Often the same transformation functions and global term weights are used for queries and documents, but different functions may be used. Other definitions of local and global term weights have been proposed. However, the classical VSM is characterized by the fact that only the terms contained in a document or query are represented in its vector (e.g., have a non-zero value in the vector representation).

2.2. *The context vector model*

In this section we briefly introduce the context vector model for IR.[4,5] The CVM is based on the vector space model, and can be considered as a variation of the VSM. The difference is that it incorporates term dependencies in the document and query representations. In this sense, each term in a document indicates the existence of a set of different concepts, where the concepts themselves are represented by terms. As a result, a term that does not actually occur in a document may have a relatively high value in its vector representation, just because it is strongly related to many of the terms contained in that document.

Context vector indexing consists of three steps: i) compute the classical tf-based document vectors, ii) generate a term correlation matrix that reflects the term relationships, and iii) transform the tf-based document vectors into context vectors. In the first step, the term frequencies of all terms are obtained for all documents. Then, a $n \times n$ term correlation matrix C is calculated, which defines the relationships among terms. Each element c_{kj} of this matrix can be interpreted in two ways: i) as the degree with which the term t_k indicates the existence of the term t_j, or ii) as the importance of t_j in the conceptual description of the term t_k. The kth column vector of C, denoted by $\vec{c}_k$, is called the term context vector for the term t_k. If C is the identity matrix (e.g., the relationships among different terms are zero), then the context vectors are pair-wise orthogonal and the resulting model simulates the classical VSM. We used four different definitions for C, denoted *prob*, *intu*, *prob0diag* and *intu0diag*. All of them are based on the concept of co-occurrence frequency, e.g., the number of times two different terms occur in the same documents. Their definitions are as follows:

$$prob\text{:}\ c_{kj} = \begin{cases} 1 & if\ k = j \\ \dfrac{\sum_{i=1}^{m} tf_{ik}\ tf_{ij}}{\sum_{i=1}^{m} \left(tf_{ik} \sum_{r=1;r=k}^{n} tf_{ir} \right)} & if\ k \neq j \end{cases} \tag{3}$$

and

$$intu\text{:}\ c_{kj} = \begin{cases} 1 & if\ k = j \\ \dfrac{\sum_{i=1;tf_{ij}=0}^{m} tf_{ik}}{\sum_{i=1}^{m} tf_{ik}} & if\ k \neq j \end{cases} . \tag{4}$$

prob0diag and *intu0diag* are defined in the same way, but all elements c_{kj} with $k = j$ are set to zero.

Document context vectors $\vec{d_i} = \langle w_{i1}, w_{i2}, \ldots, w_{in} \rangle$ are obtained by calculating the centroid of the term context vectors for all terms belonging to a document:

$$w_{ij} = \sum_{k=1}^{n} tf_{ik} \frac{c_{kj}}{|\vec{c}_k|} \Bigg/ \sum_{k=1}^{n} tf_{ik} \ , \tag{5}$$

where $|\vec{c}_k|$ is the Euclidean vector length of the term context vector $\vec{c}_k$.

During the retrieval process a query is transformed into a query context vector $\vec{q} = \langle w_{q1}, w_{q2}, \ldots, w_{qn} \rangle$ using the same indexing method as applied to documents. Then, the similarity of this vector to each document context vector is calculated using some similarity measure and some global term weights. Finally, the documents are presented as a list, ranked by their similarity to the query.

2.3. *The set of expert candidates*

The set of retrieval experts we consider consists of different VSM and CVM variations and of variants that combine both models. With respect to the latter, it should be noted that both models use the same n-dimensional space for representing documents and queries - the space spanned over all index terms. This allows us to define retrieval methods that combine the CVM with the classical VSM (e.g., queries may be indexed to tf-based vectors and documents to context vectors or vice versa).

Besides the expert variants that follow directly from the use of the two models, we define several transformation functions and global term weighting schemes. Furthermore, an expert may use different transformation functions and/or different global term weighting schemes for queries and documents. All experts use the cosine similarity measure (1) to compute the relevance of a document to a query.

With these parameters each particular retrieval expert can be specified by a tuple $\langle dtf, qtf, dw, qw, cvmtype \rangle$ where:

(i) *dtf*: specifies the transformation function used for documents,
(ii) *qtf*: specifies the transformation function used for queries,
(iii) *dw*: specifies the global term weighting scheme applied to document terms,
(iv) *qw*: specifies the global term weighting scheme applied to query terms,
(v) *cvmtype*: specifies the type of the term correlation matrix used in CVM indexing.

The last parameter (*cvmtype*) is only required if a CVM dependent transformation function or global term weighting scheme has been chosen. The next two subsections describe the different alternatives for each of these variables.

2.3.1. *Transformation functions (local term weights)*

Two basic transformation functions have been implemented:

(i) *tf*: w_{ij} is defined by the term frequencies (tf_{ij}),

(ii) *cvm*: w_{ij} is defined by Eq. (5).

With respect to query indexes, it has been argued that it is sometimes better to use binary instead of *tf* indexes.[5] This is due to the fact that highly descriptive terms often occur just once in a query. On the contrary, words that are poor descriptors may occur more often. We take this fact into account and define two additional transformation functions for queries:

(i) *bin*: w_{qj} is one if t_j occurs in q and zero if not,
(ii) *cvmbin*: w_{qj} is obtained by using the binary query vectors as the starting point for the context vector calculation.

The type of term correlation matrix used in the CVM based transformation functions is specified by the parameter *cvmtype*. The possible choices are: *prob*, *intu*, *prob0diag*, and *intu0diag*, as defined above.

2.3.2. *Term weighting schemes (global term weights)*

We use 14 different term weighting schemes:

(i) *no*: no weight is used (e.g., constant weight of one for all terms),
(ii) *idf*: inverse document frequency (Eq. (2)),
(iii) *tfmamd*: modified average mean deviation over tf document vectors,
(iv) *tfmvar*: modified variance over tf document vectors,
(v) *dcvmamd*: modified average mean deviation over document context vectors,
(vi) *dcvmvar*: modified variance over document context vectors,
(vii) *tcvmamd*: modified average mean deviation of term context vectors,
(viii) *tcvmvar*: modified variance of term context vectors,
(ix) *idftfmamd*, *idftfmvar*, *idftcvmamd*, *idftcvmvar*, *idfdcvmamd*, and *idfdcvmvar*: combinations of the previous weights with *idf*.

Weights (iii) to (vi) measure the dispersion of the local weights of a term across the tf document vectors and document context vectors, respectively. The weights (vii) and (viii) measure the dispersion of the values in a term context vector. We define the modified average mean deviation of a sample X of size k by

$$mamd(X) = 1 + \left(\sum_{i=1}^{k} \left| X_i - \overline{X} \right| \right) / (m \cdot \overline{X}) \,, \tag{6}$$

where $\overline{X}$ is the sample mean. The modified variance is given by

$$mvar(X) = 1 + \log_2 \left(1 + S_X^2 / \overline{X}\right) \,, \tag{7}$$

where S_X^2 is the sample variance. However, for *tcvmvar* we used a slightly different definition that performed better than Eq. (7) in the experiments:

$$mvar(X) = 1 + S_X^2 / \overline{X} \,. \tag{8}$$

Before calculating the weights, tf and CVM document vectors are normalized to an Euclidean length of one. The weights calculated over document and term context vectors (weights (v) to (viii) and their combinations with *idf*) depend on the definition of the term correlation matrix. They are calculated for all four different correlation matrix types.

