TUNING FOR DISTRIBUTED DATABASE USING EVOLUTIONARY APPROACH

G.Nagaleela*1, K.Sreekala*2

M.Tech, Dept of CSE, SKDEC, Gooty, D.t: Anantapuram, A.P, India

M.Tech, Assistant Professor, Dept of CSE, SKDEC, Gooty, D.t: Anantapuram, A.P, India

ABSTRACT:

In this paper, we propose a new method to discover collection- adapted ranking functions based on Genetic Programming (GP). Our Combined Component Approach (CCA) is based on the combination of several term-weighting components (i.e., term frequency, collection frequency, normalization) extracted from well-known ranking functions. In contrast to related work, the GP terminals in our CCA are not based on simple statistical information of a document collection, but on meaningful, effective, and proven components. Experimental results show that our approach was able to outperform standard TF-IDF, BM25 and another GP-based approach in two different collections.

We apply Genetic Programming (GP) techniques. Our experiments with the ACM Computing Classification Scheme, using documents from the ACM Digital Library, indicate that GP can discover similarity functions superior to those based solely on a single type of evidence. Effectiveness of the similarity functions discovered through simple majority voting is better than that of content-based as well as combination-based Support Vector Machine classifiers. Experiments also were conducted to compare the performance between GP techniques and other fusion techniques such as Genetic Algorithms (GA) and linear fusion. Empirical results show that GP was able to discover better similarity functions than GA or other fusion techniques.

KEYWORDS: Genetic Algorithm, support vector machines, programming language.

I.INTRODUCTION

In recent years, automated classification of text into pre-defined categories has attracted considerable interest, due to the increasing volume of documents in digital form and the ensuing need to organize them. However, traditional content-based classifiers are known to perform poorly when documents are noisy (e.g., digitized through speech recognition or OCR) and/or contain scarce textual content (e.g., metadata records in digital library (DL) catalogs) [3]. DLs offer both (1) the opportunity to explore the complex internal structured nature of documents and metadata records in the classification task; and (2) the social networks occuring in specific communities, as expressed for example by citation patterns in the research literature. On the other hand, many DLs, which are created by aggregation of other sub-collections/catalogs, suffer from problems of quality of information. One such problem is incompleteness (e.g., missing information). This makes it very hard to classify documents using traditional content-based classifiers like SVM, or Naive Bayes. Another quality problem is imprecision. For example, citation-based information is often obtained with OCR, a process which produces a significant number of errors. In this work we try to overcome these problems by applying automatically discovered techniques for fusion of the available evidence. Particularly, we investigate an inductive learning method – Genetic Programming (GP) – for the discovery of better fused similarity functions to be used in the classifiers, and explore how this combination can be used to improve classification effectiveness. One motivation of this work is to enhance teaching and learning in the computing field, by improving CITIDEL [5], part of the National Science Digital Library (NSDL). Many of the records in CITIDEL lack classification information. That makes it difficult for users to browse, even with the support of our advanced multischeming

approach. It also limits the scope of advanced interfaces to CITIDEL, which make use of category information. Further, the lack of classification information reduces broader downstream impact that could result from use of records harvested from CITIDEL and similar systems, such as into NSDL or NDLTD (e.g., to support browsing of computing dissertations

The growth in volume of the Web and other textual repositories, such as digital libraries, throughout the last decade, has made the information retrieval task difficult, costly, and in many cases, very complex for the end user. In this con- text, search engines became valuable tools to help users find content relevant to their information needs. Naturally, research on information retrieval models that can effectively rank search results according to document relevance has become a fundamental subject.

Information Retrieval models have come a long way. Although the most popular is still undoubtedly the vector space model proposed by Salton, many new or complementary alternatives have been proposed, such as the Probabilistic Model. From all these models, document ranking formulas can be derived for document searching. Thus, many alternatives exist on how to compose a ranking function. Most of them have a common characteristic: they attempt to be very general in nature, i.e., they were designed to be applied in any type of collection. The work of Zobel and Moffat [26], for example, presented more than one million possibilities to compute a similarity function. However, after all the experiments, they concluded that no weighting scheme is consistently good in all collections. That is, a ranking function can have success in one domain but fail in another. Further, they comment that it would be prohibitive to discover the best weighting scheme simply by an exhaustive exploration of the similarity space.

