Discovering Ranking Functions for Information Retrieval

Weiguo Fan
Virginia Polytechnic Institute and State University, USA

Praveen Pathak
University of Florida, USA

INTRODUCTION

The field of information retrieval deals with finding relevant documents from a large document collection or the World Wide Web in response to a user's query seeking relevant information. Ranking functions play a very important role in the retrieval performance of such retrieval systems and search engines. A single ranking function does not perform well across different user queries, and document collections. Hence it is necessary to "discover" a ranking function for a particular context. Adaptive algorithms like genetic programming (GP) are well suited for such discovery.

BACKGROUND

In an information retrieval system (IR), given a user query a matching function matches the information in the query with that in the documents to rank the documents in decreasing order of their predicted relevance to the user. The top ranked documents are then presented to the user as a response to her query. To facilitate this relevance estimation process both the documents and the queries need to be transformed in a form that can be easily processed by computers. Vector Space Model (VSM) (Salton, 1989) is one of the most successful models to represent the documents and queries. We choose this model as the underlying model in our research due to the ease of interpretation of the model and the tremendous success in retrieval performance studies. Moreover most existing search engines and IR systems are based on this model.

In VSM, both documents and queries are represented as vectors of terms. Suppose there are t terms in the collection, then a document D and a query Q are represented as:

$$D = (w_{d1}, w_{d2}, \ldots, w_{dt})$$
$$Q = (w_{q1}, w_{q2}, \ldots, w_{qt})$$

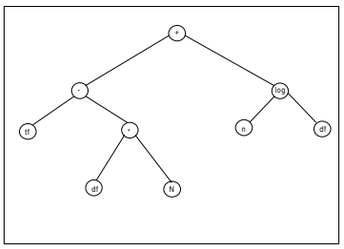
where w_{di}, w_{qi} (for i=1 to t) are weights assigned to different terms in the document and the query respectively. The similarity between the two vectors is calculated as the cosine of the angle between the two vectors. It is expressed as (Salton & Buckley, 1988):

$$Similarity(Q,D) = \frac{\sum_{i=1}^{t} w_{qi} w_{di}}{\sqrt{\sum_{i=1}^{t} (w_{qi})^2 * \sum_{i=1}^{t} (w_{di})^2}} \quad (1)$$

The score, called retrieval status value (*RSV*), is calculated for each document in the collection and the documents are ordered and presented to the user in the decreasing order of *RSV*. Various content based features are available in VSM to compute the term weights. The most common ones are the *term frequency (tf)* and the inverse *document frequency (idf)*. Term frequency measures the number of times a term appears in the document or the query. The higher this number, the more important the term is assumed to be in describing the document. Inverse document frequency is calculated as $\log(N/DF)$, where N is the total number of documents in the collection, and DF is the number of documents in which the term appears. A high value of *idf* means the term appears in a relatively few number of documents and hence the term is assumed to be important in describing the document. A lot of similar content based features are available in literature (Salton, 1989; Salton & Buckley, 1988). The features can be combined (e.g. *tf*idf*) to generate a variety of new composite features that can be used in term weighting.

Equation (1) suggests that in order to discover a good ranking function, we need to discover the optimal way of assigning weights to document and query keywords. Change in term weighting strategy will essentially change the behavior of a ranking function. In this chapter we describe a method to discover an optimal ranking function.

Figure 1. A sample tree representation for a ranking function



MAIN THRUST

In this chapter we present a systematic and automatic discovery process to discover ranking functions. The process is based on an artificial intelligence technique called genetic programming (GP). GP is based on genetic algorithms (GA) (Goldberg, 1989; Holland, 1992). Because of the intrinsic parallel search mechanism and powerful global exploration capability in high-dimensional space, both GA and GP have been used to solve a wide range of hard optimization problems. They are used in various optimal design and data-mining applications (Koza, 1992).

GP represents the solution to a problem as a *chromosome* (or an individual) in a *population* pool. It evolves the population of *chromosomes* in successive generations by following the genetic transformation operations such as reproduction, crossover, and mutation to discover chromosomes with better fitness values. A fitness function assigns a fitness value for each *chromosome* that represents how good the *chromosome* is at solving the problem at hand.

We use GP for discovering ranking functions because of four reasons. First, in GP there is no stringent requirement for an objective function to be continuous. All that is needed is that the objective function should be able to differentiate good solutions from the bad ones. This property allows us to use common IR performance measures, like "average precision" (P_Avg), which are non-linear in nature as objective functions. Second, GP is well suited to represent the common tree based representations for the solutions. A tree-based representation allows for easier parsing and implementation. An example of a term weighting formula using tree structure is given in Figure 1. We will use such a tree based representation in this chapter. Third, GP is very effective for non-linear function and structure discovery problems where traditional optimization methods do not seem to work well (Banzhaf, Nordin, Keller, & Francone, 1998). Finally, it has been empirically found that GP discovers better solutions than those obtained by conventional heuristic algorithms.

Ranking function discovery as presented in this chapter is different from classification in that we seek to find a function that will be used for ranking or prioritizing documents. There are efforts in IR that treat this as a classification problem in which a classifier or discriminant function is used for ranking. But evidence shows that ranking function discovery has yielded better retrieval results than the results obtained by treating this as a classification problem using Support Vector Machines and Neural Networks (Fan, Gordon, Pathak, Wensi, & Fox, 2004; Fuhr & Pfeifer, 1994). We now proceed to describe the discovery process using GP.

Ranking Function Discovery by GP

In order to apply GP in our context we need to define several components for it. We use the tree structure as shown in Figure 1 to represent term weighting formula. Components needed for such a representation are given in Table 1.

For the purpose of our discovery framework we will define these parameters as follows:

- An *individual* in the population is expressed in term of a tree which represents one possible ranking function. A *population* in a *generation* consists of *P* such trees.
- **Terminals:** We use the features mentioned in Table 2 and real constants as the terminals.
- **Functions:** We use +, -, *, /, and log as the functions allowed

Table 1. Essential GP components

GP Parameters	Meaning
Terminals	Leaf nodes in the tree data structure
Functions	Non-leaf nodes used to combine the leaf nodes. Typically numerical operations
Fitness Function	The objective function that needs to be optimized
Reproduction and Crossover	Genetic operators used to copy *fit* solutions from one generation to another and to introduce *diversity* in the population