Training a Classifier for the selection of Good Query Expansion Terms with a Genetic Algorithm

Lourdes Araujo, Joaquín Pérez-Iglesias

Abstract— **Retrieving precise information from large collections of documents or from the web is an important task in our world. The specification of the information needed is done in form of a sequence of terms or query, which is frequently too short or unspecific to allow selecting a set of relevant documents small enough to be inspected by the user. This problem can be alleviated by expanding the query with other terms that make it more specific. The selection of these possible expansion terms is the problem addressed in this work. We have developed a classifier which has been trained for distinguishing good expansion terms. The identification of good terms to train the classifier has been achieved with a genetic algorithm whose fitness function is based on users' relevance judgements on a set of documents. Results show that the training performed by the genetic algorithm is able to improve the quality of the query expansion results.**

I. Introduction

Information search techniques have become critical in our world of global information in which it is impossible to inspect even a tiny part of the documents in most collections, let alone the web. Different systems provide the user more and more sophisticated search techniques which aim at selecting a set of documents in which the user might be interested. Usually, the only source of information to infer this set are a few terms that the user inputs in the system (a *query*). However, user queries are often too short or too ambiguous to allow to reliably determine the information needed.

Search engines use different techniques, taking into account incoming and outcoming links, contents, and other data, for appropriately ranking the list of documents that can be relevant for the query. However, all of them use some kind of index [1] to summarize the document content and quickly identify the documents that contain terms from a particular query. Sometimes user queries are composed of terms that, although related or even synonymous to those of the desired documents are nevertheless different to them, and thus the search system, which is only guided by the index terms, is unable to retrieve those documents.

Because of these limitations of user queries, some researchers are looking for ways of expanding the queries with other terms that make them more focused in the real user interest. Depending on the source of the candidate expansion terms, we can distinguish between different approaches. In the relevance feedback approach, the original query is modified according to the retrieved documents which are judged relevant by the user. The information extracted from the relevant documents can be used either to adjust the weights of the terms in the original query, or to add new terms to the query. Genetic algorithms, which have been applied to different information retrieval problems [2], have been used for both, adjusting term weights [3], [4], [5], [6], [7] and selecting new query terms [8].

Another approach which has the advantage of not requiring user's supervision is known as pseudo-relevance feedback (PRF) [9]. PRF assumes that the top k retrieved documents are relevant and uses them as relevance judgements. However, if this assumption turns out to be false, the query expanded with terms coming from these documents would spoil the retrieval performance. Thus, the selection of good expansion terms, if there are any, becomes critical to the process. A GA has also been applied to this problem [10], but considering only a limited source of expansion terms, the morphological variants which share the same stem with the query terms. This GA uses as fitness function a measure of similarity between the documents and the query given by the cosine of the angle between the vectors representing the document and the query. However, when this technique is applied to general terms as those extracted in the PRF process, the expansion of the query is unable to improve the results.

If we have user relevance judgements available, we can design a different GA. A relevance judgement indicates if a specific document from a collection is relevant for a given query. The set of relevance judgements for a query are made manually by a group of human advisors. By letting the GA use a kind of "perfect" fitness function given by the user relevance judgements we have observed that the GA is able to find a set of expansion terms that largely improve the performance of the original query. This suggests that the problem is not the candidate expansion terms provided by the PRF techniques, but the use of a fitness function which does not reflect the user judgements with enough accuracy. However, we do not have the user relevance judgements for any possible query, and thus we can not rely on this "perfect" GA in general. But we can use the results provided by this GA to train a machine learning system able to select terms for other queries for which we do not have user relevance judgements.

This has been the strategy adopted in this work. The idea is to train a classifier to distinguish good expansion terms from others using a number of features related to the terms and the documents retrieved with the original query, as well as their relationships. Training the classifier requires

Lourdes Araujo and Joaquín Pérez-Iglesias are with Dpto. Lenguajes y Sistemas Informáticos. UNED, Madrid, Spain, email (lurdes|joaquin.perez)@lsi.uned.es.

