Using Genetic Algorithms for Query Reformulation

José R. Pérez-Agüera
Dept. Software Engineering and AI
Facultad de Informática
C/ Profesor Jose García Santesmases
University Complutense of Madrid
jose.aguera@fdi.ucm.es

Abstract: Nowadays, searching information in the web or in any kind of document collection has become one of the most frequent activities. However, user queries can be formulated in a way that hinders the recovery of the requested information. The objective of automatic query transformation is to improve the quality of the recovered information. This paper describes a new genetic algorithm used to change the set of terms that compose a user query without user supervision, by complementing an expansion process based on the use of a morphological thesaurus. We apply a stemming process to obtain the stem of a word, for which the thesaurus provides its different forms. The set of candidate query terms is constructed by expanding each term in the original query with the terms morphologically related. The genetic algorithm is in charge of selecting the terms of the final query from the candidate term set. The selection process is based on the retrieval results obtained when searching with different combination of candidate terms. The algorithm shows improvement over some other using standard collections.

Keywords: *Genetic algorithm, information retrieval, query reformulation.*

1. GENETIC ALGORITHMS AND QUERY REFORMULATION

There is an underlying common background in many of the works done in query reformulation, namely the appropriate selection of a subset of search terms among a list of candidate terms. This is a special case of the problem of *weighting* the candidates, where we allow only binary weight values.

The number of possible subsets grows exponentially with the size of the candidate set. Furthermore, we cannot evaluate a priori the quality of a subset with respect to another one: this depends on the (unknown) relevance of the documents in the collection. Finally, the subset selection will depend not only on the original term but also on the overall query: an expansion that is correct in one query context may be incorrect in another. For these reasons, standard optimisation techniques cannot be applied to this problem. Instead, we must resort to heuristic optimisation algorithms that sample the space of possible subsets and predicts their quality in some unsupervised manner.

Genetic algorithms are the best fitted optimisation techniques for this setting; they have been previously applied with success in a number of expansion contexts [5]. They can be divided in two groups: relevance feedback techniques and Inductive Query by Example (IQBE) algorithms. Systems based on relevance feedback modify the original query taking into account the user relevance judgements on the documents retrieved by this original query. IQBE, proposed in [3], is a process in which the user provides document examples and the algorithms induce the key concepts in order to find other relevant documents. In this way the user does not provide a query but a set of relevant documents. The system applies an off-line learning process to automatically generate a query describing the user's needs.

Several works apply GAs to assigning weights to the query terms [17,23,19,10,9], while others are devoted to selecting the query terms. Let us review some proposals in the latter case, the one on which we focus our work. Chen et al. [3] apply a GA as an IQBE technique, i.e. to select the query terms from a set of relevant documents provided by the user. In this work, the authors propose an individual representation that has also been used in later works: chromosomes are binary vectors of fixed size in which each position is associated with one of the candidate terms. In [11], the authors propose a genetic programming (GP) algorithm to learn boolean queries encoded as trees whose operators are AND, OR and NOT. This work was later extended in [6] by incorporating multiobjective optimization techniques to the problem. Fernández-Villacañas and Shackleton [12] compared two evolutionary IQBE techniques for boolean query learning, one based on GP and the other on a classic binary GA, which obtained the best results. Kraft et al. [15] propose the use of GP to learn fuzzy queries. The queries are encoded as expression trees with boolean operators as inner nodes, and query terms with weights as terminal nodes. Cordón et al. [4] extend Kraft's proposal by applying a hybrid simulated annealing-genetic programming scheme, what allows them to use new operators to adjust the term weights. Tamine et al. [20] use knowledge-based operators instead of the classical blind operators, as well as niching techniques.

A common factor of the above mentioned works is that they relay on some kind of information provided by the user. In some cases, the user has to provide a set of documents that are used for the inductive learning of terms. In other cases, the user provides relevance judgements on the retrieved documents, that are use to compute the fitness.

