.76% over BM25 and RankBoost respectively. Mean average precision and precision at the top 5, 10, as well as the regression analyses, are plotted against the ideal value, iv, its value is varied from 0.56 to 0.65 in increments of 0.01. The evaluation measure NDCG@5 is plotted against the ideal value, iv (0.63), in Fig. 4. Overall, the curves show increasing trends and reach the maximum when iv is varied from 0.56 to 0.65 in increments of 0.01. The evaluation measure NDCG@5 is plotted against the max generation, GenerationMax, in Fig. 5. Overall, the curves show increasing trends and reach the top of the graph when GenerationMax < 70.100.

Several experiments also confirmed our hypotheses that it was unnecessary, and even unfeasible, to construct a universally applicable selecting formula to choose the final antibody. Actually our experiments showed that an appropriate selecting formula was vital for a certain data collection.

However, similar with the evolutionary algorithms, the performance of RankIP is related to its control parameters. As yet there are still no theoretical methods to verify their perfect values suitable for all applications. At present the parameter values mainly depend on the empirical methods. We constructed several experiments and found the optimum parameters used in RankIP finally.

Acknowledgments

Thanks are given to anonymous referees for the helpful suggestions and comments that they provided. The authors acknowledge that this research is supported by the Natural Science Fund of China No. 60970047 and the Key Science-Technology Project of Shandong Province of China No. 2008GG1001026.

References


Liu, T.-Y., Xu, J., Qin, T., Xiong, W., & Li, H. (2007). LETOR: Benchmark dataset for research on learning to rank for information retrieval. In SIGIR workshop on learning to rank for IR (LBIR), ACM.


Shandong Province of China No. 2008GG10001026.