In this work, we discover specialized ranking strategies for specific collections. Our method is able to consider the important and unique characteristics of each collection so that the discovered function is more effective than any general solution. To accomplish this, we use Genetic Programming (GP), a machine learning technique inspired by Darwinian evolutionary processes, to discover specific ranking functions for each document collection. GP has been successful in many IR problems. GP was chosen due to its ability to find any arbitrary function, even when dealing with very large search spaces. However, differently from other GP-based approaches, which use only basic statistical information from terms and documents, our strategy uses rich, meaningful, and proven effective components present in well-known ranking formulas, such as Okapi BM25 and Pivoted TF-IDF. Our assumption is that by providing these human-discovered formula components as building blocks, the GP process can take advantage of all the human knowledge that has been applied to produce them. As a con- sequence, it will be able to better explore the search space.

To validate our GP approach we performed experiments with the TREC-8 and WBR99 collections. Results indicate that the use of meaningful components in a GP-based frame- work leads to effective ranking functions that significantly outperform the baselines (standard TF-IDF, BM25 and another GP-based approach [9]). Our Combined Component. Approach (CCA) ranking functions also converged to good results faster than the GP approach used as baseline, and the overtraining also was reduced.

II. BACKGROUND

2.1. Genetic programming

GAs and GP belong to a set of artificial intelligence problem-solving techniques based on the principles of biological inheritance and evolution. Each potential solution is called an individual (i.e., a chromosome) in a population. Both GA and GP work by iteratively applying genetic transformations, such as crossover and mutation, to a population of individuals to create more diverse and better performing individuals in subsequent generations. A fitness function is available to assign a fitness value for each individual. The *main difference* between

GA and GP relies on their internal representation---or data structure---of an individual. In general, GA applications represent each individual as a fixed-length bit string, like (1101110 . . .) or a fixed-length sequence of real numbers (1.2, 2.4, 4, . . .). In GP, on the other hand, more complex data structures are used. Fig. 2 shows an example of a tree representation of a GP individual. Furthermore, the length of a GP data structure is not fixed, although it may be constrained by implementation to be within a certain size limit. Because of their intrinsic parallel search mechanism and powerful global exploration capability in a high-dimensional space, both GA and GP have been used to solve a wide range of hard optimization problems that oftentimes have no known optimum solutions.

2.2. GP Components

In order to apply GP to solve a given problem, several key components of a GP system need to be defined. Table 1 lists these essential components along with their descriptions. The entire combination discovery framework can be seen as an iterative process. Starting with a set of training images with known relevance judgments, GP first operates on a large population of random combination functions (individuals). These combination functions are then evaluated based on the relevance information from training images. If the stopping criteria is not met, it will go through the genetic transformation steps to create and evaluate the next generation population iteratively. GP searches for good combination functions by evolving a population along several generations. Population individuals are modified by applying genetic transformations, such as *reproduction*, *mutation*, and *crossover*. The reproduction operator selects the best individuals and copies them to the next generation. The two main variation operators in GP are mutation and crossover. Mutation can be defined as random manipulation that operates on only one individual. This operator selects a point in the GP tree randomly and replaces the existing subtree at that point with a new randomly generated subtree.The crossover operator combines the genetic material of two parents by swapping a subtree of one parent with a part of the other (see Fig. 3).

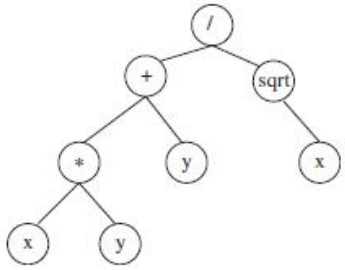


Fig. 2. A sample tree representation.

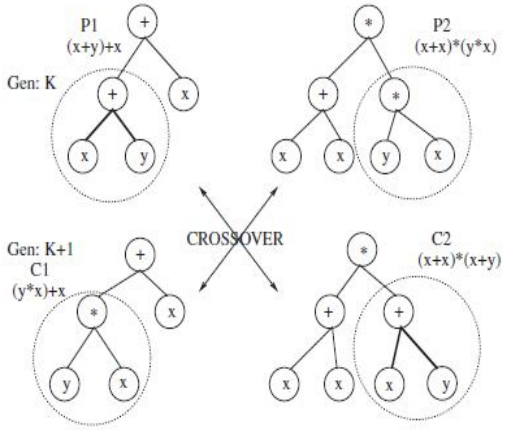


Fig. 3. A graphical illustration of the crossover operation [14].

