An Analysis of Evolved Term-weighting schemes in Information Retrieval

Ronan Cummins
Dept. of Information Technology
National University of Ireland
Galway, Ireland
ronan.cummins@nuigalway.ie

ABSTRACT

Machine Learning techniques are increasingly being applied to many areas in Information Retrieval. Evolutionary computation and Genetic Programming in particular have been shown to be a viable alternative to other standard analytical methods for developing term-weighting schemes in IR. This paper presents term-weighting schemes that have been evolved in both a global (collection-wide) and local (within-document) context.

In particular, global term-weighting schemes are evolved which have characteristics similar to that which Luhn predicted would lead to identifying terms with a high resolving power. Local (within-document) weighting schemes are evolved dependent on the best performing global scheme and we show an increase in mean average precision over the BM25 scheme for the full evolved scheme (i.e. the combined local and global scheme). A term-frequency influence analysis of best performing within-document scheme is shown to behave similarly to that of *Okapi-tf* when its term-frequency influence parameter is assigned a low value.

The document normalisation part of the evolved local scheme does not perform as well as *Okapi-tf* on long documents. We conclude that *Okapi-tf* can be tuned to interact effectively with the evolved global weighting scheme presented and increase average precision over the standard BM25 scheme.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval, Retrieval models, Search process

General Terms

Algorithms, Experimentation

Keywords

Genetic Programming, Term-Weighting Schemes, Information Retrieval

1. INTRODUCTION

There have been many approaches applying evolutionary computation techniques to the domain of Information Retrieval (IR). Evolutionary computation techniques are proving to be a viable alternative to other standard analytical methods in many areas of IR. Genetic Algorithms and Genetic Programming [10] have been shown to be effective approaches to learning term-weights and term-weighting schemes in IR. These approaches, inspired by Darwin's theory of Natural Selection [2], are stochastic in nature and efficient for searching large complex search spaces.

Genetic algorithms have been used to modify document representations (a set of keywords) to aid in the retrieval of relevant documents [6, 24]. Genetic programming techniques have also been adopted to evolve weighting functions which outperform standard weighting schemes in a vector space framework [3, 13, 21]. However, in many of these approaches a critical analysis of the solutions evolved is not presented.

This paper outlines a Genetic Programming (GP) process which evolves term-weighting schemes in a vector space framework. Our research differs from other approaches in that the process is separated into two steps. Firstly, we evolve weighting schemes in a global domain which promote the best terms to use in distinguishing documents. Then, using the best global scheme, we evolve local schemes which use within-document measures to improve the mean average precision of the system. This process eases analysis of the evolved schemes and importantly reduces the search space considerably by separating the measures into their respective domains.

Section 2 introduces some IR background and some modern term-weighting approaches. The Genetic Programming approach is discussed in section 3. Section 4 details the experimental setup and benchmark schemes used. Results and analysis are presented in section 5 and finally our conclusions are summarised in section 6.

2. IR BACKGROUND

In this section we briefly outline modern term-weighting approaches and summarise some IR background relevant to this research.

2.1 Term-Weighting

The BM25 weighting scheme, developed by Robertson et al. [16], is a weighting scheme based on the probabilistic

model. *Okapi-tf* is calculated as follows:

$$Okapi\text{-}tf = \frac{rtf}{rtf + (k_1 \times ((1-b) + b \times \frac{dl}{dl_{avg}}))} \quad (1)$$

where rtf is the raw term frequency and dl and dl_{avg} are the length and average length of the documents respectively. k_1 is the term-frequency influence parameter and b is the document length influence tuning parameter. The *idf* of a term as determined in the BM25 formula is as follows:

$$idf_t = log(\frac{N - df_t + 0.5}{df_t + 0.5}) \quad (2)$$

where N is the number of documents in the collection and df_t is the number of documents a term appears in. The score of a document d in the BM25 ranking function can then be calculated as follows:

$$score(d, q) = \sum_{t \in q \cap d} (Okapi\text{-}tf \times idf_t \times qrtf) \quad (3)$$

where $qrtf$ is the raw term frequency in the query and t is a term in the query q and document d.