In this work we propose a new application of GAs for the selection of query terms. The novelty is that our system does not require any user supervision: new candidate terms for the query are provided by a morphological thesaurus generated using Porter stemmer. In this way we will *learn* which term variants should be included in the expansion and which should not.

2. OUR GENETIC ALGORITHM

Chromosomes of our GA are fix-length binary strings where each position corresponds to a candidate query term. A position with value one indicates that the corresponding term is present in the query. Because some preliminary experiments we have performed have shown that, in most cases, the elimination of the original query terms degrades the retrieval performance, we force to maintain them among the selected terms of every individual. The set of candidate terms is composed of the original query terms, along with related terms provided by the applied thesaurus. Each term of the original query is grouped with the expanded terms related to it, and this set (*term_set*) [16] is submitted as an individual query. The weights assigned to the documents retrieved with each *term_set* are used to sort the total set of retrieved documents.

The selection mechanism uses roulette wheel. We apply one-point crossover operator and random mutation [7,8,13]. We also apply elitism. The fitness function used is some measure of the degree of similarity between a document belonging to the system and the submitted query. We will discuss this further in the different experiments.

3. TESTING THE VIABILITY OF THE APPROACH

Our first experiments have been aimed at testing how much the GA can improve the results of the search, assuming a good fitness function can be found. For this purpose, we take used as fitness directly the user relevance judgements; this is in general unknown, but it allows us to test the approach. Specifically, we have used as fitness function the standard precision measure defined as the fraction of retrieved documents (the set A) which are relevant (R_a).

$$\text{Precision} = \frac{|Ra|}{|A|}$$

Table 1 shows the precision of the best individual obtained by the GA. We have compared our system performance with the results of the original user query (*Baseline*) and with the results obtained expanding with the stems provided by the Porter stemming (*Porter Stemming*). The latter works by substituting the original query terms by their stems and performs the search in the document collection indexed by stems. We can observe that our system achieved an important improvement of the performance, greater than the one achieved with Porter. Figure 1 shows the evolution of two queries of the test set. We can observe that both of them reach convergence very quickly.

	Prec.	Prec10	Improvement	
Baseline	0.3567	0.4150	-	
Porter Stem.	0.4072	0.45	+12.40%	
Genetic Stem.	0.4682	0.54	+23.81%	

Table 1: Global precision results for the whole set of tested queries. Each individual datum has been computed as the average over 5 different GA runs. Prec. stands for precision (all documents), Prec10 stands for the precision of the results for the first ten documents retrieved. Last column is the rate of precision (all documents) improvement.

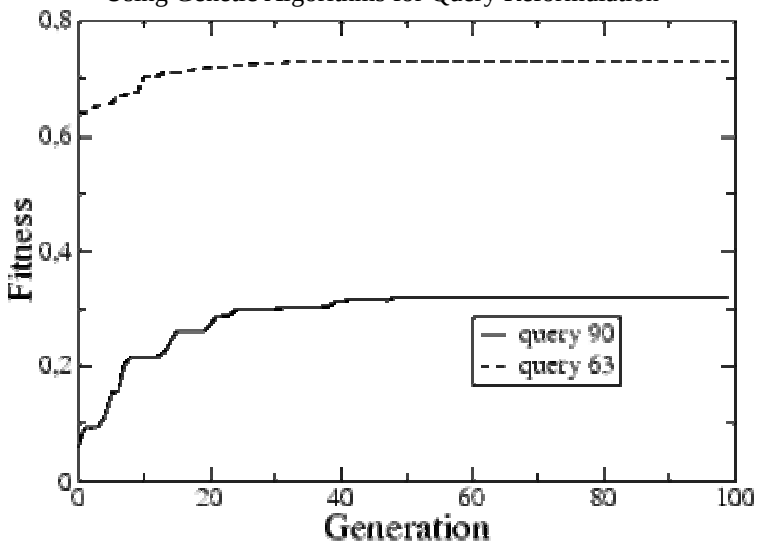

Figure 1: Fitness evolution for two queries of the test set. The GA parameters have been a population size of 100 individuals, a crossover rate of 25% and a mutation rate of 1%.