III. IMPLEMENTATION

3.1 Genetic Operations

Usually, a GP method evolves a population of tree structures, also called *individuals*, each one representing a single solution to a given problem. In our experiments, the trees are arithmetic functions, as illustrated in Figure 1. These individuals are handled and modified by genetic operations like *reproduction*, *crossover*, *mutation*, *evaluation* and *selection*, in an iterative process that hopefully spawn better individuals (solutions to the proposed problem) in the subsequent generations.

When using trees in a GP-based method, a set of terminals and functions should be defined. Terminals are inputs, constants or zero arguments2 nodes that terminate a branch of a tree, they are also called tree leaves. The function set is the collection of

operators, statements and basic or user defined functions that can be used by the GP process to manipulate the terminal values.

The crossover operation allows genetic content exchange between two parents, in a process that can generate two or more children. In a GP method, two parent trees are selected according to a matching selection policy, and then, a random subtree is selected in each parent. The children trees result from the swap of the selected subtrees between the parents, as illustrated in Figure 2. The mutation operation has the role of keeping a minimum diversity level of individuals in the population. Every solution tree resulting from the crossover phase has an equal chance of suffering a mutation process. In a GP tree schema, a random node in the chosen tree is selected and the pointed subtree is replaced by a new randomly created subtree. Normally, the mutation rate is an evolutionary run parameter and should have a low probability; to avoid damaging good The evaluation of an individual is accomplished by assigning a value based on how well it deals with the proposed problem. In the GP environment, the individuals are evaluated on how well they learn to predict good answers to a given problem, using the set of functions and terminals available. This grade is also called *individual* or *raw fitness* and the evaluation functions are called *fitness functions*. The selection operation holds the responsibility of applying a criterion for choosing the individuals that should be in the next generation. After the evaluation process, each solution has a fitness value measuring how good or bad it is to the given problem. Using these values, it is possible to decide whether an individual should be in the next generation. Strategies for the selection process may use very simple or complex techniques, varying from just selecting the best *n* individuals to randomly selecting the individuals proportionally to their fitness. Reproduction is just the process that copies the individuals that will participate in the crossover and selection processes, without modifying them.

3.2 Generational Evolutionary Algorithm

In this work, the GP evolutionary process is guided by a *generational evolutionary* algorithm. This means that there are well-defined and distinct generation cycles. This is the basic idea around all evolutionary algorithms.

The steps describing the algorithm cycle are the following:

1. Initialize the population (with random or user provided individuals).
2. Evaluate all individuals in the present population, assigning a numeric rating or fitness value to each one.
3. Reproduce the best *n* individuals into the next generation population.
4. Apply the genetic operations (reproduction, selection, crossover and mutation) to all individuals in the present population.
5. Using a selection process, select *m* individuals that will compose the next generation with the best parents.
6. If the termination criterion is fulfilled, then continue. Otherwise, replace the existing generation with new generated population and repeat steps 2 to 5.
7. Present the best individual in the population as the output of the evolutionary process.

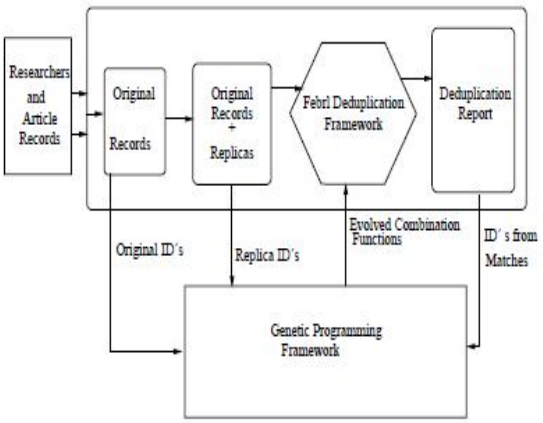
3.3 GP Experimentation Evaluation

In this work, the process of combining evidences to create a record similarity function using GP is separated in two phases:

• The GP training phase, in which the characteristics of similar records are learned.

• The testing phase, in which the best trees selected in the training set are used to identify replicas in a set of records different from the one used in the first phase in this work.

The second phase, besides measuring the real effectiveness of the generated solutions, also serves to verify if the trees evolved during the training phase have not been overspecialized for the features present in the training data set and are not generalized for other data sets.



IV. Our Experimental Environment

Our idea to guarantee generalization for the generated solutions is to use as a sample in the training phase a small but statistically representative part of the data available in the DL being examined in order to generate effective and generalizable similarity functions that could be used for the entire DL or for another DL with similar characteristics.