Another modern matching function is the pivoted document length normalisation scheme. The score of a document in this scheme is calculated as follows [19]:

$$\sum_{t \in q \cap d} (\frac{1 + log(1 + log(rtf))}{(1-s) + s\frac{dl}{dl_{avg}}} \times log(\frac{N+1}{df}) \times qrtf) \quad (4)$$

where s is the slope and is a constant value of usually about 0.2.

2.2 Resolving Power of Significant Terms

Luhn [12] proposed 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 Figure 1, the bell-shaped curve relates the frequency of terms to their distinguishing (resolving) power. Salton et al. [17, 18] validate

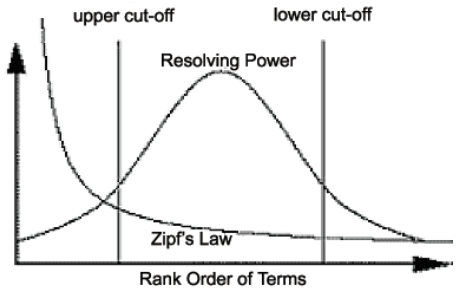

Figure 1: Resolving Power of Significant Terms

much of Luhn's work with empirical analysis. They arrive at similar conclusions to that of Luhn; that middle frequency terms are the most useful in terms of retrieval. Low frequency terms are, on average, poor discriminators while high frequency terms are the least useful [17]. It can been observed that *idf* is not fully consistent with Luhn's theory that the resolving power of terms with a low frequency is also low. Greiff [7] has also indicated that a flattening of *idf* at a low frequency level would lead to an increase in performance.

3. GENETIC PROGRAMMING

This section introduces and summarises the Genetic Programming process.

3.1 Basic Algorithm

Genetic Programming [10] is a heuristic stochastic searching algorithm, inspired by natural selection [2], and is efficient for navigating large complex search spaces. In the GP process, a population of solutions is created randomly. The solutions are encoded as trees and can be thought of as the genotypes of the individuals. Each tree (genotype) contains nodes which are either functions (operators) or terminals (operands). The values on the nodes of each tree are referred to as alleles. Each solution is rated based on how it performs in its environment. This is achieved using a fitness function. Having assigned the fitness values, selection can occur. Individuals are selected for reproduction based on their fitness value. Fitter solutions will be selected more often.

Once selection has occurred, reproduction can start. 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. Mutation (asexual reproduction) is the random change of allele of a gene to create a new individual. Selection and recombination occurs until the population is replaced by newly created individuals. Once the recombination process is complete, each individual's fitness in the new generation is evaluated and the selection process starts again. The process usually ends after a predefined number of generations, or until convergence of the population is achieved or after an individual is found with an acceptable fitness.

3.2 Selection

Tournament selection is one of the most common selection method used. In tournament selection, a number of solutions are chosen at random from the population and these solutions compete with each other. The fittest solution is then chosen as a parent. The number of solutions chosen to compete in the tournament is called the tournament size and this can be increased or decreased to increase or decrease the speed of convergence.

3.3 Reproduction

Crossover is the main reproductive mechanism in GP. When two solutions are selected from the selection process, their genomes are combined to create a new individual. An example of crossover can be seen in Figure 2.

3.4 Fitness Function

The mean average precision (MAP), used as the fitness function, is calculated for each scheme by comparing the ranked list returned by the system for each query expansion scheme against the human determined relevant documents for each query. Mean average precision is calculated over all points of recall and is frequently used as a performance measure in IR systems as it provides a measure of both the accuracy and recall of the retrieval system.

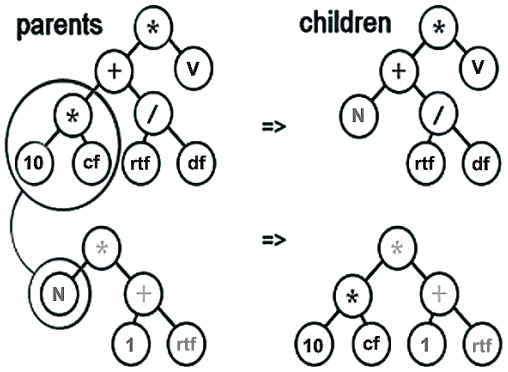

Figure 2: Example of crossover in GP

4. EXPERIMENTAL SETUP

The following section introduces the terminal and function sets, document test collections, benchmark term weighting schemes and GP parameters used in our experiments.