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

an appropriate set of training data composed of both, "good" and "bad" terms, and characterized by a number of features that can be extracted from any query and set of documents. Classifiers have been previously applied to the extraction of good expansion terms [11], but the distinction of good and bad terms for training the classifier only relies on the results of expanding the query with each candidate term, one by one. If the expanded query improves the relevance of the retrieved documents the term is considered good and bad otherwise. But the process does not consider any relationship among expansion terms. We want to provide a more refined selection of terms to train the classifier, by considering the retrieval improvement achieved by different sets of candidate expansion terms. For the number of candidate terms that we are considering (40 terms), this search could not be performed exhaustively, since it would amount to checking up to $2^{40} \approx 10^{12}$ different combinations. Accordingly, we have resorted to select the best set of expansion terms with the above mentioned "perfect" GA. These refined training data allow us to improve the quality of the classifier into good and bad expansion terms. After the training phase, the classifier can be applied to select expansion terms for other queries for which we do not have user relevance judgements.

The rest of the paper is organized as follows: section II introduces the basic methodologies in unsupervised query expansion; section III describes our strategy for training the classifier; section IV presents the evolutionary algorithm; section V is devoted to the classifier and the features used to train it; section VI presents and discusses experimental results, and section VII draws the main conclusions of this work.

II. Pseudo Relevance feedback

Pseudo relevance feedback (PRF) or blind feedback is a type of relevance feedback where the user is not involved, being the selection of candidate terms for expanding the original query automatically carried out. In this case the candidate terms are extracted from the top k documents which are retrieved as the answer to the original query. The system assumes that the top k documents retrieved are relevant, and thus a subset of terms that describe the content of these documents could increase the informative capability of the query. The new query will contain the original terms plus some of the selected expansion terms.

The process of PRF can be divided into two main stages

- the selection of a subset of terms related to the original query, and
- the assignment of a weight to each of the terms extracted in the previous stage. The weight assigned to each term defines the "importance" of this term within the query.

Assigning a weight to each expansion term constraints the effect of the term over the ranking list of documents obtained. The inclusion of some terms in the original query can deviate the original meaning expressed by the user instead of just refining the original query. Re-weighting the new terms avoids this effect by giving a quantitative value of "importance" to them. The weights of the new terms is usually significantly lower than the ones of the original query terms.

Several methods have been proposed to select the candidate terms and assigning a weight to them. One of the most successful methods is derived from the *Kullback-Leibler Divergence (KLD)*, described in detail in the next subsection. This method has been applied for several years in information retrieval competitions such as TREC[12]. The KLD method is used in this work for the extraction of candidate query terms and also as an alternative approach to the selection of expansion terms. In this approach the expansion terms are the ones with higher KLD values, without the refinement phase provided by the classifier.

Query Expansion Based on Kullback-Leibler Divergence

KLD [13] aims at extracting from the top k retrieved documents the most discriminant terms. A term is considered highly discriminating if it appears frequently on the top k documents and, at the same time, its frequency in the collection is not significant. KLD measures the divergence between two probability distributions: the probability of a term within the top k ranked documents and within the whole collection C. It can be obtained as:

$$\mathrm{KLD}_{p_F,p_C}(t) = p_F(t) \log \left(\frac{p_F(t)}{p_C(t)} \right) \tag{1}$$

where $p_F(t)$ is the probability that term t appears within the top k ranked documents, i.e. its frequency within the documents F retrieved with the original query (feedback documents). $p_C(t)$, is the probability or frequency of the same term t within the whole collection.

Applying the previous equation we will be able to rank all terms from the documents retrieved using the original query, henceforth referred to as feedback documents. After ranking terms using KLD, only those terms with divergence above a given threshold are selected.

Now a further selection process is required to select the most appropriate subset of candidate terms to expand the query. In our case this is done by using a classifier which is trained with the GA described below.

III. Our Approach to train a expansion term classifier

This section presents our proposal for PRF, a novel approach based on a combination of evolutionary algorithms and machine learning techniques. The main purpose is to train a term classifier, able to separate terms that are useful for expansion from those that introduce noise in the original query.

In order to train the classifier, a subset of terms annotated as "good" or "bad" for the expansion are needed. The idea is to use a genetic algorithm to select from each query those terms that maximize the average precision (AP) obtained per query. The AP, which is formally described in subsection IV-A, is a standard quality measure in information retrieval field for the quality of a retrieval system.