The given definitions for the five parameters of a retrieval expert lead to 6272 different parameter combinations. However, there are 72 combinations that do not use the correlation matrix (neither in the transformation functions nor in the global term weighting schemes). Considering that these combinations are repeated for each of the four possible values of *cvmtype*, the number of different retrieval experts is 6056 ($6272 - 3 * 72$).

3. Related Work

Some of the previous methods center the data fusion approach on combining the similarities obtained with different query representations or on combining different query representations directly.[9,10,7,11,12,13] In all these cases the best combinations perform better than the individual systems. Other researchers investigate combinations of different retrieval models. McCabe, et al. analyze combinations of the vector space model and the probabilistic model.[14] In their results, the fusion approach does not significantly improve the effectiveness over the single systems. They argue that this is due to the high overlap in the result sets of both combined experts.

Vogt, et al. use linear combinations of three experts: two VSM variants (using the *binary* and the *tf-idf* weighting schemes), and one based on LSI.[15] In their approach, the set of parameters for the linear combination - the weights given to the individual experts - are determined automatically through training on the top 100 documents for each query. The learned combination function performs considerably better on the training set than the single experts. However, they find that their technique on learning on the top 100 documents did not generalize well to the entire set of documents. There, the combination only does as well as the second-best expert. A similar work is described by Bartell, et al.[16] In their paper, the authors apply combinations of three different experts to two test collections. They use combinations of a "count expert" counting the number of query terms in the document and two VSM experts that analyze different types of terms (a "term expert" for white-space delimited terms and a "phrase expert" searching for adjacent word pairs). Optimization of the expert weights is done on a set of training queries and the proposed combination is then tested on a set of test queries. In both collections the optimized combination performs better than any of the individual systems.

Dumais proposes combinations of different query representations with latent semantic indexing in information routing.[7,11] Hofmann combines his probabilistic latent semantic indexing model with the classical vector space model with *tf-idf* weights in a linear combination.[8] He also combines multiple PLSI variants (each

with a different number of dimensions) and argues that such combinations have a noise reduction effect.

An interesting combination scheme has been analyzed by Savoy, et al.[17] These authors combine the OKAPI probabilistic model with various vector space schemes. However, they neither combine the retrieval status values nor the rank of each retrieved record. Instead, they use an automatic selection procedure that tries to find the best retrieval expert for each presented query. This approach has the advantage of limiting the search time to one retrieval algorithm because only one retrieval expert is selected for each query. The problem of this approach is that query characteristics have to be found that effectively determine which expert will perform better for a given query. In their experiments no performance gains could be shown over the individual experts.

Initial work on data fusion by Fox and Shaw propose six different formulae for combining similarity values of different retrieval experts.[12] Fox and Shaw basically analyze combination functions based on the minimum, the maximum, and the sum of the individual similarities. The formula that performed best is:

$$CombMNZ = SUM(\text{Individual similarities}) \times \text{Number of non-zero similarities}. \tag{9}$$

More recent studies compute the final similarity scores as a linear combination of the individual similarities.[16,8,14,18,19,15]

The problem of determining when it makes sense to combine different IR systems and which experts should be combined has been addressed in several papers.[20,21,18,19] These articles analyze numerous linear combinations of a variety of retrieval experts and try to find indicators that could determine the potential results of possible expert combinations. Lee analyzes the overlap of relevant and non-relevant documents between two different retrieval systems and concludes that their combination makes sense when the systems return similar sets of relevant documents but different sets of non-relevant documents.[20] Vogt and Cottrell propose several indicators; some are based on measures on the individual experts, such as precision and recall, and some on combination measures such as overlap.[21,18,19] They conclude that the characteristics for effective fusion of two experts are: i) at least one has high performance, ii) both return similar sets of relevant documents, iii) both return dissimilar sets of non-relevant documents, iv) both distribute scores similarly, and v) both do not rank relevant documents in the same fashion.

Several researchers have previously used GA and genetic programming for IR tasks. Much of the research is concerned with improving query representations in a relevance feedback scenario.[22,23,24,25,26,27] There, the goal of the GA is to refine the set of query terms and their associated weights using relevance information given by the user. GA have been used as well for generating or adapting the matching function employed in a retrieval system.[28,29] Matching function optimization has also been studied by Bartell, et al.[30] In the latter, however, instead of genetic algorithms the authors use conjugate gradient - a gradient-based optimization method - to determine the parameter settings. Adapting the matching function can be considered

as an expert selection approach, where the system tries to find the best retrieval method for a given document collection. In this sense, these approaches are similar to ours (e.g., when the number of combined experts is set to one). However, in our settings, all experts use the same matching function (we use the standard cosine function) and the system selects experts by selecting a term weighting scheme, and document and query transformation functions.

4. Combining Retrieval Experts – Retrieval Strategies

We combine retrieval experts by means of a linear combination of their individual similarity scores. Let $\{RE_1, \ldots, RE_k\}$ be a set of retrieval experts and let $s_{RE_j}(d_i, q)$, for $1 \leq j \leq k$, denote the similarity of the document d_i to the query q calculated with the expert RE_j. The overall similarity estimate of the combination of $\{RE_1, \ldots, RE_k\}$ is given by:

$$s(d_i, q) = p_1(\frac{\pi}{2} - \arccos(s_{RE_1}(d_i, q))) + \ldots + p_k(\frac{\pi}{2} - \arccos(s_{RE_k}(d_i, q))), \quad (10)$$

where p_j, for $1 \leq j \leq k$, are weights given to the individual experts. A set of experts with associated weights is called *retrieval strategy*.

We use $\frac{\pi}{2} - \arccos(s_{RE_j}(d_i, q))$ instead of the cosine values because this function grows linearly as the angle between the two vectors decreases. We think that this corresponds better to the underlying assumption of the vector space model, that the angle between the vectors of a document and of a query determines their similarity. In an IR system with just a single expert both measures will lead to exactly the same ranking results. However, as soon as experts are combined this will no longer be the case as it can be seen in the following example. Suppose we have two document vectors $\vec{d_1}$ and $\vec{d_2}$, a query vector $\vec{q}$, and two retrieval experts RE_1 and RE_2. Furthermore, suppose that the angles between $\vec{d_1}$ and $\vec{q}$ in RE_1 and RE_2 are 30° and 20°, respectively, and that the angles between $\vec{d_2}$ and $\vec{q}$ are 10° and 40°. A linear combination of the cosine values of RE_1 and RE_2 with uniform weights will judge d_1 more relevant than d_2 (with an estimate of 1.81 versus 1.75). Assuming that the similarity should be determined by the angles and not by the cosine values, however, a linear combination of RE_1 and RE_2 with both experts weighting one should give the same relevancy to the documents d_1 and d_2. This is insured using Eq. (10), e.g., both documents will get a relevance value of 2.27. We have tried both approaches experimentally and the above equation is generally better. Nonetheless, the differences are not significant.