4.1 Functions and Terminals

Tables 1 and 2 show the global and local terminal sets respectively. Table 3 shows the function set.

Table 1: Global Terminal Set

Terminal	Description
1	*the constant* 1
10	*the constant* 10
N	no. of documents in the collection
df	no. of documents a term occurs in
cf	no. of times a term occurs in the collection
V	no. of unique terms in the collection
C	total no. of words in the collection
0.5	*the constant* 0.5

Table 2: Local Terminal Set

Terminal	Description
1	*the constant* 1
10	*the constant* 10
rtf	raw term frequency
l	length of document vector
tl	total number of words in document
max_freq	frequency of most common term in document
l_{avg}	average length of document vector
0.5	*the constant* 0.5

Table 3: Function Set

Function	Description
+, ×, /, -	standard arithmetic functions
log	the natural log
$\sqrt{}$	square-root function
sq	square

4.2 Document Test Collections

We use the OHSUMED[1] collection [8] for training and evaluation. We divide the OHSUMED collections into collections of various sizes. Our training data consists of the first half of the 1988 documents (around 35,000 documents). We use the description field of the 63 OHSUMED topics for our queries. The relevance assessments for the OHSUMED collection are graded as *definitely* or *possibly* relevant which we regard as relevant.

We also use collections from TREC disks 4 and 5, and a set of 50 TREC topics on each of these collections. For each set of 50 topics we create a short query set which consists of only the title fields of the topics, a medium length query set which consists of the title and description fields, and a long query set which consists of the title, description and narrative fields. We ignore topics that have no relevant documents associated with them. The documents and queries are pre-processed by removing standard stop-words from the Brown Corpus[2] and are stemmed using Porter's stemming algorithm [15]. Table 4 shows some characteristics of the test collections used in this research.

Table 4: Characteristics of Document Collections

Collection	# Docs	length	# Topics	length
TRAIN'88	35,412	72.7	0-63	4.97
OHSU'88	70,825	75.2	0-63	4.97
OHSU'89	74,869	76.9	0-63	4.97
OHSU'90-91	148,162	81.4	0-63	4.97
LATIMES	131,896	251.6	301-350 (short)	2.42
			301-350 (med)	9.92
			301-350 (long)	29.86
FBIS	130,471	249.9	351-400 (short)	2.42
			351-400 (med)	7.88
			351-400 (long)	21.98

4.3 GP Parameters

All tests are run for 50 generations with an initial random population of 100 solutions. It is seen from sample tests that convergence occurs before 50 generations. The tournament size is set to 4. The depth of the solutions is limited to 6 to improve the generality of the solutions while allowing a suitably large solution space to be searched. The creation type used is the standard ramped half and half creation method used by Koza [10]. 4% mutation is used in our experiments. Due to the stochastic nature of GP a number of runs is often needed to allow the GP converge to a suitably good solution. We run each test 4 times and choose the best general solution from those runs. We test the best performing solution from each run for generality on the OHSU'88 collection. We then choose the best general solution to include in this paper. A run takes about 24 hours on a 2.0 GHz processor with 500Mbs of RAM. The training set used for *all* experiments is the TRAIN'88 collection.

4.4 Benchmark Schemes

The query weighting scheme used for the terms in the query is simply the raw term-frequency of the term in the query. The benchmark scheme used for the BM25 scheme

[1]http://trec.nist.gov/data/t9_filtering.html
[2]http://www.lextek.com/manuals/onix/stopwords1.html

has default values of k_1=1.2 and b=0.75 [23]. Suggested values for k_1 range between 1.0 and 2.0 [16]. Table 5 shows the performance of the default tuning parameters and the performance when k_1=2 as this is another commonly used value. For all the schemes which use *Okapi-tf* in this research b was set to 0.75 as default. The pivoted normalisation scheme (*Piv*) with a default value for s of 0.2 is also shown in Table 5. The BM25 scheme with k_1=1.2 was found to be the best performing scheme and is used in empirical comparisons against our evolved schemes in the results and analysis section.