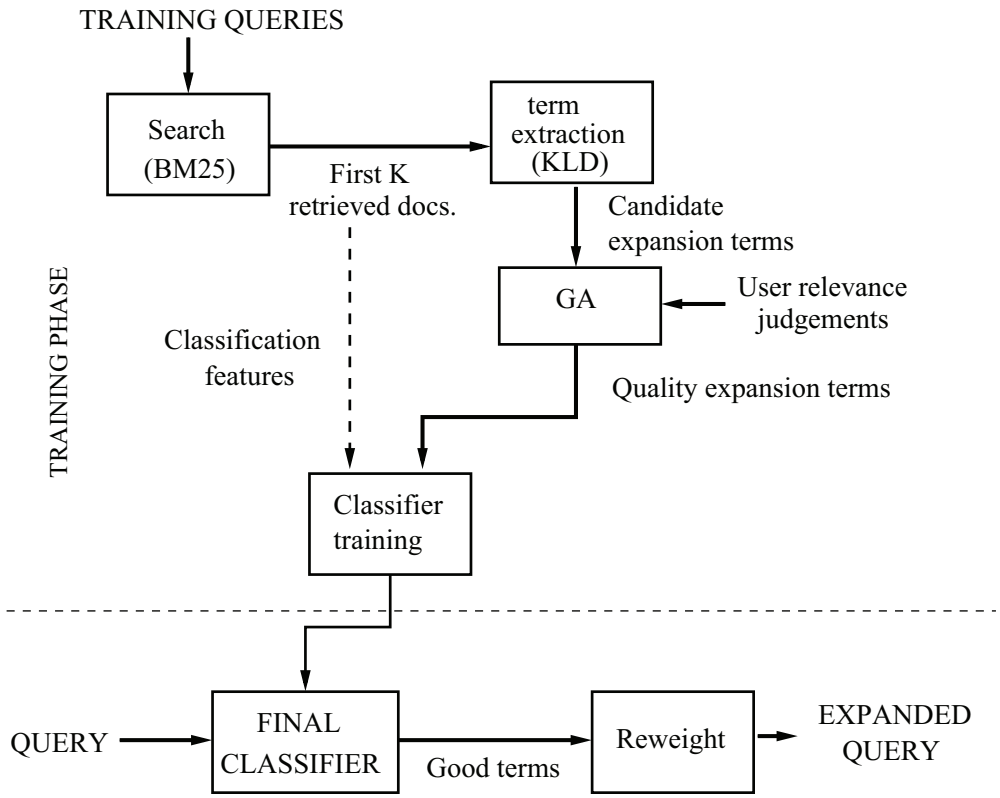

Fig. 1. Scheme of the classifier training process.

Figure 1 shows a scheme of the process applied to obtain a trained classifier which selects good expansion terms. The first steps aim at selecting the set of candidate expansion terms. This is done by submitting the query to a search engine, which implements the retrieval function BM25[14] for recovering relevant documents. This ranking function is considered as state-of-art in information retrieval. The most common variant of BM25 uses two parameters: b, and k_1. These parameters allow modelling the effect of the document length and the term frequency, respectively, in relation to the relevance of a document. The BM25 formula is:

$$R(q,d) = \sum_{t \in q} \frac{\text{occurs}_t^d}{k_1 \left[(1-b) + b \frac{l_d}{\text{avl}_d} \right] + \text{occurs}_t^d} \text{idf}(t)$$

where $occurs_t^d$ is the number of occurrences of term t in the document d; l_d is the length of the document d; avl_d is the average document length within the collection; k_1 is a free parameter, b is another free parameter which should be fixed between 0 and 1, and $idf(t)$ is obtained as:

$$idf(t) = \log \left(\frac{N - df(t) + 0.5}{df(t) + 0.5} \right)$$

where N is the number of documents in the collection and df is the number of documents in which the term t appears.

Once we have a set of relevant documents for the original query, the next step is extracting terms from the k first documents, applying the above described KLD method. Term extraction methodology includes stemming and stopword[1] removing. The obtained set of candidate terms are given to the GA, which using the user relevance judgements as fitness function, looks for the best combination of expansion terms. These combinations, along with a number of features related to them, are used to train the classifier. This way we end up with a classifier that does not require user relevance judgement and can thus provide good expansion terms for new queries without user supervision. When composing the expanded query, each term selected for expansion is weighted by a value which represents its importance for the query, decreasing its contribution with respect to the original query terms.

Let us consider a running example to clarify the process. We take one of the queries from the TREC collection used in the experiments (some examples of queries from this collection appear in Figure 2 in section VI):

Alzheimer's Drug Treatment

The query is submitted to a search system based on the BM25 approach described above. A list of documents ranked by relevance is obtained. The next step is to extract the terms from the retrieved documents and ranking the terms by the divergence value provided by KLD. Table I shows the best

[1]Words too frequent to be discriminating, such as determiners.

