Determining general term weighting schemes for the Vector Space Model of Information Retrieval using Genetic Programming

Ronan Cummins and Colm O'Riordan

Dept. of Information Technology,
National University of Ireland,
Galway, Ireland.
E-mail: ronan.cummins@nuigalway.ie
colmor@it.nuigalway.ie

Abstract. Term weighting schemes play a vital role in the performance of many Information Retrieval models. The vector space model is one such model in which the weights applied to the document terms are of crucial importance to the accuracy of the retrieval system. This paper outlines a procedure using genetic programming to automatically determine term weighting schemes that achieve a high average precision. The schemes are tested on standard test collections and are shown to perform consistantly better than the traditional *tf-idf* weighting schemes. We present an analysis of the evolved weighting schemes to explain their increase in performance. These term weighting schemes are shown to be general across various collections and are shown to adhere to Luhn's theory as both high and low frequency terms are assigned a low weight.

1 Introduction

Information Retrieval (IR) deals with the retrieval of information according to specification by subject. IR stems from traditional data retrieval and deals with retrieving information from semi-structured or unstructured information sets using natural language queries. With the advent of World Wide Web and the vast increase in the quantity of information available, the need to retrieve information based on a user's query has become increasingly important. IR systems attempt to return only documents that are relevant to a given information need. However, the use of natural language in both queries and document sets leads to difficulties in retrieving accurate information for the user due to a number of reasons - e.g. ambiguity, synonomy and polysemy. An IR system must make an effort to interpret the semantic content of a document and rank the documents in relation to the user's query.

The vector space model [14] is one of the most widely known and studied IR models. This is mainly due to its simplicity, its efficiency over large document collections and the fact that it is intuitively appealing. The effectiveness of vector based models depend crucially on the term weighting applied to the terms of the document vectors [13].

Founded in the early 1990's, the Genetic Programming area [7] has grown quickly and helped solve problems in a wide variety of areas including robotic control, pattern recognition, music and synthesis of artificial neural networks. GP is based on the Darwinian theory of natural selection [1], where individuals that have a higher fitness will survive and thus produce offspring. These offspring will inherit characteristics similar to their parents and, through successive generations, beneficial characteristics will flourish. GP can be viewed as an artificial way of selective breeding.

This paper presents a Genetic Programming framework that artifically breeds term weighting schemes for the vector space model. The next section introduces some background material in both Information Retrieval and Genetic Programming. Some past approaches of evolutionary computation techniques applied to IR are also reviewed in this section. Section three describes the system and experimental design. Results and analysis are discussed in detail in section four. Finally, our conclusions are discussed in section five.

2 Background

This section presents background material for the vector space model and genetic programming. Traditional term weighting schemes are introduced and Luhn and Zipf's contribution to IR is summarised. The basic genetic program is outlined and some existing evolutionary approaches to the IR problem are briefly reviewed.

2.1 Standard term weighting approaches

The classic vector space model represents each document in the collection as a vector of terms with weights associated to each term. The weight of each term is based on the frequency of the term in the documents and collection. The query (user need) is also modelled as a vector and a matching function is used to compare each document vector to the query vector. Once the documents are compared, they are sorted into a ranked list and returned to the user. The *tf-idf* family of weighting schemes [13] is the most widely used weighting schemes for the vector space model. The term-frequency *(tf)* is a document specific local measure and is calculated as follows:

$$tf = \frac{rtf}{max\_freq} \tag{1}$$

where *rtf* is the actual term frequency and the *max_freq* is the frequency of the most common term in the document. The *max_freq* measure is used as the normalisation factor because longer documents tend to have more terms and higher term frequencies. The *idf* part of the weighting scheme is an Inverse Document Frequency measure [16]. The main heuristic behind the *idf* measure, is that a term that occurs infrequently is good for discriminating between documents. The *idf* of a term is a global measure and is determined by using the formula:

$$idf_t = log(\frac{N}{df_t}) \tag{2}$$

where N is the number of documents in the collection and df_t is the number of documents containing the term t. Therefore idf_t will be a larger figure for terms that are rarer across the document set. The weight assigned to a term is a product of *tf* and *idf*. One of the most common and simplest matching functions of the vector space model is the inner-product measure and is calculated as follows:

$$sim(d_i, q) = \sum_{k=1}^{t}(w_{ik} \times q_k) \tag{3}$$

where q is the query, d_i is the i^{th} document in the document set, t is the number of terms in the document, w_{ik} is the weight of term k in document i and q_k is the weight of term k in the query.