5. Learning Retrieval Strategies with Genetic Algorithms

Genetic algorithms are inspired by the principles of selection and inheritance of natural evolution.[31] In our framework, the goal of the GA is to find an optimal retrieval strategy (combination of retrieval experts) for a given document collection. Each individual of the population represents a set of k retrieval experts, where each expert is specified by an instantiation of the system parameters $\langle dtf, qtf, dw, qw, cvmtype \rangle$

and the weight given to that expert in the combination function. Experts are encoded as strings of (binary) genes of length 18, $i = \langle i_1, \ldots, i_{18} \rangle$, where the encoding is as follows:

(i) i_1 : document transformation function (*dtf*): 0 - *tf*, 1 - *cvm*;
(ii) $\langle i_2, i_3, i_4, i_5 \rangle$: global term weighting scheme for documents (*dw*): 0000 - *no*, 0001 - *idf*, 0010 - *tfmamd*, 0011 - *tfmvar*, 0100 - *dcvmamd*, 0101 - *dcvmvar*, 0110 - *tcvmamd*, 0111 - *tcvmvar*, 1000 - *idftfmamd*, 1001 - *idftfmvar*, 1010 - *idfdcvmamd*, 1011 - *idfdcvmvar*, 1100 - *idftcvmamd*, 1101 - *idftcvmvar*;
(iii) $\langle i_6, i_7 \rangle$: query transformation function (*qtf*): 00 - *tf*, 01 - *bin*, 10 - *cvm*, 11 - *cvmbin*;
(iv) $\langle i_8, i_9, i_{10}, i_{11} \rangle$: global term weighting scheme for queries (*qw*): 0000 - *no*, 0001 - *idf*, 0010 - *tfmamd*, 0011 - *tfmvar*, 0100 - *dcvmamd*, 0101 - *dcvmvar*, 0110 - *tcvmamd*, 0111 - *tcvmvar*, 1000 - *idftfmamd*, 1001 - *idftfmvar*, 1010 - *idfdcvmamd*, 1011 - *idfdcvmvar*, 1100 - *idftcvmamd*, 1101 - *idftcvmvar*;
(v) $\langle i_{12}, i_{13} \rangle$: type of term correlation matrix used in CVM indexing (*cvmtype*): 00 - *prob*, 01 - *prob0diag*, 10 - *intu*, 11 - *intu0diag*; (only necessary when CVM information is used in the transformation functions or global term weights).
(vi) $\langle i_{14}, \ldots, i_{18} \rangle$: weight given to the expert in the combination (between zero and 32).

The individuals in the population are the concatenation of k strings encoding k particular experts. The number of experts used, the parameter k, has to be specified by the user. In the experiments we tried combinations of up to four experts. With this gene encoding there may exist "ill formed" chromosomes (e.g., when $\langle i_2, i_3, i_4, i_5 \rangle$ is set to 1111, because 14 weight specifications are encoded with 4 bits). We control such chromosomes by assigning them some minimum fitness value.

In the reproduction process we used an elitist strategy, keeping the l fittest individuals to be part of the new generation. The remaining individuals were selected using a proportional selection approach following the roulette principle. The selected individuals are modified with two standard genetic operators: one-point crossover, and mutation. One-point crossover is a binary operator, which creates new individuals by taking parts of the genes from both parents. This is done by selecting randomly a crossover point and exchanging the genes between both parents up to this point. The unary mutation operator randomly changes the value of genes.

The chromosomes in the presented framework encode particular retrieval strategies for an IR system. Thus, the fitness of individuals can be computed using classical IR performance measures. We defined a first fitness function that is based on the non-interpolated average precision over all relevant documents (calculated as specified in the TREC experiments).[32] Let $\{q_1, q_2, \ldots, q_p\}$ be the set of training queries and let $avp_{RS}(q_i)$ denote the average precision value for the query q_i

obtained with the retrieval strategy RS. The fitness function is defined as follows:

$$fit_1(RS) = \frac{\sum_{i=1}^{p} avp_{RS}(q_i)}{p}. \quad (11)$$

With this definition the fitness of a chromosome is simply the average of the precision values of all training queries on the corresponding retrieval strategy.

6. Experiments

In this section, we present experimental results on four medium sized test collections that have been extensively used for evaluating IR systems. The collection characteristics are as follows: i) MEDLARS (1033 documents; 30 queries), ii) CRANFIELD (1398 documents; 225 queries), iii) CISI (1460 documents; 76 queries), and iv) CACM (3204 documents; 52 queries).

In the indexing process some preprocessing is done. First, we eliminate common functional words (stop words) by using the stop word list that comes with the SMART system at `ftp://ftp.cs.cornell.edu/pub/smart/`. Afterwards, the word stem of each remaining word is calculated using the Porter stemmer.[33] We then eliminate all those word stems that occurred just once in the collection. These word stems are poor discriminators and their elimination reduces drastically the set of considered terms. The remaining word stems build the set of index terms.

6.1. *Selecting the best strategy for each collection*

In the first set of experiments we try to optimize the retrieval strategy for each of the four collections taking all queries as a training set. We train the GA on each collection searching for the best retrieval strategy consisting of combinations of two, three and four retrieval experts. We take the relevance judgments of all queries provided with the collections to calculate the fitness of the individuals in the GA with Eq. (11).

As parameters for the GA we use a probability of 0.9 for one-point crossover and a probability of 0.1 for mutation. These values have been coarsely optimized by hand. Furthermore we use a population size of 20 and the number of chromosomes that are directly promoted to the next generation is six. The GA is stopped after 100 generations. The initial population is randomly chosen.

Table 1 shows the results obtained on the different document collections. We present the average precision values obtained with the retrieval strategy proposed by the GA. These values correspond to the non-interpolated average precision calculated on each query and averaged over all queries in the collection. To calculate these values we use the method employed in TREC.[32] The results are compared to the classical VSM with *tf-idf* weighting (denoted as *vsm*). We also present the average precision of the best single retrieval expert. We have done an exhaustive search to find the best single expert. In order to compute the statistical significance of the results with respect to *vsm* we do a T-test on paired observations. We present

the value of the T-statistic in brackets. On all four collections, a value of $T > 2$ implies statistically reliable improvements at a significance level lower than 5%.

Table 1. Non-interpolated average precision ($prec$) for the best single expert ($best_s$) and the retrieval strategies found by the GA for combinations of two, three, and four experts and % improvement w.r.t. the baseline (vsm). The values in brackets represent the T-statistic comparing the results to vsm.

Experts	MEDLARS		CRANFIELD		CISI		CACM	
	$prec$	%	$prec$	%	$prec$	%	$prec$	%
vsm	0.518		0.391		0.237		0.332	
$best_s$	0.665	28.5(5.9)	0.421	7.8(3.1)	0.263	11.2(2.9)	0.381	14.6(3.2)
2	0.672	29.8(6)	0.45	15.2(7.8)	0.284	19.9(2.9)	0.414	24.4(5.3)
3	0.675	30.3(6.1)	0.456	16.6(9.4)	0.278	17.5(4.1)	0.423	27.2(4.9)
4	0.67	29.3(6.1)	0.454	16.3(8.5)	0.279	17.9(4)	0.418	25.8(5.2)

Figure 1 shows some precision recall curves. We present the curve for the best combination of experts for each collection. The curves for vsm and the best single expert are given for comparison. The precision values are interpolated recall level precision averages. We calculate the interpolated precision at recall points $0, 0.1, 0.2, \ldots, 1$ for each query and average the results over all queries in a collection. We use the same calculation method as employed in TREC.[32]

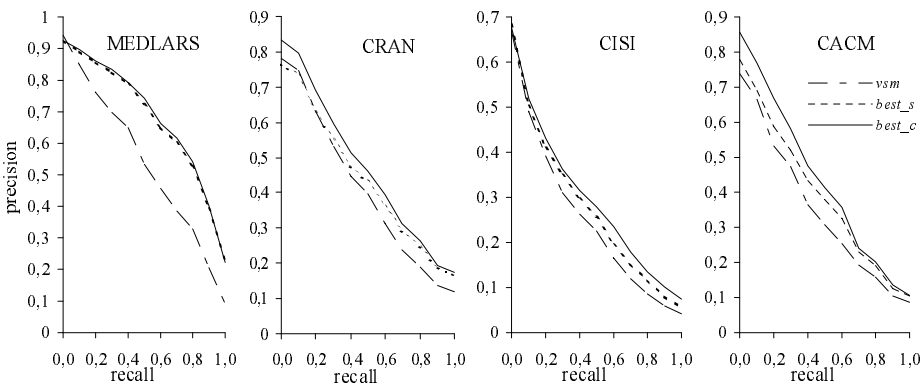

Fig. 1. Precision recall curves for the best combination of experts ($best_c$) in comparison to the best single expert ($best_s$) and vsm.