Table 5: % MAP for Benchmarks

Collection	Topics	BM25		Piv
		k_1=2.0	k_1=1.2	
TRAIN'88	0-63	21.10	**23.25**	21.57
OHSU'88	0-63	30.98	**32.79**	31.08
OHSU'89	0-63	29.00	**30.69**	30.07
OHSU'90-91	0-63	27.56	**28.08**	26.28
LATIMES	301-350 (short)	23.32	**24.18**	24.26
	301-350 (med)	24.29	**25.61**	25.48
	301-350 (long)	25.84	**25.82**	24.56
FBIS	351-400 (short)	16.34	**17.55**	15.90
	351-400 (med)	18.49	**19.53**	17.92
	351-400 (long)	19.60	**21.84**	18.41

4.5 Approach Adopted

The underlying framework adopted here is similar to those adopted previously [4, 13, 20]. However, there are some fundamental differences in the aim of the experiments. While others have evolved full weighting schemes and have shown an increase in average precision over standard weighting schemes, reasons for the increase are not presented and are difficult to analyse. This two step process reduces the size of the search space and also provides a means of analysis against standard *tf-idf* type solutions.

Firstly, we evolve the global weight (gw_t) with a binary weighting on the local (within-document) weighting (lw_t) using terminals from Table 1 and functions from Table 3. This is done so that significant terms, i.e. terms which aid retrieval, are promoted. Then, we evolve a local weighting using Tables 2 and 3 while keeping the previously evolved global weight constant. The full evolved term-weighting scheme can then be identified as follows:

$$evol_t(d_i, q) = \sum_{t \in q \cap d} (lw_t \times gw_t \times qrtf) \quad (5)$$

where $qrtf$ is the raw term frequency in the query as used in the benchmark schemes. The best evolved solutions presented in the results section in this paper are simplified and re-written to aid the readability and analysis of the weighting schemes. However, the weighting schemes presented are functionally equivalent to those output by the GP process.

5. EXPERIMENTAL RESULTS

The first experiment evolves global schemes and an analysis of the best performing scheme is presented. The second experiment evolves local schemes while keeping the global scheme constant and further analysis is presented.

5.1 Global Schemes

The following is the best global scheme evolved after 4 runs of the GP using Tables 1 and 3 as our terminal and function set:

$$gw_t = log(\frac{cf}{df}) \times \sqrt{\frac{N}{df} \times (\frac{1}{df} + 1)} \quad (6)$$

Table 6: % MAP for idf and gw_t

Collection	Topics	idf	gw_t	%increase
TRAIN'88	0-63	19.22	22.10	14.98
OHSU'88	0-63	25.89	27.85	07.57
OHSU'89	0-63	25.22	27.63	09.55
OHSU'90-91	0-63	21.72	25.03	15.23
LATIMES	301-350 (short)	17.91	18.90	05.53
	301-350 (med)	19.04	23.49	23.37
	301-350 (long)	13.79	24.78	79.64
FBIS	351-400 (short)	11.25	11.77	04.62
	351-400 (med)	10.42	14.55	39.63
	351-400 (long)	06.97	14.08	102.01

We can see from Table 6 that the MAP of the gw_t weighting is higher than that of idf on the training collection and also on all of the collections not included in training. These increases are in the range of about 0.5% to 11% MAP. It is also worth noting that the size of the training collection is significantly large enough to allow us to learn a general scheme. Figure 3 shows an average precision histograms for each query for the OHSU'89 collection. We found that although overall mean average precision increased, certain queries in particular performed worse. An example of this can be seen in Figure 3 for the OHSU'89 collection. Many of these poor performing queries (query 12 in particular) contain terms whose collection frequency is equal to the document frequency (cf=df) and whose concentration is low. These terms are assigned a zero weighting (effectively being eliminated) because of the $log(cf/df)$ part of the gw_t scheme.

