Using Genetic Programming to Evolve Weighting Schemes for the Vector Space Model of Information Retrieval

Ronan Cummins and Colm O'Riordan

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

Abstract. Term weighting in many Information Retrieval models is of crucial importance in the research and development of accurate retrieval systems. This paper explores a method to automatically determine suitable term weighting schemes for the vector space model. Genetic Programming is used to automatically evolve weighting schemes that return a high average precision. These weighting functions are tested on well-known test collections and compared to the *tf-idf* based weighting scheme using standard Information Retrieval performance metrics.

1 Introduction

The ability to retrieve information based on a user's need has become increasingly more important with the emergence of the World Wide Web and the huge increase in information available online. In spite of the success of the Web, problems with the retrieval of relevant information have emerged. The complexity of the problem is further increased due to the fact that much of this information appears in natural language and not in structured formats. Information Retrieval (IR) is primarily concerned with the retrieval of information rather than data. A very popular IR model used in recent years has been the vector space model [10].

It has been recognized that the effectiveness of vector space type models depend crucially on the term weighting applied to the terms in the document vectors [8]. These term weights are found by applying a weighting scheme based on the frequency of the term in the document. Terms that occur more often in a document are treated as more important, i.e. they better describe the document content, and thus are given a higher weight. Thus, the term weights in a document are calculated using measures of the terms in the document and collection. The number of ways these measures can be combined, even using a small number of operators (e.g. +, -, $\times$, and /) is vast. Genetic Programming [4], a stochastic searching algorithm based on Darwinian principles [1], is effective

for searching such large solution spaces. Thus, Genetic Programming is used to artificially breed a weighting scheme that achieves high average precision.

This paper presents a framework based on the vector space model that allows different weighting schemes to be evaluated. Genetic Programming is then utilised to allow selective breeding of fitter schemes. Background material in both the vector space model and Genetic Programming is introduced in the next section. Section three outlines the design and experimental design of the framework. Section four details the results and analysis for the experiments conducted. Finally, our conclusions are presented in the fifth section.

2 Background

2.1 Information Retrieval

This section presents background material for the vector space model. Traditional term weighting schemes are introduced and Luhn [5] and Zipf's [15] contribution to IR is summarised.

The Vector Space Model is one of the most widely known and studied IR models. The classic vector space model represents each document in the collection as a vector of terms with weights associated with each term. The weight of each term is based on the frequency of the term in that document 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 [8] is the most widely used weighting scheme 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 *max_freq* is the frequency of the most common term in the document. *max_freq* is used as the normalisation factor. The frequency is normalised since longer documents tend to have more terms and higher term frequencies. Due to the nature of IR algorithms, this would result in long documents being ranked as more relevant than short documents, when in fact this may not be true. The *idf* part of the weighting scheme, first introduced by Sparck Jones [13] is an Inverse Document Frequency measure. The main heuristic behind the *idf* factor is that a term that occurs infrequently is good at 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. Variations of these formulas exist but the motivation behind each remains the same. The weight for a term is then calculated using the formula:

$$w_t = tf \times idf_t \tag{3}$$

The standard matching function of the vector space model is the cosine measure [9];

$$sim(d_i, q) = \frac{\sum_{k=1}^{t}(w_{ik} \times q_k)}{\sqrt{\sum_{k=1}^{t}(w_{ik})^2 \times \sum_{k=1}^{t}(q_k)^2}} \tag{4}$$

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. This measures the cosine of the angles between the document and query in a multidimensional term space. The inner-product matching function is another common function and is the numerator of the cosine function.

Zipf [15] recognized that the frequency of terms in a collection, when placed in decreasing rank order, approximately follows a log curve. The Zipfian distribution for terms in a collection states that the product of the rank order of terms in a collection and their frequencies is approximately constant.

Luhn [5] further proposed that terms that occur too frequently have little power to distinguish between documents and terms that appear infrequently are also of little use in distinguishing between documents. Luhn used the Zipfian characteristics to devise 2 cut-off points for determining terms with a higher resolving power. Terms that appeared outside these points were considered as having a low distinguishing (resolving) power. Thus in *fig.1.*, the bell-shape in the graph relates the frequency of terms to their resolving power.

Salton and Yang [11] validate much of Luhn and Zipf's earlier work with empirical analysis. The standard *tf-idf* weighting scheme stems from these ideas. As previously discussed the *tf* part of the scheme identifies terms that appear more frequently as more important within a document (i.e. on a local level). This curve would follow the Zipfian curve in a local context. The *idf* part of this scheme weights the high frequency terms lower on a global scale. The *idf* feature increases the importance of terms that occur less often. Thus the curve of the *idf* measure is an inverse of the Zipfian curve in *fig.1.* Yu and Salton [14] suggest that the best distinguishing terms are terms which occur with a high frequency in certain documents but whose overall frequency across a collection is low (low document frequency). They conclude from this that a term weighting function should vary directly with term frequency and inversely with document frequency.