TABLE I

LIST OF CANDIDATE TERMS EXTRACTED FROM THE 10 FIRST DOCUMENTS RETRIEVED WITH THE ORIGINAL QUERY. THE FIRST COLUMN CORRESPONDS TO THE STEM OF THE TERMS AND THE SECOND ONE TO THE KLD VALUE. THE MORE SIGNIFICATIVE TERMS, WITH HIGHER VALUE OF KLD, APPEAR AT THE END OF THE LIST.

victim	0.012740664240960575
pharmaceut	0.013260748581091232
placebo	0.013757979004208828
progress	0.013903531163630214
ssris	0.01401376123587397
lambert	0.014168910386894128
depress	0.014309260148836383
known	0.014320527769786599
clinic	0.014978064634819124
tacrin	0.01566397727276215
beta	0.01609770439935924
plaqu	0.016538279020580835
acetylcholin	0.017568833184277568
ondansetron	0.018543504292871212
app	0.018776575588463868
infarct	0.01906392617792763
aami	0.019723540084793323
age	0.01989447962045959
hydergin	0.02093388439896785
receptor	0.02135932028956392
cell	0.021525551305622708
cognex	0.021871270539525027
sumatriptan	0.022545580042821284
neurotransmitt	0.02268004016699352
caus	0.022840430358082496
parkinson	0.02557228570019061
research	0.028619411639784716
disord	0.02924727295203292
degen	0.030845864308654904
protein	0.03180334548985989
memori	0.03263777729752926
symptom	0.03493321776394026
treatment	0.03811911814761063
patient	0.06404161093658585
amyloid	0.0749817928760831
drug	0.08034370024655754
brain	0.10152882234386738
diseas	0.1408059687558589
dementia	0.1567107039483545
alzheim	0.31536285305749

40 candidate expansion terms according to KLD extracted from the 10 first retrieved documents. In fact, the table shows the term stems, since the term extraction process includes a stemming step to increase the term frequencies.

We can observe that the list includes very general terms (their stems), such as *victim*, *known*, *cell*, etc. If we expand the original query with one or more of these general terms, the documents retrieved become less relevant for the query, thus worsening the retrieval performance.

Therefore, we use a genetic algorithm to select the subset of terms from the candidate list that achieved better performance when expanding the query. In our example the terms selected by the GA are the following:

treatment clinic ondansetron cognex patient drug diseas alzheim

This subset of terms are closely related to the query, being *ondansetron* and *cognex* specific drugs for the Alzheimer disease.

Now these terms are labeled as "good" terms in the classifier training. Other terms not selected by the GA are labeled are "bad". A number of features, described in section V, are extracted from the query and the feedback documents, and then used to train the classifier. Once the classifier has been trained with enough queries, it can be applied to new queries, for which we do not have relevance judgements.

IV. EVOLUTIONARY ALGORITHM

For each query the generational GA starts with a set of candidate expansion terms. These terms are extracted from the top ten documents retrieved using the original query and BM25 as the retrieval model, the ranking function described in section III. The set of candidate terms is composed of the 40 terms with a higher divergence value according to the KLD extraction method. No weights will be applied at this point, i.e. all terms (original and expanded) will have the same weight and therefore will have an equivalent importance. Accordingly, a binary representation for the problem is the most natural one, assigning each gene to a candidate query term, where 1 means that the term is included in the expanded query, and 0 that is not included.

TABLE II

EVOLUTIONARY ALGORITHM PARAMETERS.

Parameter	Value
Crossover length	40
Population	200
Crossover rate	0.5
Mutation rate	0.05
Elite size	2
Max number of evaluations	200

Table II shows the GA parameters used in our experiments. The initial population is created at random, with a size of 200 individuals, being two of them considered as elite. The GA runs for 200 generations, using one point crossover and one bit mutation, with a crossover rate of 0.5 and a mutation rate of 0.05.

Next section describes the fitness function used by the GA.

A. Fitness Function

The Average Precision (AP) measure is a standard measure of the quality of a search system in information retrieval, and it is used as the GA fitness function. The computation of AP for a query requires a set of relevance judgements.