The results show that an appropriate linear combination of retrieval experts performs clearly better than using just a single retrieval method. On all four collections the learned combinations of two experts perform better than the best single variant. Moreover, the statistical significance of the results increases. The improvement is very small on MEDLARS but impressive on the other collections. Our explanation of this fact is that in MEDLARS the best single expert is already quite good and no significantly better strategy can be found when combining several experts.

Precision increases rather slowly and may even fall when more experts are being combined. The latter is due to the nature of genetic algorithms. The size of the search space increases dramatically when more experts are combined. Thus, it is harder to find an optimal solution. In fact, in the proposed encoding the search space of all combinations of four experts includes all possible combinations of one, two, and three experts, because the weights given to an expert can be set to zero. Therefore, if the precision falls when combining more experts it is because the GA did not find this or a better solution in the 100 generations that were generated.

There are different ways to overcome this problem. One could simply augment the number of generations, or one could use the last population of the GA runs with k experts as the initial population in the runs with $k+1$ experts. However, we have not used such criteria in our tests. Also, we have not analyzed combinations of more than four experts. Because of the increasing search space, genetic algorithms would probably need more than 100 generations to find better solutions for five and more experts. Because the fitness function we use is costly, this would increase the learning time. Furthermore, we believe that the obtained improvements are quite acceptable and that combining more experts will not do considerably better. In fact, a study by Vogt empirically confirms this hypothesis.[21] In his experiments, the improvements have diminished when the number of experts grows above a certain value and the optimum number has been around four.

In this set of experiments, about 90% of the selected experts are CVM variants, e.g., they used CVM indexing for documents and/or queries. Only 10% of the experts are "pure" VSM variants (using the VSM for document and query indexing). Furthermore, all of the combinations use at least one CVM expert. This empirically confirms our hypothesis that the advantages of the CVM can be best exploited by combining different CVM variants or by combining it with traditional word-matching approaches.[5]

6.2. *Reducing memory and time consumption in the CVM*

One of the limitations of the CVM is its memory and time consumption. The indexing process takes longer than in the classical VSM, because the term correlation matrix has to be generated and document context vectors have to be calculated. Furthermore, the method requires more memory because it has to store the correlation matrix and the document context vectors. With respect to the matrix, the memory requirements increase with the number of index terms. This also holds for the document context vectors, where the number of non-zero elements increases with the number of index terms. In comparison, the vectors in the classical VSM have the same size - they use the same dimensions - but they will usually have only a few non-zero elements (only the words occurring in the documents will have non-zero values).

However, the main problem regarding memory and time consumption is not in the indexing process because this can be done in a batch mode. This issue is more important in the retrieval process, where usually short response times are

required. If the document and query vectors are dense, as it is the case in the CVM, the computational cost of calculating the document/query similarities is high. Moreover, it will be more difficult to load the document indexes into memory and the use of inverted files does not improve the efficiency in a considerable way. A solution to this problem could be a dimension reduction of the vector space, as it is done in LSI or PLSI. However, such an approach changes the interpretation of the dimensions from terms to some abstract factors.

We propose two other solutions. The first one consists of "moving" the CVM calculations to the query indexing process, which is possible by doing some mathematical transformations on the equations. In this case query vectors are obtained that can be directly mapped to the *tf* document vectors using the standard inverted file approach. Although this method reduces the memory consumption for document representations, it increases the time for query indexing and still requires the use of the term correlation matrix. The second method we propose consists of eliminating the least significant elements in document, query, and term context vectors. That is, we set all but the highest elements in the vectors to zero. This does not reduce the dimensions of the vector space but generates sparser vectors similar to the classical VSM. However, it seems obvious that this reduction approach limits the power of the context vector model.

We have tested this reduction method in another set of experiments, where we keep only the 150 highest values in document, query, and term context vectors. After some experiments, we have chosen this number because it reasonably reduces the vectors and still models the term relationships sufficiently well. We only reduce the context vectors; the *tf* and *bin* based document and query indexes are not changed. Before reducing the document and query context vectors, we temporarily multiply each element with its global term weight (*idf* for document vectors; the weight specified by *qw* for query vectors). The global term weights that depend on the document context vectors are recalculated with the reduced model. Table 2 shows the results obtained when "reduced" CVM experts are used. In order to avoid biases due to the selection of the initial population we repeat each test five times, each with a different seed and, thus, with a different, randomly chosen, initial population. The presented results are averages of the five runs. In the experiments with the "full" context vectors we have not repeated the tests because the learning process took very long.

In general, different retrieval strategies are found when the GA is started with another initial population on the same collection and with the same number of experts. However, in each case the precision results are very similar for the five retrieval strategies found and, thus, the presented averages are representative.

From Tables 1 and 2 it can be seen that reducing the number of non-zero elements in context vectors leads to worse precision. This seems obvious taking into account the loss of some of the power of context vectors. We are not considering the influence of one term on another if this influence is very low. In MEDLARS, for example, the average number of non-zero elements in the document context vectors decreases

Table 2. Non-interpolated average precision (*prec*) for the best single expert ($best_s$) and the retrieval strategies found by the GA for combinations of two, three, and four experts when using "reduced" CVM experts, and % improvement w.r.t. the baseline (*vsm*). The values in brackets represent the T-statistic comparing the results to *vsm*.

Experts	MEDLARS		CRANFIELD		CISI		CACM	
	prec	%	*prec*	%	*prec*	%	*prec*	%
vsm	0.518		0.391		0.237		0.332	
$best_s$	0.659	27.2(5.7)	0.412	5.5(4.4)	0.258	9.1(2.3)	0.375	12.8(2.9)
2	0.673	29.9(5.7)	0.437	11.7(7)	0.272	14.9(3.3)	0.408	22.8(5)
3	0.674	30.1(6)	0.445	13.8(7.3)	0.271	14.5(3.3)	0.412	23.9(5.1)
4	0.67	29.4(5.8)	0.447	14.4(7.5)	0.269	13.6(3.2)	0.414	24.4(5.4)

from about 5080 in the "full" model to 150 in the "reduced" model. Nevertheless the results obtained with the reduced vectors are still much better than those obtained with *vsm*. This is quite interesting because the "reduced" CVM overcomes the problems of high memory requirements and computational cost. It is also surprising that the degradation of the "reduced" model with respect to the "full" context vector model is rather small. This is even more interesting because only about 6% of the selected experts were "pure" VSM variants; all others were variants of the CVM using either document or query context vectors, or both. Pure VSM experts were only selected in CACM (in three of the five different runs with combinations of four experts and two runs with combinations of three experts), CISI (in three runs with combinations of four experts and one run with combinations of two experts), and CRANFIELD (in one run with combinations of three experts). And in all these cases they were combined with CVM variants. This phenomenon is similar to what happens in machine learning induction; here we are reducing the overfitting problem by not considering spurious relationships among concepts. Therefore, in some sense we are increasing the robustness of the model to noise. The fact that almost always CVM experts were selected indicates that this model has advantages over the VSM, at least in adaptive data fusion approaches. Nonetheless, we have shown elsewhere that some particular CVM variants work reliably better on all four collections.[5]

With respect to the expert parameters, all different weighting schemes, query and document transformation functions, and term correlation matrixes were used in some of the selected retrieval experts. Even the *no* weight was chosen several times. Thus, none of these definitions seems to be "useless".