2.2 Genetic Programming

John Koza [4] founded Genetic Programming (GP) in the early 1990's. Inspired by the successes in traditional Genetic Algorithms (GA) by Holland [3] and

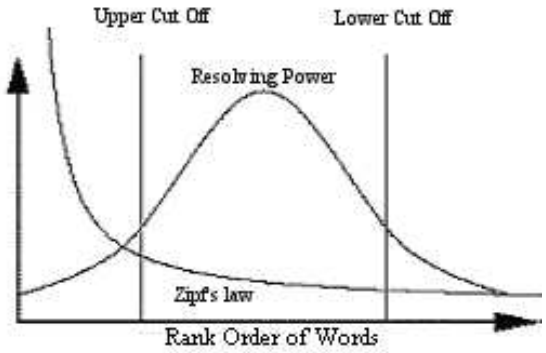

Fig. 1. Zipf's law and Luhn's proposed cut-off points [12]

Goldberg [2], the GP area has grown to help solve problems in a wide variety of areas. GP is based on the Darwinian theory of natural selection [1], where individuals that have a higher fitness will survive longer and thus produce more offspring. These offspring will inherit characteristics similar to their parents and through successive generations the useful characteristics will flourish. GP can be thought of as an artificial way of selective breeding.

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 where the roles of variables are not fully 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 encoded as trees in a GP. Each solution is rated based on how close it is to the perfect solution. 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 [2]. 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. For GP, the coded solution is a combination of operators and operands. This can be modelled in a tree like structure with operators on internal nodes and operands on the leaf nodes. These nodes are often referred to as genes and their values as

alleles. Once the recombination process is complete each individuals fitness in the new generation is evaluated and the selection process starts again. The algorithm usually ends after a certain number of generations or after an individual is found whose fitness is within a certain tolerance of the exact solution.

Genetic Programming has previously been used to evolve term weighting schemes for the vector space model. Research, conducted by Oren [6], has yielded relatively little success in the discovery of weighting schemes that outperform the traditional *tf-idf* on general document collections. However, a number of different avenues have yet to be explored in this area. These include the introduction of other document and query measures.

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) with a set of operators (e.g. +, -, $\times$, /). The fitness of these schemes are evaluated using the average precision obtained on a document test collection. The average precision, used as the fitness function, is calculated for each scheme by comparing the ranked list against the human determined relevant documents for each query. The system outputs the best scheme at each generation. 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. The documents and queries are pre-processed by removing standard stop-words and stemmed using Porter's stemming algorithm [7].

All experiments are run for 50 generations with an initial population of 1000. It was seen from prior experiments that when using the largest terminal and function set, the population converges before 50 generations. The experiments were trained on an entire collection and solutions were tested for generality on the collections that were not included in training.

The following tables show the function and terminal set used in our experiments:

Table 1. Terminal and function 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 Experimental Results and Analysis

4.1 Experiment one

The first experiment uses all of the document measures from *Table 1* and all of the operators from *Table 2*. The depth of the trees is initally set to 10. The following is the best formula evolved in the experiment:

$$w_t = ((cf/df) \times (sqrt((((((cf/df) \times N) \times ((df \times N)/(df + (df + tl))))/(df + (((cf/df) \times N)/(df + tl)))) + ((((cf/df) \times N) \times (cf/df))/(df + df))) \times ((((cf/df) \times N) \times (sqrt(rtf)))/(df + (df + (df + (tl + 1))))))))$$

Table 3. shows the average precision for the Medline training collection and the collections that were not included in training.

Table 3. Average precision for best solution on the Medline collection

Collection	*Docs*	*Qrys*	$tf - idf$	$best - evolved$
Medline	1033	30	49.04%	59.51%
Cranfield	1400	225	37.44%	41.45%
CISI	1460	76	20.74%	24.29%

The best solution found shows a 10.5% increase in average precision for the training set. It is also significant that the average precision increased when tested against all other document sets. The CISI collection shows an increased average precision of about 3.5%, while the Cranfield collection sees an increase of about 4%. The increase achieved is significant over the *tf-idf* scheme. The aim of the second experiment is to find out which measure or combination of measures leads to the increase in average precision seen.