Precision for a given document d is defined as the fraction of relevant documents within the set of documents retrieved with a higher rank than document d (including d). The Average Precision for a set of relevant documents $Rel = [d_1, d_2, \ldots, d_n]$ is obtained as the mean precision of all these documents, i.e.

$$AP = \frac{1}{n} \sum_{j=1}^{n} Precision(R_j)$$

Notice that if the documents of *Rel* appear at positions $j_1 < j_2 < \cdots < j_n$ in the retrieved list, then AP can also be obtained as

$$AP = \frac{1}{n}\left(\frac{1}{j_1} + \frac{2}{j_2} + \cdots + \frac{n}{j_n}\right).$$

At the end of the evolution (either when the maximum number of generations is reached or when some individual with fitness value 1 appears) the GA provides a subset of expansion terms which increases the number of relevant documents retrieved.

The training corpus of terms for the classifier is composed of an equal number of "good" and "bad" terms, so as to build a balanced corpus. The terms selected by the GA for each query are the "good" examples of the corpus. The same number of "bad" terms are selected among those that have not been selected by the GA. Specifically, we selected as "bad" terms those that have been present less often in the best individual of the GA along the generations.

V. Term Classifier

The next phase is the term classifier training. In the training phase each term is represented with the feature vector and with the class label (Good,Bad) provided by the genetic algorithm, indicating whether the term is or is not suitable for expansion. The best results have been obtained by applying the SVM (Support Vector Machine) implementation from Weka[15]. The default training SVM implementation within Weka is the well-known SMO[16], using a Radial Basis Function Kernel with a default value for γ equals to 0.1.

Features for Term Classification

This section describes the features applied to represent a term. They mainly refer to the relative frequency of the candidate term in different sets of documents, values of coocurrence between query terms and the candidate term and the classical Inverse Document Frequency.

- *Feature 1*
  Probability of candidate term t within the top feedback documents. It is a measure of the ratio of the candidate term frequency (tf) within the top feedback documents to the total number of terms in the top feedback documents.
  $$f(t) = \log_2\left(1 + \frac{\sum_{d\in\text{top}} \text{tf}(t)}{\sum_{d\in\text{top}} N}\right)$$
- *Feature 2*
  Probability of candidate term t within the collection. It is a measure of the ratio of the candidate term frequency within the collection to the total number of terms in the collection.
  $$f(t) = \log_2\left(1 + \frac{\sum_{d} \text{tf}(t)}{\sum_{d} N}\right)$$
- *Feature 3*
  This feature measures the value of coocurrence between the candidate term t and a query term q_i within the feedback documents. This process is repeated for each query term. The accumulated value is divided by the t frequency within the top feedback documents, and normalized by the query length l_q.
  $$f(t) = \log_2\left(1 + \frac{\sum_{d\in\text{top}} \sum_{q_i\in Q} \text{cooc}(q_i, t)}{l_q \sum_{d\in\text{top}} \text{tf}(t)}\right)$$
- *Feature 4*
  This feature measures the value of coocurrence between the candidate term t and a query term q_i within the whole collection. This coocurrence value is computed for each query term. The accumulated value is divided by the term frequency within the collection, and normalized by the query length l_q.
  $$f(t) = \log_2\left(1 + \frac{\sum_{d} \sum_{q_i\in Q} \text{cooc}(q_i, t)}{l_q \sum_{d} \text{tf}(t)}\right)$$
- *Feature 5*
  As in feature 3, the coocurrence between the candidate term t and the query terms is computed within the feedback documents. In this case, it is considered that t and q_i appears together if both are at a distance (in number of terms) lower than or equal to p (p was set to 10).
  $$f(t,p) = \log_2\left(1 + \frac{\sum_{d\in\text{top}} \sum_{q_i\in Q} \text{cooc}(q_i, t, p)}{l_q \sum_{d\in\text{top}} \text{tf}(t)}\right)$$
- *Feature 6*
  Equivalent to the previous feature, but in this case the measure is calculated within the whole collection and not only within the feedback document (again p was set to 10).
  $$f(t,p) = \log_2\left(1 + \frac{\sum_{d} \sum_{q_i\in Q} \text{cooc}(q_i, t, w)}{l_q \sum_{d} tf(t)}\right)$$
- *Feature 7*
  Equivalent to feature 5 but with coocurrence between at least two query terms and a candidate term.
- *Feature 8*
  Equivalent to feature 6 but with coocurrence between at least two query terms and a candidate term.
- *Feature 9*
  Equivalent to feature 5 but with coocurrence between all query terms and a candidate term.
- *Feature 10*
  Equivalent to feature 6 but with coocurrence between all query terms and a candidate term.
- *Feature 11*
  The Inverse Document Frequency of the term t.
  $$f(t) = \log\left(\frac{N}{\text{docFreq}_t}\right)$$
  where N is the number of documents in the collection, and $docFreq_t$ is the number of documents in the collections where t appears.