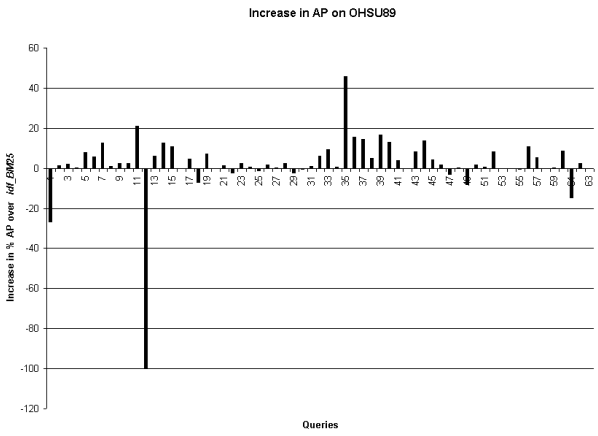

Figure 3: AP Histogram for gw_t vs idf for each query

To ammend this weakness in the scheme, we evolved a further factor (C_1) using a population of 100 for 50 generations, using Tables 1 and 3, dependent on the gw_t weight so

that certain terms would not be asssigned a weight of zero. The following formula shows how the evolved C_1 factor fits into the evolved global weighting:

$$gw2_t = log(\frac{cf + C_1}{df}) \times \sqrt{\frac{N}{df} \times (\frac{1}{df} + 1)} \quad (7)$$

$$C_1 = \frac{0.5}{\sqrt{\sqrt{cf}}} = \frac{0.5}{cf^{\frac{1}{4}}} \quad (8)$$

In Table 7, we see that the new global evolved scheme ($gw2_t$) is equal to or better than the original evolved gw_t scheme in terms of mean average precision on many of the collections. It is worth mentioning that the reason certain low concentration terms are assigned a zero weight in the original scheme is because they are of little benefit in the training set. However, as seen on the validation test collections assigning them some weight is beneficial for certain queries as the MAP increases again on some collections, although often only slightly. Query 12 from OHSU'89 (Figure 3) performs similarly to *idf* after the modification and is responsible for the larger increase on that collection. It would seem that in general these terms are of little benefit as the MAP does not rise significantly. However, completely eliminatimg them can lead to certain documents being left irretrievable.

Table 7: % MAP for *idf* and $gw2_t$

Collection	Topics	*idf*	$gw2_t$	%increase
TRAIN'88	0-63	19.22	22.28	15.92
OHSU'88	0-63	25.89	28.59	10.43
OHSU'89	0-63	25.22	29.23	15.78
OHSU'90-91	0-63	21.72	25.03	15.23
LATIMES	301-350 (short)	17.91	19.05	06.37
	301-350 (med)	19.04	23.31	22.43
	301-350 (long)	13.79	24.24	75.78
FBIS	351-400 (short)	11.25	11.73	04.26
	351-400 (med)	10.42	14.56	39.73
	351-400 (long)	06.97	14.09	103.19

5.1.1 Global Scheme Analysis

The global scheme increases mean average precision over the *idf* measure because the cf/df part of the global weighting (5) can be viewed as the average within-document frequency and has previously been used by Kwok [11] and Pirkola [14]. It is interesting that this measure has been found independently by evolutionary techniques. This measure (or a slight variation of it as in $gw2_t$) will increase the average precision in a global context as we have a rough estimate as to how many times a term will appear in the documents in which that term appears. It is important to note that the *cf* measure has information not available from other measures and can lead to re-ordering of documents. Figure 4 shows the terms in the OHSU'88 collection placed in rank order and the *idf* of each term assigned. Figure 5 shows the terms in the OHSU'88 collection placed in rank order and the $gw2_t$ weighting (6) applied.

Terms of a low concentration (i.e. where cf=df) are assigned the lowest weight for any specific document frequency. This scheme weights terms that tend to be concentrated higher than those that tend to be more dilute. It has previously been shown on small collections that term-weighting schemes that contain the *cf* measure can achieve a higher average precision than those that do not contain it [1]. Traditional *tf-idf* type schemes do not have any measure of the concentration of a term because within-document term-frequencies cannot change the global weight of a term. If a term tends to occur many times in a document, it is likely that it describes some aspect of that document. In a *tf-idf* type scheme, the fact that a term has described another document is not used to increase the weight of that term in other documents.