4.2 Experiment two

Often a measure may have a beneficial but complex interaction with other measures which other standard techniques may not be able to uncover and which the GP will detect and, importantly, maximize. This experiment aims to discover which measure(s) is responsible for the increase in average precision over the *tf-idf* weighting scheme. The experiment starts with a limited terminal set, adds terminals one by one and examines how the average precision changes as the number of terminals increase. The GP parameters chosen are for the largest search space i.e. the full terminal set. Terminals are added in the order they appear in *Table 1.* Three runs of the GP for each terminal set is conducted and the

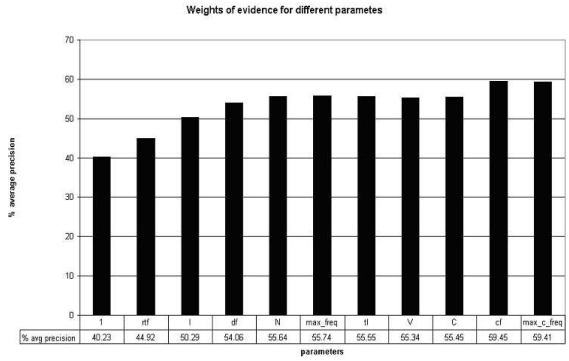

Fig. 2. weights of evidence for measures

best solution from the three runs is chosen and plotted. The Medline collection is the training set used in this experiment.

Fig. 2. shows that when the *cf* measure is added to the terminal set the average precision increases quite considerably. The experiment is run again with the *cf* measure added as the sixth terminal.

Fig. 3. shows the *cf* measure added as the sixth terminal. This shows that the combination of the *cf* measure with the original *tf-idf* measures are the most important measures in the terminal set for determining relevence.

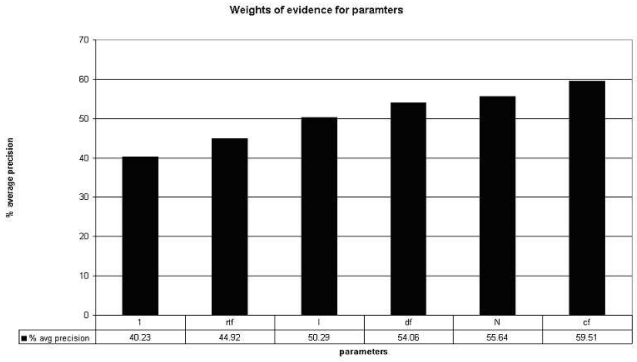

Fig. 3. weights of evidence for tf-idf and cf measures

From this experiment it can be seen that the *cf* is an important measure. We can also see that weighting schemes can be reduced to a combination of the *tf-idf* measures and the *cf* measure. In the first experiment the *cf/df* combination

appears in the best solution a number of times. This combination determines the average number of times a term appears in the documents in which that term appears. This measure is investigated in the next experiment.

4.3 Experiment three

This experiment uses only the global measures in the terminal set to evolve the global part of a weighting scheme to investigate the properties of the *cf* measure. Thus, the terminals used in this experiment are limited to *cf, df, 1* and *N*. The solutions were limited to a depth of 6. The following is the best solution evolved on the CISI collection using only these global measures:

$$gw_t = \frac{log(N/df)}{\sqrt{df}} \times log(\frac{cf}{df}) \times log(df) \tag{5}$$

Table 4. shows the average precision for the CISI training collection and the collections that were not included in training.

Table 4. Average precision for best global solution for the CISI collection

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

When the terms in the collection are placed in rank order, the gw_t weight of these terms is similar to that which Luhn predicted would lead to identifying terms with a high resolving power. *Fig 4.*, *Fig 5.* and *Fig 6.* show the terms placed in rank order and the gw_t weight for each term on the CISI training collection and the Medline and Cranfield collections which were not trained for.

An interesting point to note is that the global weighting schemes 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 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.

All being equal in a local context, the *idf* part of the *tf-idf* scheme will incorrectly identify terms with a low frequency as having a high resolving power.