4. EXPERIMENTAL RESULTS

The system has been implemented in Java, using the JGAP library, on a Pentium 4 processor.
We have carried out experiments to evaluate the fitness functions considered. We have investigated the best range of the GA parameters. Finally, we provide global measures for the whole set of queries that we have considered. The collection used is EFE94, a spanish documents collection from CLEF (Cross Language Evaluation Forum) and the relevance assessments used are from 2001 year (queries 40-90).

5. SELECTING THE FITNESS FUNCTION

In principle we would like to use Average precision as the fitness function. However, this is not known at query time. Instead, it has been suggested in previous work to use the document scores as the fitness. While this may not be intuitive, it turns out that variations of these scores after expansion are correlated with relevance. One intuitive explanation would be that adding an unrelated term to a query will not bring in new document with high scores, since it is unlikely that it will retrieve new documents; on the other hand adding a term that is strongly related to the query will bring new documents that also contain the rest of the terms of the query and therefore it will obtain high scores.
We have considered three alternative fitness functions, $\sqrt{\cos\theta}$ $\cos\theta$ and . $\cos^2\theta$ To select the fitness function to be used in the remaining experiments, we have studied the fitness evolution for different queries of our test set. The three functions converge to different numerical values that correspond to the same precision value (.68). We can observe that the square-root cosine function if the first one to converge. A similar behaviour is observed in other queries. Accordingly, the square-root cosine has been the fitness function used in the remaining experiments. We show an example of the correlation between fitness and relevance in Figure 2.

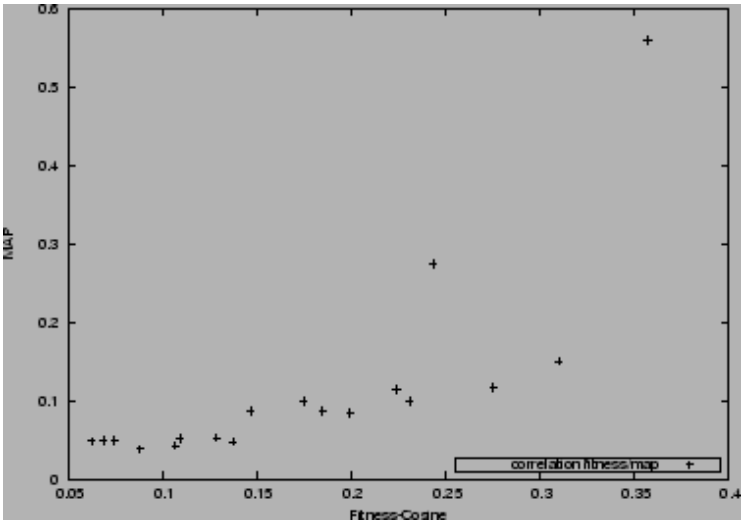

Figure 2: Relationship among the fitness functions considered and the corresponding precision for the *one_query*.

6. TUNING THE GA PARAMETERS

The next step taken has consisted in tuning the parameters of the GA. Figure 3 shows the fitness evolution using different crossover (a) and mutation (b) rates the best query. Results show that we can reach a quickly convergence with values of the crossover rate around 25%. Mutation rates values around 1% are enough to produce a quick convergence.