V.CONCLUSION

In this paper, we considered the problem of classification in the context of document collections where textual content is scarce and imprecise citation information exists. A framework for tackling this problem based on Genetic Programming has been proposed and tested. Our experimental results on two different sets of documents from each level of the ACM Computing Classification System have demonstrated that the GP framework can be used to discover better similarity functions that, when applied to a *kNN* algorithm, can produce better classifiers than ones using individual evidence in isolation. Our experiments also showed that the framework achieved results better than both traditional content-based and combination-based SVM classifiers. Comparison between the GP framework and linear fusion as well as GA also showed that GP has the ability to discover better similarity functions.

VI.REFERENCES

[1] R. Amsler. Application of citation-based automatic classification. Technical report, The University of Texas at Austin, Linguistics Research Center, Austin, TX, 2012.

[2] P. Calado, M. Cristo, E. S. de Moura, N. Ziviani, B. A. Ribeiro-Neto, and M. A. Gon¸calves. Combining link-based and content-based methods for Web document classification. In *Proc. of CIKM-03*, pages 394–401, New Orleans, US, 2013.

[3] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proc. of SIGMOD*, pages 307–318, Seattle, 2008.

[4] S. M. Cheang, K. H. Lee, and K. S. Leung. Data classification using genetic parallel programming. In *GECCO-03*, volume 2724 of *LNCS*, pages 1918–1919, Chicago, 2003.

[5] CITIDEL. Computing and Information Technology Interactive Digital Educational Library, www.citidel.org, 2004. [6] C. Clack, J. Farringdon, P. Lidwell, and T. Yu. Autonomous document classification for business. In *AGENTS-97*, pages 201–208, 1997.

[7] D. Cohn and T. Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In *NIPS 13*, pages 430–436. MIT Press, 2001.

[8] J. Dean and M. R. Henzinger. Finding related pages in the World Wide Web. *Computer Networks*, 31(11–16):1467–1479, 1999. Also in Proceedings of the 8th International World Wide Web Conference.

[9] M. D. del Castillo and J. I. Serrano. A multistrategy approach for digital text categorization from imbalanced documents.*SIGKDD*, 6(1):70–79, 2004.

[10] J. Eggermont, J. N. Kok, and W. A. Kosters. Genetic programming for data classification: Refining the search space. In *Proc. of BNAIC-03*, pages 123–130, Nijmegen, 2003.

[11] W. Fan, E. A. Fox, P. Pathak, and H. Wu. The effects of fitness functions on genetic programming-based ranking discovery for web search. *JASIST*, 55(7):628–636, 2004.

[12] W. Fan, M. D. Gordon, and P. Pathak. Discovery of context-specific ranking functions for effective information retrieval using genetic programming. *TKDE-04*, 16(4):523–527, 2004.

[13] W. Fan, M. D. Gordon, P. Pathak, W. Xi, and E. A. Fox. Ranking function optimization for effective web search by genetic programming: An empirical study. In *Proc. Of HICSS-04*, pages 105–112, Hawaii, 2004.

[14] M. Fisher and R. Everson. When are links useful? Experiments in text classification. In *Proc. of ECIR-03*, pages 41–56, 2003.

[15] J. Furnkranz. Exploiting structural information for text classification on the WWW. In *IDA-99*, pages 487–498, 1999.

[16] E. J. Glover, K. Tsioutsiouliklis, S. Lawrence, D. M. Pennock, and G. W. Flake. Using Web structure for classifying and describing Web pages. In *Proc. of WWW-02*, 2002.

[17] M. Gordon. Probabilistic and genetic algorithms for document retrieval. *CACM*, 31(10):1208–1218, 1988.

[18] J. H. Holland. *Adaption in Natural and Artificial Systems.* MIT Press, Cambridge, MA, 1992.

[19] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proc. of ECML-98*, pages 137–142, Chemnitz, Germany, 1998.

[20] T. Joachims, N. Cristianini, and J. Shawe-Taylor. Composite kernels for hypertext categorisation. In *Proc. of ICML-01*, pages 250–257, Williams College, US, 2001.

AUTHOR'S PROFILES:

G.Nagaleela,
M.Tech (CSE),
SRI KRISHNA DEVARAYA ENGINEERING COLLEGE,
JNTU Ananthapuram.

K.Sreekala M.Tech,
Assistant Professor, SRI KRISHNA DEVARAYA ENGINEERING COLLEGE,
JNTU Ananthapuram.