Table 3 compares our results with the results reported by Hofmann for LSI and the PLSI variants PLSI-U* and PLSI-Q*.[8] These are the best results we have found so far for the tested collections. The LSI results correspond to a linear combination of LSI with the VSM. Hofmann reports two variants of PLSI, PLSI-Q and PLSI-U. In PLSI-Q, documents and queries are represented in a low-dimensional factor space and in PLSI-U they are back-projected from the low-dimensional space to the original term space. Thus, document vectors in PLSI-U are a smooth version of the original term frequency vectors. PLSI-U* and PLSI-Q* denote combinations of the VSM with five different PLSI-U and PLSI-Q variants, respectively, each with a

different number of dimensions. In his paper, Hofmann reports the average of the precision values at the recall levels 0.1 to 0.9. In the table we have adjusted our results to this measure.

A direct comparison with Hofmann's results is difficult because the average precision values he reports for the baseline - he also uses the VSM with *tf-idf* weights - are much lower than the values we calculate. This holds especially for CACM, where he reports an average precision of 0.219, as compared to the value of 0.34 we use. The differences may be due to the use of different preprocessing techniques in indexing or different methods for calculating the average precision. Therefore, in order to allow a more consistent comparison, Table 3 presents the precision values of the retrieval models together with the absolute differences and the % improvements with respect to the baseline that was used in each case. We present the best combination of experts found for each collection with the "full" and with the "reduced" context vector approach. The "full" CVM approach is best compared with PLSI-U* because PLSI-U* does not reduce dimensionality and will have similar non-sparse document and query vectors. On the other hand, the "reduced" CVM approach should be compared to PLSI-Q* and LSI because the computational cost of these models are similar.

Table 3. Average precision ($prec$), absolute differences ($diff$) and relative improvements (%) w.r.t. the baseline used in each experiment (vsm_1 for Hofmann's tests and vsm_2 for ours). Comparison of the best retrieval strategies with "full" and "reduced" context vectors ($best_{full}$ and $best_{red}$), LSI, and the two PLSI variants PLSI-U* and PLSI-Q*.

Strategy	MEDLARS			CRANFIELD			CISI			CACM		
	$prec$	$diff$	%	$prec$	$diff$	%	$prec$	$diff$	%	$prec$	$diff$	%
vsm_1	0.49			0.352			0.202			0.219		
LSI	0.646	0.156	31.8	0.387	0.035	9.9	0.219	0.017	8.4	0.238	0.019	8.7
PLSI-U*	0.721	0.231	47.1	0.404	0.052	14.8	0.246	0.044	21.8	0.276	0.057	26.0
PLSI-Q*	0.663	0.173	35.3	0.401	0.049	13.9	0.244	0.042	20.8	0.283	0.064	29.2
vsm_2	0.541			0.404			0.235			0.34		
$best_{full}$	0.705	0.165	30.4	0.469	0.065	16.2	0.284	0.049	21	0.427	0.087	25.7
$best_{red}$	0.705	0.164	30.4	0.46	0.056	13.9	0.272	0.037	16.1	0.419	0.079	23.3

The best "full" CVM approach outperforms PLSI-U* in terms of average precision and absolute difference to the baseline and it is in the same range for the % improvements. This holds for all but the MEDLARS collection, where PLSI-U* performs clearly better. Regarding the "reduced" CVM approach, it obtains better precision values than PLSI-Q* on all four collections. With respect to the absolute differences to the baseline, they are higher than the values for PLSI-Q* in the collections CACM and CRANFIELD and lower in MEDLARS and CISI. In terms of % improvement, PLSI-Q* obtains better values than the "reduced" CVM approach with the exception of the CRANFIELD collection. Both CVM approaches clearly outperform LSI on CRANFIELD, CISI and CACM, and obtain similar results on MEDLARS.

6.3. *Learning performance*

In the experiments reported above, we have used a strong elitist strategy for the genetic algorithm by keeping the six fittest members in the population and using a population size of 20. These settings clearly restrict the possibilities of the GA to explore the search space. Much higher population sizes are normally used and usually only one individual is directly passed to the next generation. Such an approach, however, is unaffordable in the proposed framework because a wrapper approach is used and the fitness evaluation is costly. Nevertheless, in order to analyze the influence of the parameters on the learning performance we repeat the experiments for the "reduced" CVM with different values for the number of generations (*maxgen*), the population size (*popsize*) and the number of chromosomes directly promoted to the next generation (l). We use three different strategies: i) GA_{200} (*maxgen* = 10, *popsize* = 200, $l = 2$), ii) GA_2 (*maxgen* = 1000, *popsize* = 2, $l = 1$), and iii) GA_{20} (*maxgen* = 100, *popsize* = 20, $l = 6$). We run all three strategies on all collections and for combinations of two, three and four experts. We repeat each run five times, each with a different initial population (different random seed). The learning curves of the three strategies are compared in Figure 2 that shows the evolution of the precision of the best individual of a generation over the number of evaluated individuals when learning combinations of two, three and four experts, respectively. The curves represent the average of all five runs and all four collections. It should be noted that the individuals that are directly passed to the next generation need only to be evaluated the first time they are constructed. Therefore, GA_2 evaluates less individuals than GA_{20} and GA_{20} less than GA_{200}.

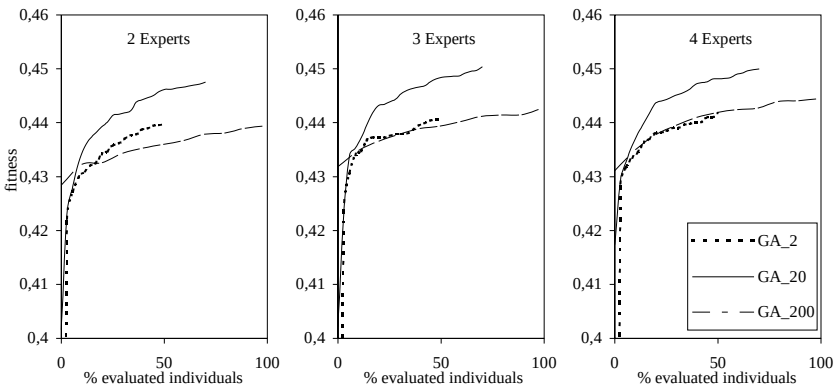

Fig. 2. Learning curves for GA_2, GA_{20}, and GA_{200}.

It can be seen that GA_{20} performs generally better than GA_2 and GA_{200}. This is somewhat interesting because GA_{20} does not take full advantage of the principles of genetic algorithms. To explain this behavior we must analyze the three strategies in more detail. In GA_2 the population consists of two individuals and only one new

individual is constructed in each generation (the other one is directly passed). The parent of this new individual is normally the best chromosome from the former generation because it has the highest probability to be selected for reproduction. The modification of this individual is limited to mutation. Therefore, GA_2 simulates a special kind of hill climbing, where the best currently found solution is slightly modified in each step. Nonetheless, also the worst of the two individuals in the population can be selected for the new generation. Thus, the algorithm can "jump" to other areas in the search space. With GA_{20} the scenario is somewhat similar. Because of the strong elitist approach, GA_{20} also searches mainly in the same region of the space. But the region is much bigger and the algorithm can "jump" more easily to other areas. GA_{200}, in comparison, performs a wide search, exploring the space in different regions at a time, but with less detail. However, GA_{200} has a higher probability to find the optimal solution when the number of generations increases.