VI. Experimental results

This section introduces a experimental design to prove the viability of the proposed approach. The obtained results are described in detail below.

A. Experimental Setup

<num> Number: 330
<title> Iran-Iraq Cooperation
<desc> Description:
This query is looking for examples of cooperation or friendly ties between Iran and Iraq, or ways in which the two countries could be considered allies.
<narrative> ...

<num> Number: 339
<title> Alzheimer's Drug Treatment
<desc> Description:
What drugs are being used in the treatment of Alzheimer's Disease and how successful are they?
<narrative> ...

<num> Number: 372
<title> Native American casino
<desc> Description:
Identify documents that discuss the growth of Native American casino gambling.
<narrative> ...

Fig. 2. Some queries from the TREC collection for information retrieval evaluation.

We have used a standard benchmark collection of documents in information retrieval to test our approach. Evaluation has been carried out with the set of documents from TREC Disk4 & 5, minus Congressional Record. These data contain around 528,000 documents with a total size of almost 2 GB [17]. The collection is composed of the full text of various newspaper and newswire articles plus government proceedings. A test collection consists of three parts: a set of documents, a set of queries that can be answered by some of the documents in the collection, and the relevance judgement which are the set of relevant documents for the query according to an advisor. Figure 2 shows examples of queries for which the TREC collection provides relevance judgements. Each query is assigned an identifier (*num*), a *title*, and a description (*desc*). Only the field *title* from each query was employed in our experiments. The set of queries are the first 150 available from the same track, i.e. queries 301-450, with their relevant judgements. The relevance judgements are just a list per query with the document identifiers and a 1 if it is relevant or a 0 otherwise.

The parameters for the retrieval function BM25 have been fixed to their standard values $k_1 = 1.2$ and $b = 0.75$.

For the two compared methods, KLD and GA, the candidate terms for expansion have been extracted from the top ten documents ranking, and a total of 40 terms per query have been selected. We have applied to both methods a term reweighting scheme to decrease the importance of the expansion terms with respect to the original query terms. For the classical KLD expansion method, the weights computed with KLD have been used to constraint the importance of expanded terms. For the terms selected by the classifier the next equation with a free parameter β is applied to reweight the expanded terms, before executing the expanded query:

$$w(t,q) = \beta \cdot \frac{w(t)}{w_{max}(t)}$$

where $w(t)$ is the weight of term t and $w_{max}(t)$ is the maximum term weight found in the expanded query. This equation is a simplification based on the Rocchio[18] reweighting algorithm. In this case β has been set to 0.7, after an exhaustive exploration within the range $(0, 1)$.

The candidate terms selected by the GA are all weighted with a value of 1, thus $w(t)$ and $w_{max}(t)$ are equal to 1. Therefore all expanded terms will have a weight equal to 0.7, and the original ones to 1.

B. Classification Results

The SVM classifier has been trained with a 10-fold cross validation, using 1466 terms labeled as "Good", and the same number of "Bad" terms. The percentage of terms classified correctly is 61%. Table III shows the results obtained after the evaluation of the classifier.

Results in table III indicate the difficulty of the task, since the overall performance of the system is not very impressive. Concerning "Good" terms, less than half of the terms are labeled correctly, although with a low rate of false positives (0.193). This fact suggests that although a great number of real "Good" terms are classified incorrectly, those labeled as "Good" are really good terms for expansion in most cases. As for "Bad" terms, the classifier achieves a better performance (0.807), but with a large number of false positives (0.584), that is, the classifier is biased toward classify terms as "Bad". As a consequence of this, the classifier will label more terms as "Bad" than as "Good", but "Good" terms will be labeled with high precision.

It should be taken into account that in the case of query expansion the best results are obtained adopting conservative approaches according to which the query is not expanded if there are chances to worsen the results. Thus, the results of the classifier are better than the figures can suggest at a first glance since the rate of false positive for good expansion terms is very low.

TABLE III
Classifier results after the 10-fold cross validation. TP stands for the true positives ratio and FP for the false positives ratio. Precision, Recall and the F-Measure results are per class.