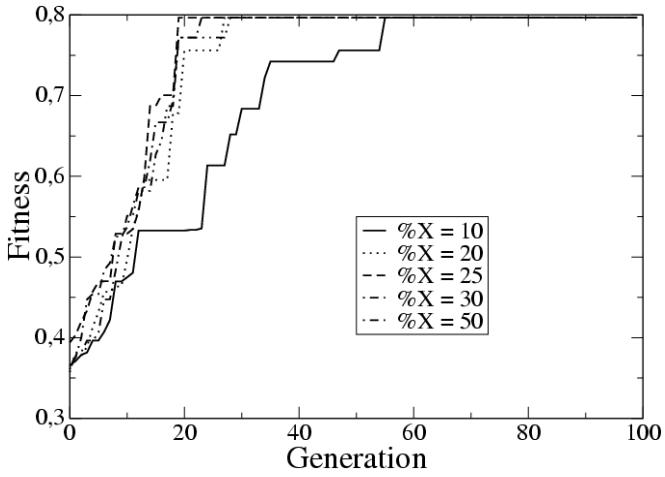

Figure 3: Studying the best crossover (a) and mutation (b) rates for the best_query.

Figure 4 show the fitness evolution for the best (a) and the worst (b) queries, with different population sizes. The plots indicate that small population sizes, such as one of 100 individuals, are enough to reach convergence very quickly.

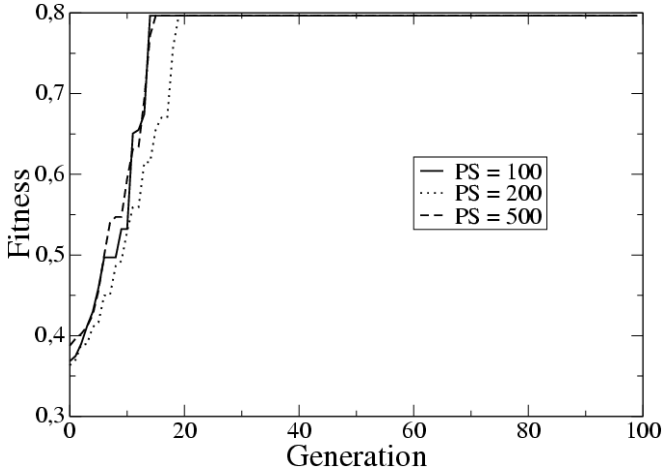

Figure 4: Studying the population size for the best_query (a) and the worst one (b).

7. OVERALL PERFORMANCE

Table 2 presents precision results obtained for the whole set of 40 test queries that we have considered. We have compared our system performance with the results obtained with the original user query (*Baseline*) and with the results obtained using the Porter stemming (*Porter Stemming*). Each individual datum has been computed as the average over 5 different GA runs. We can observe that our system is able to obtain an improvement of 15.20% over the baseline, being this improvement larger that the one obtained by other methods, such as Porter stemming. Furthermore, our system also achieves a great improvement in the system *recall.*
Table 3 shows the recall results. Thus, our system is able to improving the coverage, improving precision at the same time. With the current implementation the mean execution time per query is 1.3 minutes.

	Prec.	Prec5	Prec10	Improvement
Baseline	0.3567	0.4750	0.4150	-
Porter Stem.	0.4072	0.50	0.45	+12.40%
Genetic Stem.	0.4206	0.5150	0.4575	+15.20%

Table 2: Global precision results for the whole set of tested queries. Prec. stands for precision (all documents), Prec5 stands for the precision of the results for the first five documents retrieved, and Prec10, stands for precision for the first ten documents retrieved. Last column is the rate of precision (all documents) improvement.

	Recall	Improvement		
Baseline	0.7035	-		
Porter Stemming	0.7228	+2.67%		
Genetic Stemming	0.7521	+6.46%		

Table 3: Global recall results for the whole set of tested queries.

8. ANALYSIS OF AN EXAMPLE QUERY

Apart from evaluating the numerical performance of our system, we have analysed the queries resulting from the applied expansion process. Let us first consider the query 63 of the collection, the one with best performance (precision increases from .55 to .68, 19.11% of improvement). Table 4 shows the results for this query. The original query *reserva de ballenas* (whale reserve) is a compound term in Spanish. The first observation is the large size of the set of candidate terms, what makes clear the need for a selection. In the final query we can observe that the most representative terms related to the topic, such as *ballena* and *ballenas* (singular and plural Spanish words for whale), and the word *reserva* (reserve) have been added to the query.