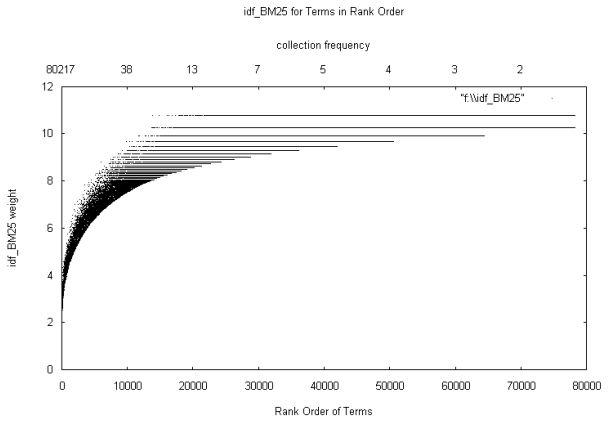

Figure 4: *idf* for terms in Rank order (OSHU'88)

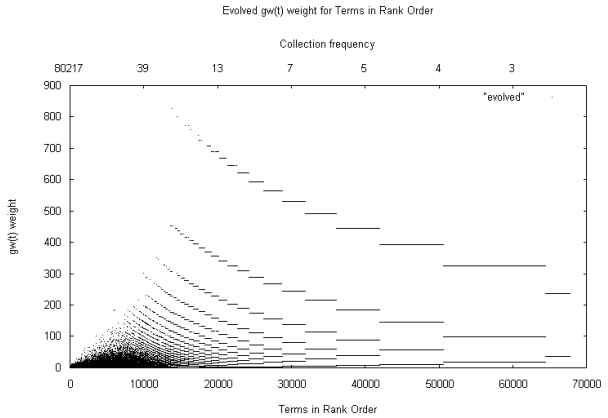

Figure 5: $gw2_t$ for terms in Rank order (OSHU'88)

5.2 Local Schemes

The local scheme identified (9) is the best local scheme found on the training collection after 4 runs of the GP using tables 2 and 3 as the terminal and function set. This local weight is evolved dependent on the global scheme ($gw2_t$) evolved in the previous experiment.

$$lw_t = \sqrt{(1 + \frac{1}{log(l)}) \times (1 + \frac{log(rtf)}{log(l)})} \quad (9)$$

From table 8 we can see that the complete evolved solution ($evol_t$) has a higher MAP than that of BM25 which ranges from 1.4% to 2.2% on the OHSUMED subset collections not

included in training. The FBIS collection shows a large decrease in MAP when used with medium and long queries. The MAP on the LATIMES collection shows very little improvement using the $evol_t$ weighting. The main difference between the OHSUMED type collections and the FBIS and LATIMES collections is that the document lengths of the latter are significantly longer. As the local weighting is evolved on the OHSUMED data it would seem that the document normalisation learned is not generalisable for all document lengths. We will investigate whether this is true in the next section.

Table 8: % MAP for $BM25_{k_1=1.2}$ and $evol_t$

Collection	Topics	$BM25$	$evol_t$	%increase
TRAIN'88	0-63	23.25	25.28	08.73
OHSU'88	0-63	32.79	33.90	03.38
OHSU'89	0-63	30.69	32.70	06.54
OHSU'90-91	0-63	28.08	29.81	06.16
LATIMES	301-350 (short)	24.18	24.67	02.02
	301-350 (med)	25.61	25.82	00.82
	301-350 (long)	25.82	26.86	04.02
FBIS	351-400 (short)	17.55	20.96	19.43
	351-400 (med)	19.53	18.75	-03.99
	351-400 (long)	21.84	17.09	-21.75

5.2.1 Term-Frequency Analysis

As the local scheme evolved (lw_t) contains only term-influence and normalisation parameters like those that already appear in the *Okapi-tf* scheme and in order to verify that the document normalisation of the evolved local scheme can be improved, we attempt to tune the term-frequency influence of *Okapi-tf* scheme so that it is similar to our lw_t scheme and simply use the default *Okapi-tf* document normalisation aspect.