It has been shown by Zipf [18] that the frequency of terms in a collection when placed in rank order approximately follows a log curve . Furthermore, it was proposed by Luhn [10] that terms that occur too frequently have little power to distinguish between documents and that terms that appear infrequently are also of little use in distinguishing between documents. Thus in Fig.1, the bell-shaped curve of the graph relates the frequency of terms to their distinguishing (resolving) power.

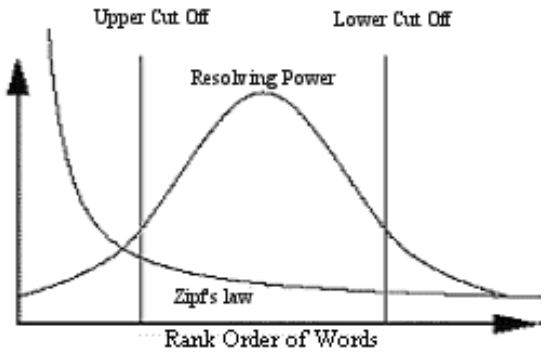

Fig. 1. Zipf's law and Luhn's proposed cut-off points [15]

2.2 Genetic Programming

GP is a heuristic stochastic searching method that is efficient for navigating large, complex search spaces. The advantage of this evolutionary approach is that it can help to solve problems in which the roles of variables are not correctly understood. GP is often used to automatically define functions whose variables combine and react in complex ways.

Initially, a random population of solutions is created. The solutions are modelled in a tree-like structure with operators as internal nodes and operands as leaf nodes. These nodes are often referred to as genes and their values as alleles. Each solution is rated based on how it performs in its environment. This is achieved using a fitness function. Once this is done, reproduction can occur. Solutions with a higher fitness will produce more offspring. Goldberg uses the roulette wheel example where each solution is represented by a segment on a roulette wheel proportionately equal to the fitness of the solution [3]. Reproduction (recombination) can occur in variety of ways. The most common form is sexual reproduction where two different individuals (parents) are selected and two separate children are created by combining the genotypes of both parents. The coded version of a solution is called its genotype, as it can be thought of as the genome of the individual, while the solution in its environment is called its phenotype. The fitness is evaluated on the phenotype of a candidate solution while reproduction and crossover is performed on the genotype. Once the recombination process is complete each individual's fitness in the new generation is evaluated and the selection process starts again. The algorithm usually ends after a certain number of generations, after convergence of the population or after an individual is found with an acceptable fitness.

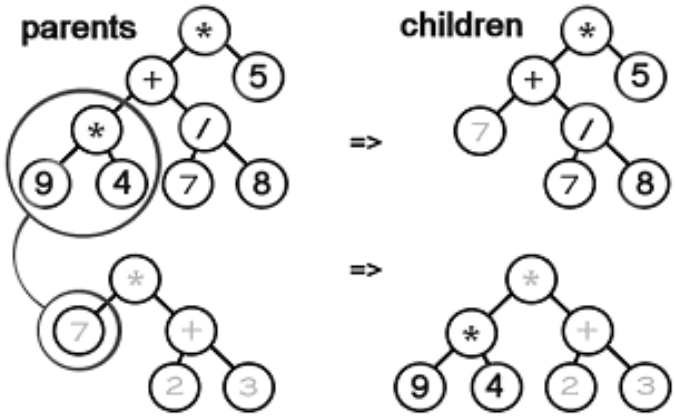

Fig. 2. Example of crossover in GP

2.3 Evolutionary Computation applied to IR

There have been several approaches to adopting and applying ideas from evolutionary computation in the domains of information retrieval. These vary in the form of evolutionary algorithm used and also in the manner of their application.