User query: *reserva de ballenas*
Set of candidate terms:
reserva, reservaba, reservaban, reservaciones, reservación, reservada, reservadamente, reservadas, reservado, reservados, reservamos, reservan, reservando, reservandose, reservar, reservara, reservaran, reservarlas, reservarle, reservarles, reservarlo, reservarlos, reservarnos, reservaron, reservarse, reservará, reservarán, reservaría, reservarían, reservas, reservase, reserve, reserven, reservista, reservistas, reservo, reservoir, reservándole, reservándolos, reservándose, reservó, ballenas, ballena, ballén, balléna
GA final clausad query: *(ballenas ballena) reserva*

Table 4: Retrieval results of query number 63 . English translation given in the text

Another query is *energía renovable* (Renewable Energy), for which the final query is *(energías energía) (renovables renovable)*. It achieves an improvement of 39.21%.

In other cases, such as for the query *Verduras, frutas y cáncer* (vegetables, fruits and cancer), the final query and the original query are the same. In this case, the whole point of the query is the relationship between the query terms. Because of this, searching independently for alternative forms of the query terms can only worsen the precision. However, the GA is capable of clearing the expanded query, recovering the original query and thus maintaining the system performance.

9. DISCUSSION & RELATED WORK

In this paper we have shown how an evolutionary algorithm can help to reformulate a user query to improve the results of the corresponding search. Our method does not require any user supervision. Specifically, we have obtained the candidate terms to reformulate the query from a *morphological thesaurus*, with provides, after applying stemming, the different forms (plural and grammatical declinations) that a word can adopt. The evolutionary algorithm is in charge of selecting the appropriate combination of terms for the new query. To do this, the algorithm uses as fitness function a measure of the proximity between the query terms selected in the considered individual and the top ranked documents retrieved with these terms.

We have carried out some experiments to have an idea of the possible improvement that the GA can achieve. In these experiments we have used the precision obtained from the user relevance judgements as fitness function. Results have shown that in this case the GA can reach a very high improvement.

We have investigated different proximity measures as fitness functions without user supervision, such as cosine, square cosine, and square-root cosine. Experiments have shown that the best results are obtained with square-root cosine. However, results obtained with this function do not reach the reference results obtained using the user relevance judgements. This suggests investigating other similarity measures as fitness functions.

We have also studied the GA parameters, and see that small values such as a population size of 100 individuals, a crossover rate of 25% and a mutation rate of 1%, are enough to reach convergence. Measures on the whole test set of queries have revealed a clear improvement of the performance, both over the baseline, and over other stemming expansion methods.

A study of the queries resulting after the reformulation has shown that in many cases the GA is able to add terms which improve the system performance, and in some cases in which the query expansion spoils the results, the GA is able to recover the original query.
For the future, we plan to investigate the use of fitness functions related with query performance prediction, because we think that GA are a good framework to test functions oriented to predict the performance of a query. On the other hand, we plan to apply GA to other query expansion and query reformulation methods.

ACKNOWLEDGEMENT

This research study has been done under supervision of Lourdes Araujo from UNED.and Hugo Zaragoza from Yahoo! Research

REFERENCES.