Class	TP Rate	FP Rate	Precision	Recall	F-Measure
Good	0.416	0.193	0.687	0.416	0.518
Bad	0.807	0.584	0.576	0.807	0.673

A second evaluation in terms of document precision (AP) obtained by expanding the query with the terms suggested by the classifier, should prove the quality of the terms classification.

C. MAP Results

After computing the AP value for each query, the average of these values for the set of m queries is known as Mean Average Precision (MAP):

$$MAP = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{n}\sum_{j=1}^{n}Precision(R_j)$$

Table IV shows the performance of our approach in terms of the MAP measure, i.e. the change in the relevance of the retrieved documents after expanding the query with terms obtained with different approaches. The two first lines correspond to two baselines: the ranking obtained after the execution of the original query using the retrieval model BM25 and the genetic algorithm output (GA-Oracle which uses the user relevance judgement). The next line corresponds to the classical expansion method based on KLD, and finally the last line shows the results of the classifier trained with the data obtained from the genetic algorithm.

It can be observed that the use of the expanded query instead of the original query improves the overall quality of the search system in terms of MAP, as it was expected. The classical KLD expansion method improves the quality obtained with the execution of the original query in more than a 10%. Next, the automatic expansion based on the trained classifier outperforms clearly the previous approach in more than 8%, and almost 19% over the original query. This improvement proves that although the classifier does not obtain strong results, it was enough to improve a state-of-the-art PRF technique. Thus the machine learning approach to carry out PRF is able to achieve better results in comparison with classical approaches, whenever it is trained with data of enough quality. Moreover it is expected that improving the performance of the classifier, in at least two ways *a*) adding more features; and *b*) trying with different classification algorithms, the quality of the ranking list retrieved should be improved.

It should be highlighted the room for improvement suggested by the GA-Oracle, with a 112% increase in terms of MAP. It should be taken into account though that this improvement is a consequence of an overfitted model, that is, the use of the same function for evaluation as for the fitness of the GA. The obtained results with the use of the GA-Oracle can be interpreted as an upper-bound to query expansion methods.

Table V shows a comparison between the different methods in relation with the number of queries improved or worsened by each method. As expected the oracle improves the AP measure for all the queries related to the other methods, except for two queries which the classifier is able to improve, but not the GA-Oracle. Comparing the KLD approach against the trained classifier, it is interesting to observe how the proposed method obtains better results than KLD for 92 queries (61%). On the other hand KLD obtains better results than the classifier just for 58 queries (38%). With the use of the classifier the performance for 48 queries (32%) decreases. The same behavior is observed with the KLD approach, decreasing the performance of 60 (40 %) queries. Finally the classifier and KLD improve 100 (66%) and 90 (60%) respectively.

VII. Conclusions

In this paper a pseudo relevance feedback method has been proposed. This method uses a genetic algorithm to generate a set of terms that are appropriate for expansion. The genetic approach has been proven as a strong method to select those terms that maximize the quality of the retrieved documents in terms of MAP. The use of the GA has shown that there is a large room for improvement, which is not currently covered by standard methods of term selection and term weighting. Current methods add the whole set of candidate terms to the expanded query, whereas the GA has shown that more than half of them have not effect at all or even worse, dramatically decrease the quality. Assigning weights to terms in order to constrain the effect of these ones over the final query avoids a performance degradation.

The high quality data provided by the GA are then used to train a classifier in order to automate the expansion process. The classifier evaluation showed that the set of features used to represent a term were not sufficiently descriptive with a 61% of terms correctly labeled. These results suggest the introduction of new features. Some terms can induce a gain in AP in combination with other specific terms, but not by their own. The set of features could be extended with the introduction of statistics not based in single terms but in different term combinations, with more complexity than just coocurrence.

Although the results obtained indicate that there is still a large room for improvement, it has been shown empirically that a simple classifier can improve the quality of the retrieved documents. This leads us to conclude that the information extracted from the GA was good enough to improve the overall quality. It should be noted that the GA was very simple and does not includes any re-weighting for terms, i.e. only a Boolean representation was applied in order to model queries and terms. A more sophisticated approach

TABLE IV

Improvement of the relevance of the retrieved documents (in terms of MAP) expanding the query with terms obtained with different methods: the BM25 retrieval function (baseline), the GA with the user judgements (GA-Oracle), the KLD method and the proposed classifier.