For the analysis of the term-frequency influence in this section, we assume an average length document (i.e. $dl = dl_{avg}$). To compare *Okapi-tf* to other local weighting formula in terms of term-frequency influence, it is neccessary to calculate the relative increase in weight as the raw term-frequency increases. We calculate the relative increase in weight using the actual increase in weight seen as the term-frequency increases by 1, divided by the previous weight.

$$\Delta w(rtf) = \frac{w(rtf) - w(rtf-1)}{w(rtf-1)} \tag{10}$$

where $rtf \geq 2$ and $\Delta w(rtf)$ is the relative change in weight from $w(rtf-1)$. Figure 6 shows this relative increase in weight given to higher term frequencies. The relative term-frequency influence decreases as the term-frequency increases for all schemes as expected. We can see in this diagram that a value of 0.2 for k_1 in *Okapi-tf* closely relates to the term-frequency influence of the evolved scheme.

To empirically test the term-frequency influence we combine our $gw2_t$ evolved weight with *Okapi-tf* when k_1=0.2 in place of the lw_t scheme. We will call this combination scheme *ok-*$gw2_t$. Table 9 shows the MAP for the *ok-*$gw2_t$ weighting scheme. We can see that the *ok-*$gw2_t$ weighting performs better than $evol_t$ on OHSU'89 and OHSU'90-91 and worse on OHSU'88. The differences between them are small but are still higher than those of the original BM25 scheme. However on the longer documents the MAP has increased significantly on the medium and long queries.

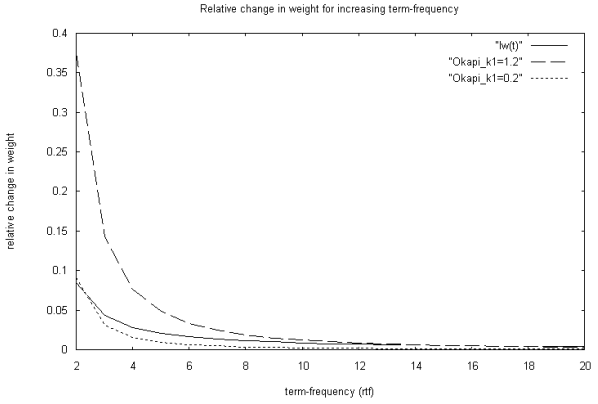

Figure 6: Relative increase in weight for lw_t and *Okapi-tf* when k_1=1.2 and k_1=0.2

Table 9: % MAP for $BM25_{k_1=1.2}$ and *ok-*$gw2_t$

Collection	Topics	$BM25$	*ok-*$gw2_t$	%increase
TRAIN'88	0-63	23.25	**25.14**	08.12
OHSU'88	0-63	32.79	**34.26**	04.48
OHSU'89	0-63	30.69	**33.56**	09.35
OHSU'90-91	0-63	28.08	**30.43**	08.36
LATIMES	301-350 (short)	24.18	**25.24**	04.38
	301-350 (med)	25.61	**29.68**	15.89
	301-350 (long)	25.82	**30.60**	18.51
FBIS	351-400 (short)	17.55	**19.29**	09.91
	351-400 (med)	19.53	**22.02**	12.75
	351-400 (long)	21.84	**25.70**	17.67

The results suggest that *Okapi-tf* can be tuned to interact with the evolved global weighting ($gw2_t$). The reason that a lessening of the term-frequency influence of the *Okapi-tf* scheme seems to be sucessful when combined with the $gw2_t$ evolved weighting is due to the fact that the collection frequency (cf) appears in the global weighting. As a result, certain frequency information is already used in a global context (albeit often a rough estimation). This collection frequency information is an accurate portrayal of the within-document term frequency at low document frequencies. As some average frequency information is available prior to developing a within-document weighting, the weight assigned to within-document term frequencies should be reduced. The best scheme identified is a combination of *Okapi-tf* and $gw2_t$.

6. CONCLUSIONS

We have shown that weighting schemes that have new properties can be found using genetic programming ($gw2_t$) as well as weighting scheme which have similar properties to that which already exist (lw_t). The most effective scheme identified is a combination of *Okapi-tf* and our evolved global scheme. The increase in mean average precision is also consistent over the various collections tested to date.