We suspect that there are many similar suboptimal solutions in many regions of the search space. If one such region has been found it seems better to explore it instead of searching for other good areas in the space. This explains the fact that GA_2 does relatively well. On the other hand, GA_{200} spends too much effort in searching for different regions with possibly optimal solutions and does not analyze the regions in more detail. Thus, it would need more than just ten generations to find better solutions, which increases the number of individuals that have to be evaluated. GA_{20} seems to be a compromise between both techniques. It explores "good" regions already found and still searches for other areas with possibly better solutions.

GA_{20} finds the best solution in 55 of the 60 runs (five runs for four collections and combinations of two, three and four experts), whereas GA_2 found it twice and GA_{200} three times. The performance of GA_{200} increases, in comparison to the other strategies, as more experts are combined and the search space grows. This seems to confirm that genetic algorithms perform better than other search techniques on very big search spaces and "ill-behaved" problems. GA_{200} will probably outperform GA_{20} when the number of combined experts is higher, or when a higher number of generations is chosen. The "hill climbing" strategy (GA_2) is performing relatively well for two combined experts (e.g., smaller search spaces) and is less effective when more experts are combined. It performs worse than both other methods for four experts.

6.4. *Training the system*

In the previous sections we have analyzed the performance of our approach when all queries of a collection are taken as training data. This method is clearly too optimistic. In particular, it cannot be expected that the learned retrieval strategies will perform equally well on new queries. To analyze the predictive performance of the system on unseen queries we do another set of experiments using standard cross-validation techniques.

We randomly split the query set of each collection into five equally sized folds. These folds are used to define five training/test splits: in each split one fold is used as the test set (20% of the queries) and the remaining four folds define the training set (80% of the queries). That is, we use 5-fold cross validation. We then run the GA on each of the five training sets and the best retrieval strategy found is tested on the corresponding test set. We use the same parameter settings as in the previous experiments with the full query sets and the "reduced" CVM approach. That is, we run the tests for each collection and combinations of two, three and four experts, using GA_{20} and reducing the context vectors to 150 non-zero elements. Each test is repeated with five different initial populations (different random seeds).

The fitness function defined in Eq. (11) simply calculates the average of the precision values of all training queries obtained with the corresponding retrieval strategy. Using this specification, the fitness may be biased towards strategies that work very well on a subset of the training queries, but may work less well on the rest of the queries. This is because the variance of the individual query results is not taken into account. The problem of high variances in the retrieval performance on different queries is quite common in IR. Furthermore, this may be a problem whenever GA are used in a wrapper approach, as proposed here. In order to avoid this problem we define a second fitness function, which is based on the idea of statistical significance tests. We define the fitness in such a way that those individuals that produce results, which are statistically significantly better than some base line, get higher values and, thus, are considered more valuable.

For two populations X and Y and a sample of paired observations

$$(X_1, Y_1), (X_2, Y_2), \ldots, (X_p, Y_p)$$

the significance of the difference between the sample means $\overline{D} = \overline{X} - \overline{Y}$ w.r.t. the difference between the population means $\Delta\mu = \mu_X - \mu_Y$ can be tested with the T-test for paired observations, calculating the T-statistic:

$$T = \frac{\overline{D} - \Delta\mu}{S_D/\sqrt{p}}, \tag{12}$$

where S_D is the sample standard deviation (with divisor $p-1$) of the sample of differences $D_i = X_i - Y_i$, for $1 \leq i \leq p$. The null hypothesis $\mu_X - \mu_Y = \Delta\mu$ is rejected and the alternative hypothesis $\mu_X - \mu_Y > \Delta\mu$ is accepted if T is higher than a certain value (this value depends on the number of observations and the chosen significance level).

Eq. (12) builds the basis for the second fitness function (fit_2) we use. We set $T = 2$ and rearrange (12) such that $\Delta\mu$ is a function of the paired observations. Then, fit_2 is defined as:

$$fit_2(RS) = \overline{D} - 2(S_D/\sqrt{p}), \tag{13}$$

and the set of paired observations is:

$$(avp_{RS}(q_1), avp_{BL}(q_1)), \ldots, (avp_{RS}(q_p), avp_{BL}(q_p))$$

where $avp_{BL}(q_i)$ denotes the average precision for the query q_i obtained with the base line (vsm).

In the GA framework this function is maximized. This implies, in fact, that we search a strategy that maximizes the improvement over the base line but at the same time assures that the results are statistically significant and will be similar on unseen queries. The significance level is fixed with the choice of T. Choosing $T = 2$ gives a significance level of about 5% or lower if the number of observations is greater than five.

Table 4 shows the average precision results obtained with both fitness functions in the 5-fold cross validation experiments. The values are calculated as follows. First, we train the GA on each of the five training sets. Then we test the learned strategies on the corresponding test sets and obtain the non-interpolated average precision for each test query. These values are averaged over all queries in all five test sets. Thus, each query counts exactly once, when it is tested as an "unseen" query. For each collection and each number of combined experts, the values are further averaged over the five runs with different random seeds.

Table 4. Non-interpolated average precision ($prec$) on all test queries for the learned retrieval strategies (with "reduced" CVM experts) for combinations of two, three, and four experts, and % improvement w.r.t. the baseline (vsm). The values in brackets represent the T-statistic comparing the results to vsm.

Experts		MEDLARS		CRANFIELD		CISI		CACM	
		$prec$	%	$prec$	%	$prec$	%	$prec$	%
vsm		0.518		0.391		0.237		0.332	
2	fit_1	0.639	23.4(4.2)	0.429	9.8(5.8)	0.255	7.4(2.5)	0.375	12.8(2.3)
	fit_2	0.639	23.4(4.2)	0.43	9.9(6.5)	0.259	9.2(4.4)	0.372	11.8(2.9)
3	fit_1	0.64	23.6(4.5)	0.44	12.6(7.3)	0.251	6(2.1)	0.385	15.9(3.2)
	fit_2	0.635	22.6(4.4)	0.436	11.6(7.3)	0.256	8.3(3.6)	0.397	19.6(4.3)
4	fit_1	0.64	23.6(4.4)	0.44	12.6(7.1)	0.252	6.4(2.3)	0.386	16(3.4)
	fit_2	0.633	22.1(4.5)	0.439	12.2(7.5)	0.256	8.3(3.7)	0.393	18.2(3.6)

The improvement gains shown in Table 4 are generally smaller than those obtained when learning a strategy for the whole set of queries. However, they are still significant and show that adapting a retrieval system to the collection at hand can do better than using some fixed approach.

In general, the learned retrieval strategies perform better on the test queries than vsm. This holds for all 75 runs (five training/test splits $\times$ three different combinations of experts $\times$ five repetitions with different random seeds) in CRANFIELD and MEDLARS and for both fitness functions. On CISI, the learned strategies perform better than vsm for 56 runs with fit_1 and 71 runs with fit_2, and on CACM for 70 runs with fit_1 and 72 runs with fit_2. In the cases, where the performance has deteriorated, this always happens for the same training/test splits and is basically due to always the same few queries. Taking a look at these "hard" queries, for most of them there exists only one single relevant document and vsm ranks this document very high (e.g., as the first in the list) whereas the learned strategy ranks it

slightly lower. In such cases, the precision differences are very high and, moreover, these differences count strongly in the final average precision values. This means that a single query may be the cause of worse average precision on a set of queries (e.g., a test set), although the learned retrieval strategy actually performs better than *vsm* for most of the queries of this set.