	MAP	**Δ MAP**
BM25	0.2130	
GA-Oracle	0.4518	(+112.1%)
KLD	0.2352	(+10.4 %)
Classifier	0.2534	(+18.9 %)

TABLE V

DIFFERENCE IN THE NUMBER OF QUERIES IMPROVED (OR WORSENED) BY ONE RESPECT ANOTHER ONE. READING FROM LEFT TO RIGHT SHOWS THE WORSENED QUERIES AND FROM TOP TO BOTTOM THE IMPROVED QUERIES.

	BM25	**GA-Oracle**	**KLD**	**Classifier**
BM25	-	150	90	100
GA-Oracle	0	-	0	2
KLD	60	150	-	92
Classifier	48	148	58	-

able to reweight terms could improve the results obtained in this work.

ACKNOWLEDGEMENTS

This paper has been funded in part by the Spanish MICINN projects NoHNES (Spanish Ministerio de Educación y Ciencia - TIN2007-68083) and QEAVis-Catiex (TIN2007-67581-C02-01), as well as by the Regional Government of Madrid under the Research Network MA2VICMR (S2009/TIC-1542).

REFERENCES

[1] R. A. Baeza-Yates and B. A. Ribeiro-Neto, *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999. [Online]. Available: sunsite.dcc.uchile.cl/irbook/

[2] 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*, vol. 34, no. 2-3, pp. 241–264, 2003.

[3] 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*, vol. 52, no. 4, pp. 405–420, 1996.

[4] J.-J. Yang and R. R. Korfhage, "Query modification using genetic algorithms in vector space models," *Int. J. Expert Syst.*, vol. 7, no. 2, pp. 165–191, 1994.

[5] 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*, 1995, pp. 293–296.

[6] C. Lopez-Pujalte, V. P. G. Bote, and F. de Moya Anegón, "A test of genetic algorithms in relevance feedback," *Inf. Process. Manage.*, vol. 38, no. 6, pp. 793–805, 2002.

[7] J.-T. Horng and C.-C. Yeh, "Applying genetic algorithms to query optimization in document retrieval," *Inf. Process. Manage.*, vol. 36, no. 5, pp. 737–759, 2000.

[8] 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.*, vol. 6, no. 5, pp. 308–319, 2002.

[9] C. Buckley, G. Salton, and J. Allan, "Automatic retrieval with locality information using smart." in *TREC*, 1992, pp. 59–72.

[10] L. Araujo and J. R. Pérez-Agüera, "Improving query expansion with stemming terms: A new genetic algorithm approach," in *EvoCOP*, 2008, pp. 182–193.

[11] G. Cao, J.-Y. Nie, J. Gao, and S. Robertson, "Selecting good expansion terms for pseudo-relevance feedback," in *SIGIR*, 2008, pp. 243–250.

[12] C. Lioma, C. Macdonald, V. Plachouras, J. Peng, B. He, and I. Ounis, "University of glasgow at trec 2006: Experiments in terabyte and enterprise tracks with terrier." in *TREC*, E. M. Voorhees and L. P. Buckland, Eds., vol. Special Publication 500-272. National Institute of Standards and Technology (NIST), 2006. [Online]. Available: http://dblp.uni-trier.de/db/conf/trec/trec2006.html#LiomaMPPHO06

[13] C. Carpineto, R. de Mori, G. Romano, and B. Bigi, "An information-theoretic approach to automatic query expansion," *ACM Trans. Inf. Syst.*, vol. 19, no. 1, pp. 1–27, 2001.

[14] S. E. Robertson and S. Walker, "Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval." in *SIGIR*, W. B. Croft and C. J. van Rijsbergen, Eds. ACM/Springer, 1994, pp. 232–241. [Online]. Available: http://dblp.uni-trier.de/db/conf/sigir/sigir94.html#RobertsonW94

[15] E. Frank, M. A. Hall, G. Holmes, R. Kirkby, B. Pfahringer, and I. H. Witten, *Weka: A machine learning workbench for data mining.* Berlin: Springer, 2005, pp. 1305–1314. [Online]. Available: http://researchcommons.waikato.ac.nz/handle/10289/1497

[16] J. C. Platt, *Fast training of support vector machines using sequential minimal optimization*. Cambridge, MA, USA: MIT Press, 1999, pp. 185–208.

[17] E. M. Voorhees, "Overview of the trec 2004 robust retrieval track," in *In Proceedings of the Thirteenth Text REtrieval Conference (TREC)*, 2004.

[18] J. Rocchio, *Relevance Feedback in Information Retrieval*. Prentice-Hall Inc., 1971, ch. 14, pp. 313–323.