[1] C. Carpineto, R. de Mori, G. Romano, and B. Bigi. An information-theoretic approach to automatic query expansion. *ACM Trans. Inf. Syst.*, 19(1):1-27, 2001.
[2] Y. Chang, I. Ounis, and M. Kim. Query reformulation using automatically generated query concepts from a document space. *Inf. Process. Manage.*, 42(2):453-468, 2006.
[3] H. Chen, G. Shankaranarayanan, L. She, and A. Iyer. A machine learning approach to inductive query by examples: An experiment using relevance feedback, id3, genetic algorithms, and simulated annealing. *JASIS*, 49(8):693-705, 1998.
[4] O. Cordón, F. de Moya Anegón, and C. Zarco. A new evolutionary algorithm combining simulated annealing and genetic programming for relevance feedback in fuzzy information retrieval systems. *Soft Comput.*, 6(5):308-319, 2002.
[5] O. Cordón, E. Herrera-Viedma, C. López-Pujalte, M. Luque, and C. Zarco. A review on the application of evolutionary computation to information retrieval. *Int. J. Approx. Reasoning*, 34(2-3):241-264, 2003.
[6] O. Cordón, E. Herrera-Viedma, and M. Luque. Improving the learning of boolean queries by means of a multiobjective iqbe evolutionary algorithm. *Inf. Process. Manage.*, 42(3):615-632, 2006.
[7] D. E. Goldberg. *Genetic Algorithms in search, optimization and machine learning*. Addison Wesley, 1989.
[8] J. J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
[9] J.-T. Horng and C.-C. Yeh. Applying genetic algorithms to query optimization in document retrieval. *Inf. Process. Manage.*, 36(5):737-759, 2000.
[10] C. Lopez-Pujalte, V. P. G. Bote, and F. de Moya Anegón. A test of genetic algorithms in relevance feedback. *Inf. Process. Manage.*, 38(6):793-805, 2002.
[11] S. M. and S. M. The use of genetic programming to build boolean queries for text retrieval through relevance feedback. *Journal of Information Science*, 23(6):423-431, 1997.
[12] J. L. F.-V. Martín and M. Shackleton. Investigation of the importance of the genotype-phenotype mapping in information retrieval. *Future Generation Comp. Syst.*, 19(1):55-68, 2003.
[13] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution programs*. Springer-Verlag, 2nd edition edition, 1994.
[14] C. Peters and M. Braschler. European research letter: Cross-language system evaluation: The clef campaigns. *JASIST*, 52(12):1067-1072, 2001.
[15] F. E. Petry, B. P. Buckles, T. Sadasivan, and D. H. Kraft. The use of genetic programming to build queries for information retrieval. In *International Conference on Evolutionary Computation*, pages 468-473, 1994.
[16] B. Pôssas, N. Ziviani, J. Wagner Meira, and B. Ribeiro-Neto. Set-based vector model: An efficient approach for correlation-based ranking. *ACM Trans. Inf. Syst.*, 23(4):397-429, 2005.
[17] A. M. Robertson and P. Willet. An upperbound to the performance of ranked-output searching: optimal weighting of query terms using a genetic algorithm. *J. of Documentation*, 52(4):405-420, 1996.
[18] E. Sanchez, H. Miyano, and J. Brachet. Optimization of fuzzy queries with genetic algorithms. application to a data base of patents in biomedical engineering. In *VI IFSA Congress, vol. II*, pages 293-296, 1995.
[19] I. R. Silva, J. N. Souza, and K. S. Santos. Dependence among terms in vector space model. In *IDEAS '04: Proceedings of the International Database Engineering and Applications Symposium (IDEAS'04)*, pages 97-102, Washington, DC, USA, 2004. IEEE Computer Society.
[20] L. Tamine, C. Chrisment, and M. Boughanem. Multiple query evaluation based on an enhanced genetic algorithm. *Inf. Process. Manage.*, 39(2):215-231, 2003.
[21] M. Taylor, H. Zaragoza, N. Craswell, S. Robertson, and C. Burges. Optimisation methods for ranking functions with multiple parameters. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 585-593, New York, NY, USA, 2006. ACM Press.
[22] S. K. M. Wong, W. Ziarko, V. V. Raghavan, and P. C. N. Wong. On modeling of information retrieval concepts in vector space. *ACM Trans. Database Syst.*, 12(2):299-321, 1987.
[23] J.-J. Yang and R. R. Korfhage. Query modification using genetic algorithms in vector space models. *Int. J. Expert Syst.*, 7(2):165-191, 1994.