Related work where a genetic programming technique is used to evolve weighting functions which outperform the traditional *tf-idf* weighing schemes has previously been conducted [11]. Fan et al. [2] also adopt a genetic programming framework to search for weighting functions and test them against more modern weighting functions. However, these approaches lack a detailed analysis of the weighting schemes presented.

A genetic algorithm approach to modifying document representations (a set of keywords) based on user interaction has also been adopted [4]. By evolving sets of weights for the document (i.e. evolving the representation), better descriptions for the document in the collection can be found.

Horng and Yeh [6] extract keywords from a subset of relevant documents to construct a query and use a genetic algorithm to adapt the weights to best suit the relevant documents.

3 Design and Experimental Setup

The GP approach adopted in this work evolves the weighting scheme over a number of generations. An initial population is created randomly by combining a set of primitive meaures (e.g. df_t, rtf, N) using a set of operators (e.g. $+$, -, $\times$, /). The average precision, used as the fitness function, is calculated for each scheme by comparing the ranked list returned by the system for each weighting scheme against the human determined relevant documents for each query. Average precision is calculated over all points of recall and is frequently used as a performance measure in IR systems. The matching function used in all experiments is the inner-product matching function.

The three document collections used in this research are the Medline, CISI and Cranfield collections[1]. The documents and queries are pre-processed by removing standard stop-words and stemmed using Porter's stemming algorithm [12]. The weighting scheme applied to the query terms is a relative term frequency weighting scheme.

All experiments are run for 50 generations with an initial population of 1000. Experimental analysis shows us that the population converges before 50 generations when using the largest terminal and function set. The solutions are trained on an entire collection and query set. They are then tested for generality on the collections that were not included in training.

Trees are limited to a depth of 6 to promote generality as shorter solutions are usually more general [8]. The aim is to discover general natural language characteristics that will aid retrieval performance and not to learn characteristics specific to a single document collection. The following tables show the function and terminal sets used in our experiments:

[1] ftp://ftp.cs.cornell.edu/pub/smart

Table 1. Terminal Set

Terminal	*Description*
1	*the constant* 1
rtf	raw term frequency within a document
l	document length (no. of unique words in a document)
df	no. of documents a term appears in
N	no. of documents in a collection
max_freq	frequency of the most common term in a document
tl	total document length (no. of words in a document)
V	vocabulary of collection (no. of unique terms in the collection)
C	collection size (total number of terms in the collection)
cf	collection frequency (frequency of a term in the collection)
max_c_freq	frequency of the most common term in the collection

Table 2. Function Set

Function	*Description*
+, ×, /, -	addition, multiplication, division and subtraction functions
log	the natural log
sin, tan	trigonometric functions
sqrt	square-root function
sq	square

4 Results and Analysis

The aim of the first experiment is to evolve general solutions for each of the document collections and show that they are general by testing the solutions on previously unseen data. The second experiment analyses the characteristics of the evolved solutions.

4.1 Experiment one

This first experiment uses all of the terminal and function set in Table 1 and Table 2. The depth of the solutions is limited to a depth of 6 to promote generality as it is seen from prior experiments that solutions with higher depths are often very specific to the training set. Solutions are evolved on each of the document sets to compare the generality of the solutions. Table 3 shows the differences in average precision for solutions when evolved on different document collections.

Firstly, it can be seen that the evolved weighting schemes show a significant improvement in average precision over the *tf-idf* solution. It can also be seen that the solutions achieved an average precision, on the collections on which they were not trained, which is within 1% to 2% of the best solution found on the collection on which they were trained (underlined). This shows that the solutions are quite general and exploit general natural language characteristics.

Table 3. Average precision for best solutions found on each collection

				Training set		
Collection	*Docs*	*Qrys*	$tf-idf$	*CISI*	*Medline*	*Cranfield*
CISI	1460	76	20.74%	25.47%	24.86%	24.03%
Medline	1033	30	49.04%	56.74%	58.85%	56.69%
Cranfield	1400	225	37.44%	41.33%	41.85%	43.04%

The *cf* measure is seen to occur consistantly in the fittest solutions from this experiement.