The following example shows this effect in more detail. Suppose we have two retrieval strategies, RS_1 and RS_2, and five queries. Further suppose that the precision values for the queries obtained with RS_1 are $(0.1; 0.2; 0.05; 0.1; 1)$ and that the corresponding values for RS_2 are $(0.2; 0.4; 0.1; 0.2; 0.5)$. While the average precision over all queries is higher for RS_1 (0.29 versus 0.28), RS_2 clearly works better given that it doubles precision for all but the last query. Furthermore, assuming that there exists only one single relevant document for the last query, this has been ranked at the top by RS_1 and in the second place by RS_2. Thus, even for this query, RS_2 is not very much poorer than RS_1. This phenomenon is the cause of the deterioration in 25 of the 31 cases, where the learned retrieval strategy performs worse than *vsm* on the test set in terms of average precision.

Nevertheless, in the tests on CISI there were some runs on two training/test splits where the learned retrieval strategy performed poorer on the test set or the improvement gains over *vsm* are relatively small. We think that this happens because the queries in the test sets of these two splits are different from the queries in the training sets. It is well known in IR that retrieval models do not work equally well on all types of queries. Queries may ask for rather precise or for more general information and may also describe the information needs in very different ways. In this sense, if an optimal retrieval strategy has been learned for a set of training queries of a certain type, this strategy will usually not work well when other types of queries are presented.

Regarding the different fitness functions, the experiments do not consistently confirm that fit_2 performs better than fit_1. It does on CISI and CACM but not on CRANFIELD and MEDLARS. fit_2 should generally find more robust solutions and should have fewer problems with overfitting. This means that strategies learned with fit_2 should work better on types of queries that are under-represented in the training set. Thus, fit_2 should behave better on collections with more differences in the queries, in other words, where the variance of the precision values is high. This seems to be the case in CACM and CISI as the values of the T-statistics confirm (see Table 2). These values are lower than those for CRANFIELD and MEDLARS. Indeed, fit_2 works much better than fit_1 on the two "complicated" training/test splits in CISI.

fit_1, on the other hand, will work better if the test queries are very similar to the training queries. In fact, if the training queries are presented again to the system, a strategy learned with fit_1 should achieve better results as a strategy learned with fit_2. This is because theoretically the optimal solution with fit_2 will have a lower, or at most the same, average precision on the training set as the optimal solution with fit_1. In the experiments, however, this is not always the case because it cannot

be assured that a genetic algorithm will find the optimal solution.

We also test the predicting performance when using "full" CVM experts. Here, we use the same experimental setup, but because of the computational cost we only use fit_2 and do not repeat each test with different initial populations. The results are presented in Table 5. As seen before, the performance improves as compared to the "reduced" CVM approach. The gains, however, are relatively small and may not compensate for the decrease in efficiency.

Table 5. Non-interpolated average precision ($prec$) on all test queries for the learned retrieval strategies (with "full" CVM experts) for combinations of two, three, and four experts, and % improvement w.r.t. the baseline (vsm). The values in brackets represent the T-statistic comparing the results to vsm.

Experts	MEDLARS		CRANFIELD		CISI		CACM	
	$prec$	%	$prec$	%	$prec$	%	$prec$	%
vsm	0.518		0.391		0.237		0.332	
2	0.626	20.9(3.6)	0.446	14(7.1)	0.259	9.5(3.9)	0.4	20.2(5.1)
3	0.647	24.9(4.3)	0.448	14.5(7.9)	0.264	11.7(4.1)	0.403	21.2(4.6)
4	0.641	23.8(4.5)	0.445	13.9(8)	0.264	11.7(5.3)	0.397	19.4(3.7)

7. Conclusions and Future Work

In this paper, we have presented our work on learning a retrieval strategy for a given document collection using genetic algorithms. A retrieval strategy is a linear combination of a set of retrieval methods or experts. The learning process automatically chooses i) the experts to be combined, and ii) the weights given to these experts in the linear combination. In recent research, work has been going on to find indicators that determine good candidates for retrieval expert combinations.[20,21,18,19] And only the weights given to the experts are obtained in an optimization process. In our framework, no previous selection of appropriate experts is required. They are automatically chosen by the system. We think that this approach is more appropriate because otherwise good solutions may be missed.

To see the potential gains of combination approaches, we have tested the system taking all provided queries as a training set. The results empirically confirm that appropriate combinations of CVM and VSM variants perform significantly better than the classical VSM and than LSI. Their performance is similar to the PLSI model. All learned retrieval strategies contain at least one CVM expert and only very few solutions contain "pure" VSM variants. Thus, CVM variants seem to be very appropriate for combination approaches. In another set of experiments we have shown that the learned strategies generalize well to unseen queries. This empirically confirms the hypothesis that adapting a retrieval system to the collection at hand can do better than using some fixed approach.

The experiments show that appropriate combinations of retrieval systems can do better than single methods. This has also been found in most of the recent studies on data fusion. The strongest improvement gains are obtained when changing from a single expert to a combination of two experts. Improvement gains are smaller as

the number of combined experts increases. We argue that this is basically due to the following two reasons: i) it is harder to find suboptimal solutions for combinations of more experts because the search space is getting bigger, and ii) the optimal solution in each collection may require only a limited number of experts.

We have used genetic algorithms to find suboptimal retrieval strategies. They seem to be appropriate for this problem if their parameters are carefully chosen. Since we use the GA in a wrapper approach, the fitness evaluation is quite costly and the parameters should be tuned such that good solutions are found after a relatively small number of evaluated individuals. In this sense, we propose the use of a strong elitist strategy, that is, using a small population size and keeping a relatively high number of best individuals in the population. In the experiments, this approach has had the best ratio between the quality of the generated solutions and the number of evaluated individuals. However, less elitist strategies may find better solutions but at the cost of evaluating more individuals.

We have also presented a novel method for the calculation of the fitness of the individuals in a genetic algorithm. It is based on the statistical significance of the results obtained on a training set. Solutions found by a GA using this fitness function will generalize better to unseen data than solutions found with a function based on mean measures. The experiments seem to confirm this, albeit the differences are not significant and the mean based fitness function works slightly better on two collections.

The mayor limitation of the context vector model is its computational cost. This is because the document and query vectors are not sparse as in the classical VSM. To overcome this problem we have eliminated the insignificant elements in document, term, and query context vectors, keeping only 150 non-zero elements. This approach does not reduce the dimensions of the vector space, as proposed by other authors, but has the same effect on memory and time consumption. Average precision with the "reduced" model deteriorates at the benefit of more efficiency. However, the performance of the "reduced" CVM is still comparable to PLSI and the model outperforms the classical VSM and LSI.

At this stage, we wanted to assess whether the proposed approach works and adapting the system to larger databases (e.g., TREC data) has not been a mayor concern. Therefore, we have used rather small collections in the experiments, which are more manageable than bigger databases. Nonetheless, in our future work we hope to verify the results on bigger databases.