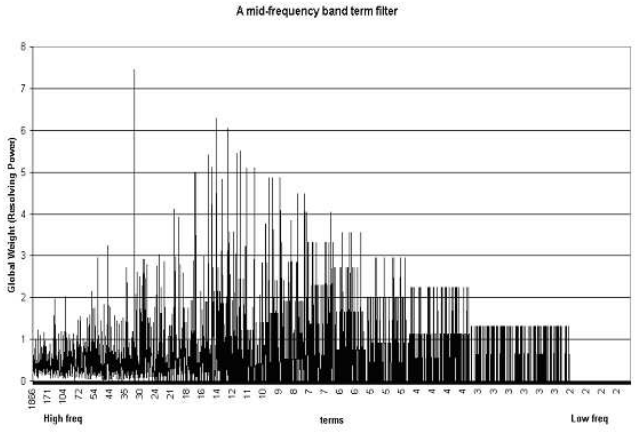

Fig. 4. gw_t for terms placed in rank order for the CISI collection

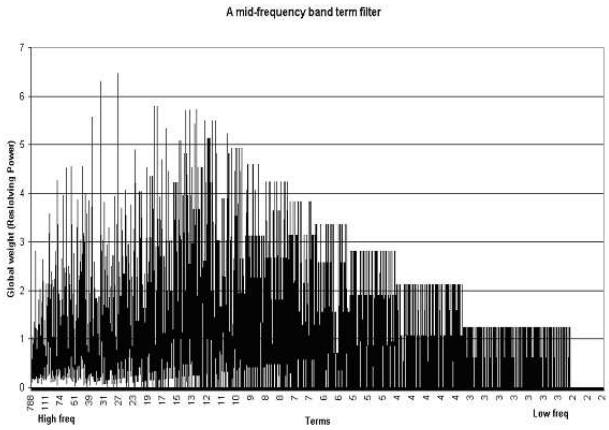

Fig. 5. gw_t for terms placed in rank order for the Medline collection

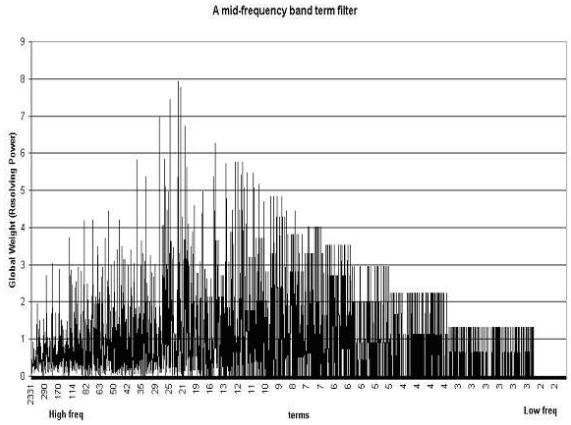

Fig. 6. gw_t for terms placed in rank order for the Cranfield collection

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 terms within the collection with a high resolving power must be 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. To amend this weakness within the *idf* weighting, the *cf* measure can be viewed as a high pass frequency filter which works on the global collection. By combining a high and low pass filter, a middle frequency band-pass term filter can be built, which will correctly identify terms of a high resolving power. This is in fact what the evolutionary technique adopted has discovered as the cf/df combination presents us with a global scheme that adheres to Luhn's theory.

5 Conclusion

The cf/df measure has been independently found by evolutionary techniques to be advantageous in terms of average precision on general document collections. The cf/df 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 *tf-idf* solution. The fact that all of the document sets see an increase in average precision over *tf-idf* also suggests 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, most notably that adopted by Oren [6], it is worth noting that the main reason for the increase in performance is due to the inclusion of the collection frequency terminal.

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] Goldberg, D. E.: Genetic Algorithms in Search, Optimisation and Machine learning. (1989)

[3] Holland, J.: Adaptation In Natural and Artificial Systems. The University of Michigan Press, Ann Arbour (1975) 159–165

[4] Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)

[5] Luhn, H.P.: The automatic creation of literature abstracts. IBM Journal of Research and Development (1958) 159–165

[6] Oren, N.: Re-examining tf.idf based information retrieval with Genetic Programming. Proceedings of SAICSIT (2002)

[7] Porter, M.F.: An algorithm for suffix stripping. (1980) 130–137

[8] Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing & Management, 24(5) (1988) 513–523

[9] Salton, G., McGill, M. J.: Introduction to Modern Information Retrieval. (1983) 123-125.

[10] Salton, G., Wong, A. and Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM, 18:613, 620 (1975)

[11] Salton, G., Yang, C.S.: On the specification of term values in automatic indexing. Journal of Documentation, 29 (1973) 351–372

[12] Schultz, C.K.: H.P. Luhn: Pioneer of Information Science - Selected Works. Macmillan, London (1968)

[13] Sparck Jones, K.: A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 28 (1972) 11–21

[14] Yu, C. T., Salton, G.: Precision weighting - An effective automatic indexing method. Journal of the ACM, 23(1) (1976) 76–88

[15] Zipf, G. K.: Human Behaviour and the Principle of Least Effort. Addison-Wesley, Cambridge, Massachusetts (1949)