4.2 Experiment two

This experiment uses only the global measures in the terminal set. The global part of the weighting scheme is evolved to investigate the properties of the *cf* measure. Thus, the terminals used in this experiment are limited to *cf, df, 1* and *N*. The following is the best solution evolved on the CISI collection using only these global measures:

$$gw_t = \frac{log(N/df)}{\sqrt{df}} \times log(\frac{cf}{df}) \times log(df) \tag{4}$$

Table 4 shows the average precision for the CISI training collection and the collections that were not included in training. We see that the precision of new global scheme is consistantly higher than that of the *idf* measure.

Table 4. Average precision for gw_t and *idf* for the CISI solution

Collection	*Docs*	*Qrys*	*idf*	gw_t
CISI	1460	76	18.85%	22.25%
Medline	1033	30	46.63%	54.09%
Cranfield	1400	225	33.41%	37.06%

When the terms in the collection are placed in rank order, as shown in Fig.3, the gw_t weight of these terms is similar to that which Luhn predicted would lead to identifying terms with a high resolving power. More recently, it has also been predicted that a flattening of the *idf* measure at both high and low frequencies would result in increased precision [5]. The reason for the flattening of the curve at low frequency levels is due to the presents of the *cf* measure. An interesting point to note is that the global weighting scheme identified completely ignores terms that occur once or twice across a collection. This weighting scheme also completely eliminates terms of all frequencies that are totally concentrated in

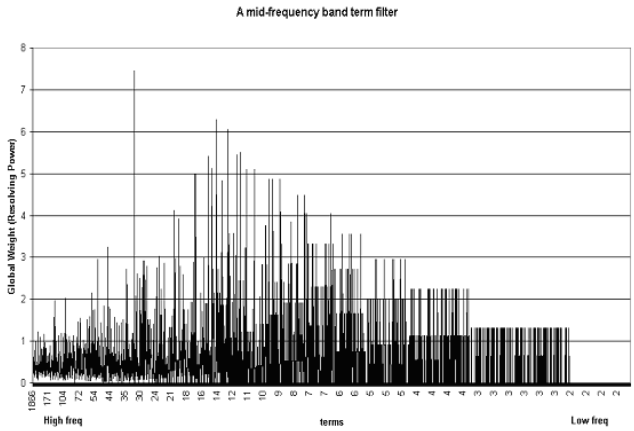

Fig. 3. gw_t for terms placed in rank order for the CISI collection

one document (i.e. have a document frequency of 1). It also eliminates terms that occur exactly once in every document and thus whose concentration is very low. This has the effect of considerably reducing the size of the vocabulary of the collection as typically words that appear once or twice represent about 65% of the vocabulary of a corpus. It is interesting that these characteristics are identified by evolutionary techniques to be advantageous, since they are used in some feature extraction techniques like document-frequency thresholding [9]. The traditional *idf* measure also appears in the evolved global weighting scheme, reinforcing its value in global weighting schemes.

Low frequency terms are poor discriminators because they cannot distinguish between as many documents as higher frequency terms. Consider a query with 3 terms *(t1, t2 and t3)*. If each term in the query has a document frequency of 1, the maximum number of different sets of documents that can be distinguished by the query is 4 (3 of the sets containing only one document), regardless of the frequency of the term within the document. However, if the document frequency of all 3 terms is greater than 4, the maximum number of different sets of documents that can possibly be distinguished increases to 8. Thus, if a query has k terms, the terms with the highest resolving power have a document frequency of $N - k \geq df_t \geq 2^{k-1}$ where df_t is the document frequency of a term and N is the number of documents in the collection. In Fig.4, C represents the document collection and the sub-sets represent document sets that contain terms with the same document frequency.