The obtained improvement gains are certainly limited to the set of experts the system can choose from and it can be expected that adding other types of retrieval models would lead to even better results. The limitations lay also in the fact that the same retrieval strategy is applied to all queries. In an optimal setting, the retrieval process should be adapted to both the document collection and the presented information need. Our proposal only treats the first of these two points. The dependency of the effectiveness of a retrieval system on the differences among queries can be avoided in a relevance feedback scenario, where the retrieval strategy

can be optimized for a single query using the relevance judgments provided by a user. However, relevance feedback approaches require the collaboration of the user, which is often not possible. Therefore, we have used the proposed technique to adapt the system to the collection rather than to the query.

In our future research, we plan to analyze another solution, one that does not require the participation of the user. We would like to classify queries in different classes such that a different retrieval strategy can be learned for different types of queries. New queries would be classified into one of the classes and the strategy assigned to that class would be selected for retrieval. For this kind of expert combination approach one needs to find indicators that are computable from the query, and that yield a classification where it makes sense to use a different (specialized) retrieval strategies for each class. Finding such indicators seems to be difficult and some researchers have tried this approach without much success.[17]

References

1. G. Salton and M. McGill, *Introduction to Modern Information Retrieval* (McGraw Hill, New York, 1983).
2. G. Salton, *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer* (Addison Wesley, Reading, 1989).
3. R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval* (ACM Press, New York, 1999).
4. H. Billhardt, D. Borrajo and V. Maojo, "Using term co-occurrence data for document indexing and retrieval", *Proc. 22nd Annual Colloquium on Information Retrieval Research (IRSG'2000)*, Cambridge, UK, April 2000, pp. 105–117.
5. H. Billhardt, D. Borrajo and V. Maojo, "A context vector model for information retrieval", *Journal of the American Society for Information Science and Technology*, **53** (2002) 236–249.
6. S. Deerwester, S. Dumais, G. Furnas, T. Landauer and R. Harshman, "Indexing by latent semantic analysis", *Journal of the American Society for Information Science*, **41** (1990) 391–407.
7. S. Dumais, "Latent semantic indexing (lsi): Trec-3 report", in *Overview of the Third Text REtrieval Conference (TREC-3)*, ed. D. K. Harman (NIST Special Publication 500-225, Gaithersburg, 1995) pp. 219–230.
8. T. Hofmann, "Probabilistic latent semantic indexing", *Proc. 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, Berkeley, CA, August 1999, pp. 50–57.
9. N. J. Belkin, C. Cool, W. B. Croft and J.P. Callan, "The effect of multiple query representations on information retrieval performance", *Proc. Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'93)*, Pittsburgh, PA, June 1993, pp. 339–346.
10. N. J. Belkin, P. Kantor, E.A. Fox and J.A. Shaw, "Combining the evidence of multiple query representations for information retrieval", *Information Processing & Management*, **31** (1995) 431–448.
11. S. Dumais, "Combing evidence for effective information filtering", *Proc. AAAI Spring Symposium on Machine Learning in Information Access*, Stanford, March 1996.
12. E. A. Fox and J. A. Shaw, "Combination of multiple searches", in *The Second Text REtrieval Conference (TREC-2)*, ed. D. K. Harman (NIST Special Publication 500-

215, Gaithersburg, 1994) pp. 243–252.
13. J. A. Shaw and E. A. Fox, "Combination of multiple searches", in *Overview of the Third Text REtrieval Conference (TREC-3)*, ed. D. K. Harman (NIST Special Publication 500-225, Gaithersburg, 1995) pp. 105–108.
14. C. McCabe, A. Chowdhury, D. Grossman and O. Frieder, "A unified environment for fusion of information retrieval approaches", *Proc. Eighth International Conference on Information Knowledge Management (CIKM 1999)*, Kansas City, November 1999, pp. 330–342.
15. C. C. Vogt, G. W. Cottrell, R. K. Belew and B. T. Bartell, "Using relevance to train a linear mixture of experts", in *The Fifth Text REtrieval Conference (TREC-5)*, eds. E. M. Voorhees and D. K. Harman (NIST Special Publication 500-238, Gaithersburg, 1997) pp. 503–516.
16. B. T. Bartell, G. W. Cottrell and R. K. Belew, "Automatic combination of multiple ranked retrieval systems", *Proc. 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, Dublin, Ireland, July 1994, pp. 173–181.
17. J. Savoy, M. Ndarugendamwo and D. Vrajitoru, "Report on the trec-4 experiment: Combining probabilistic and vector-space schemes", in *The Fourth Text REtrieval Conference (TREC-4)*, ed. D. K. Harman (NIST Special Publication 500-236, Gaithersburg, 1996) pp. 537–548.
18. C. C. Vogt and G. W. Cottrell, "Predicting the performance of linearly combined IR systems", *Proc. 21st Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, Melbourne, Australia, August 1998, pp. 190–196.
19. C. C. Vogt and G. W. Cottrell, "Fusion via a linear combination of scores", *Information Retrieval*, **1** (1999) 151–173.
20. J. H. Lee, "Analysis of multiple evidence combination", *Proc. 20th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, Philadelphia, PA, July 1997, pp. 267–276.
21. C. C. Vogt, "How much more is better? Characterizing the effects of adding more IR systems to a combination", *Proc. Sixth Conference on Content-Based Multimedia Information Access (RIAO2000)*, Paris, France, April 2000.
22. M. Boughanem, C. Chrisment and L. Tamine, "Genetic approach to query space exploration", *Information Retrieval*, **1** (1999) 175–192.
23. M. Boughanem, C. Julien, J. Mothe and C. Soule-Dupuy, "Mercure at trec8: Adhoc, web, clir and filtering tasks", in *The Eighth Text REtrieval Conference (TREC-8)*, eds. E. M. Voorhees and D. K. Harman (NIST Special Publication 500-246, Gaithersburg, 2000) pp. 431–444.
24. H. Chen, "Machine learning for information retrieval: Neural networks, symbolic learning, and genetic algorithms", *Journal of the American Society for Information Science*, **46** (1995) 194–216.
25. D. H. Kraft, F. B. Petry, B. P. Buckles and T. Sadasivan, "The use of genetic programming to build queries for information retrieval", *Proc. First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, Orlando, FL, June 1994, pp. 468–473.
26. D. Vrajitoru, "Crossover improvement for the genetic algorithm in information retrieval", *Information Processing and Management*, **34** (1998) 405–415.
27. J. Yang, R. Korfhage and E. Rasmussen, "Query improvement in information retrieval using genetic algorithms - a report on the experiments of the trec project", in *The First Text REtrieval Conference (TREC-1)*, ed. D. K. Harman (NIST Special Publication 500-207, Gaithersburg, 1993) pp. 31–58.

28. W. Fan, M. D. Gordon and P. Pathak, "Automatic generation of a matching function by genetic programming for effective information retrieval", *Proc. 1999 American Conference on Information Systems (AIS AMCIS'99)*, Milwaukee, WI, August 1999.
29. P. Pathak, M. D. Gordon and W. Fan, "Effective information retrieval using genetic algorithms based matching function adaption", *Proc. of 33rd Annual Hawaii International Conference on System Science*, Maui, Hawaii, 2000.
30. B. T. Bartell, G. W. Cottrell and R. K. Belew, "Learning the optimal parameters in a ranked retrieval system using multi-query relevance feedback", *Proc. Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94)*, Las Vegas, 1994.
31. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley, Reading, 1989).
32. E. M. Voorhees and D. K. Harman (eds.), *The Seventh Text REtrieval Conference (TREC-7)* (NIST Special Publication 500-242, Gaithersburg, 1999).
33. M. F. Porter, "An algorithm for suffix stripping", *Program*, **14** (1980) 130–137.