For future work we will investigate evolving query expan-

sion schemes using genetic programming. We propose using genetic programming to find and weight expansion terms in both local and global query expansion frameworks. For example in local expansion techniques, a number of top ranked documents are deemed relevant. Using genetic programming we can use the characteristics of a term in these top ranked documents and the characteristics of the term in the entire collection to evolve a selection value for potential expansion terms. These terms can then be added to the original query with a weight based on their selection value. This process can be used to find good term selection schemes and also provide a correct weight for the expansion term automatically.

7. ACKNOWLEDGMENTS

This work is being supported by IRCSET (the Irish Research Council for Science, Engineering and Technology) under the Embark Initiative.

8. REFERENCES

[1] Ronan Cummins and Colm O'Riordan. Using genetic programming to evolve weighting schemes for the vector space model of information retrieval. In Maarten Keijzer, editor, *Late Breaking Papers at the 2004 Genetic and Evolutionary Computation Conference*, Seattle, Washington, USA, 26 July 2004.

[2] Charles Darwin. *The Origin of the Species by means of Natural Selection, or The Preservation of Favoured Races in the Struggle for Life*. First edition, 1859.

[3] Weiguo Fan, Michael D. Gordon, and Praveen Pathak. A generic ranking function discovery framework by genetic programming for information retrieval. *Information Processing & Management*, 2004.

[4] Weiguo Fan, Michael D. Gordon, Praveen Pathak, Wensi Xi, and Edward A. Fox. Ranking function optimization for effective web search by genetic programming: An empirical study. In *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 4*, page 40105. IEEE Computer Society, 2004.

[5] Weiguo Fan, Ming Luo, Li Wang, Wensi Xi, and Edward A. Fox. Tuning before feedback: combining ranking discovery and blind feedback for robust retrieval. In *Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 138–145. ACM Press, 2004.

[6] M. Gordon. Probabilistic and genetic algorithms in document retrieval. *Commun. ACM*, 31(10):1208–1218, 1988.

[7] W.R. Greiff. 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 1998.

[8] William Hersh, Chris Buckley, T. J. Leone, and David Hickam. Ohsumed: an interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 192–201. Springer-Verlag New York, Inc., 1994.

[9] J.T. Horng and C.C Yeh. Applying genetic algorithms to query optimization in document retrieval. *Information Processing & Management*, 36(5):737–759, 2000.

[10] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[11] K. L. Kwok. A new method of weighting query terms for ad-hoc retrieval. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 187–195. ACM Press, 1996.

[12] H.P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, pages 159–165, 1958.

[13] N. Oren. Re-examining tf.idf based information retrieval with genetic programming. *Proceedings of SAICSIT*, 2002.

[14] A. Pirkola and K. Jarvelin. Employing the resolution power of search keys. *J. Am. Soc. Inf. Sci. Technol.*, 52(7):575–583, 2001.

[15] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[16] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. Okapi at TREC-3. In *In D. K. Harman, editor, The Third Text REtrieval Conference (TREC-3) NIST*, 1995.

[17] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

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

[19] A. Singhal. Modern information retrieval: A brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24(4):35–43, 2001.

[20] Andrew Trotman. An artificial intelligence approach to information retrieval (abstract only). In *SIGIR*, page 603, 2004.

[21] Andrew Trotman. Learning to rank. *Information Retrieval*, 8:359 – 381, 2005.

[22] Dana Vrajitoru. Crossover improvement for the genetic algorithm in information retrieval. *Inf. Process. Manage.*, 34(4):405–415, 1998.

[23] Steve Walker, Stephen E. Robertson, Mohand Boughanem, Gareth J. F. Jones, and Karen Sparck Jones. Okapi at trec-6 automatic ad hoc, vlc, routing, filtering and qsdr. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the 6th Text REtrieval Conference (TREC-6)*, pages 125–136. NIST Special Publication 500-240, 1997.

[24] Jing-Jye Yang and Robert Korfhage. Query optimization in information retrieval using genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 603–613. Morgan Kaufmann Publishers Inc., 1993.