The *idf* part of the *tf-idf* scheme will incorrectly identify terms with a low frequency as having a high resolving power. This weakness in the *tf-idf* scheme is due to the fact that there is only one global measure. The *tf* measure counteracts *idf* to some degree but only in a local context. In formulating a scheme for term weighting, firstly terms within the collection with a high resolving power, must be

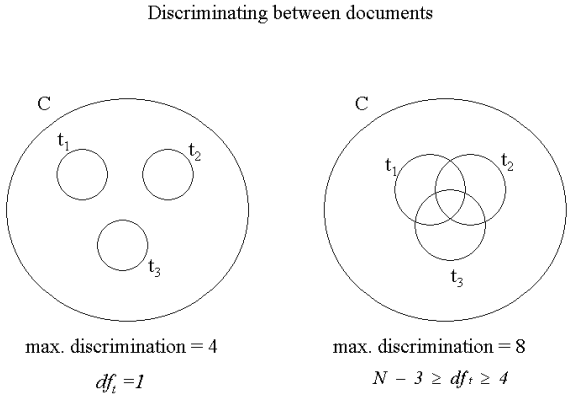

Fig. 4. Measuring the discriminating power of terms

identified. Then, documents containing these terms can be examined on a local level so as to distinguish these documents from each other as best as possible. This weakness with the *idf* measure is ammended by the global scheme presented here. The *cf* measure can be viewed as a high pass filter and *idf* as a low pass filter. The GP combines these in the correct way to create a scheme which will correctly identify terms of a high resolving power.

5 Conclusion

The cf/df measure presented in this paper has been independently found by evolutionary techniques to be advantageous in terms of average precision on general document collections. This measure further strengthens Luhn's hypothesis that middle frequency terms contain a higher resolving power. The measure is shown to increase average precision on general test collections over the standard solutions. The results presented here also suggest that the evolved schemes work for general natural language document collections.

In evaluating the success of this approach over similar approaches adopted in the past [11], it is worth noting that the main reason for the increase in performance is due to the inclusion of the collection frequency terminal. Future work will include experiments on larger document collections.

References

1. Darwin, C.: Chapter 4, The Origin of the Species by means of Natural Selection, or The Preservation of Favoured Races in the Struggle for Life. First Edition (1859)

2. Fan, W., Gordon, M. D. and Pathak, P.: A generic ranking function discovery framework by genetic programming for information retrieval. Information Processing and Management, in press (2004)
3. Goldberg, D. E.: Genetic Algorithms in Search, Optimisation and Machine learning. (1989)
4. Gordon, M.: Probabilistic and genetic algorithms for document retrieval. Communications of the ACM, 31(10): 1208-1218 (1988)
5. Greiff, W.R.: A theory of term weighting based on exploratory data analysis. In Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98), Melbourne, Australia, August 24-28 (1998)
6. Horng, J.T. and Yeh, C.C: Applying genetic algorithms to query optimization in document retrieval. Information Processing and Management: an International Journal, v.36 n.5 (2000) 737–759
7. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
8. Kuscu, I.: Generalisation and domain specific functions in Genetic Programming. Proceedings of the 2000 Congress on Evolutionary Computation CEC00, IEEE Press (2000) 1393–1400
9. Lewis, David D.: Feature Selection and Feature Extraction for Text Categorization. Proceedings of Speech and Natural Language Workshop (1992) 212–217
10. Luhn, H.P.: The automatic creation of literature abstracts. IBM Journal of Research and Development (1958) 159–165
11. Oren, N.: Re-examining tf.idf based information retrieval with Genetic Programming. Proceedings of SAICSIT (2002)
12. Porter, M.F.: An algorithm for suffix stripping. (1980)
13. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing & Management, 24(5) (1988) 513–523
14. Salton, G., Wong, A. and Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM, 18:613, 620 (1975)
15. Schultz, C.K.: H.P. Luhn: Pioneer of Information Science - Selected Works. Macmillan, London (1968)
16. Sparck-Jones, K.: A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 28 (1972) 11–21
17. Yu, C. T., Salton, G.: Precision weighting - An effective automatic indexing method. Journal of the ACM, 23(1) (1976) 76–88
18. Zipf, G. K.: Human Behaviour and the Principle of Least Effort. Addison-Wesley, Cambridge, Massachusetts (1949)