Adaptive Information Filtering using Evolutionary Computation

D.R. Tauritz & J.N. Kok & I.G. Sprinkhuizen-Kuyper
Department of Computer Science, Leiden University
P. O. Box 9512, 2300 RA Leiden, The Netherlands

1 Introduction

We live in what is often termed the information age. It might also be called the data age, for only relevant data is information, and finding relevant data among the ever-faster growing heaps of available data is becoming increasingly more difficult. There are two different ways in which Information Filtering can be employed [1]. One is to process new data for storage in an Information Retrieval system, the other is to provide all users with a personal information delivery system specially tailored to their personal information needs. Information Filtering is concerned with filtering data streams in such a way as to leave only pertinent data (information) to be perused. When the data streams are produced in a changing environment the filtering has to adapt too in order to remain effective. Adaptive Information Filtering (AIF) is concerned with filtering in changing environments. The changes may occur both on the transmission side (the nature of the streams can change), and on the reception side (the interest of a user can change).

Intelligent AIF requires a learning system which can adapt to a changing environment. We would like it to have such useful properties as domain independence, spelling error insensitivities, adaptability, and optimal use of user feedback while minimizing the amount of user feedback required to function properly.

In this paper we describe a case study whose purpose was to investigate whether evolutionary computation can be useful in AIF systems. In this case study we took as input articles from a fixed number of different Internet newsgroups and gave them to the system. The goal of the system is to cluster these articles in groups: that is, each article should be assigned a label which corresponds to a newsgroup. We propose an AIF system for this task based on the novel combination of weighted trigram analysis, incremental clustering, and evolutionary computation.

A trigram is a combination of three symbols. Trigram analysis [3] consists of determining the frequency distribution of all trigrams in a textual document. An enhancement to trigram analysis is the assignment of weights to all the trigrams indicating their relative importance for discriminating between documents on different topics. This ‘weighted trigram analysis’ was used in the AIF system described here.

An incremental clustering algorithm is applied to weighted trigram representations of the documents creating a classification of the documents. An incremental clustering algorithm is required because AIF is a dynamic process. In incremental clustering the number of clusters is not determined in advance, and can change over time [6]. Also the prototype vectors corresponding to the clusters can move over time.

To find the right weights for the trigram analysis, we designed an evolutionary algorithm. The most complex step in this AIF system is finding the (near) optimal weights for the trigrams: even using only the 26 letters of the Latin alphabet and the space as delimiter symbol, that still leaves 19683 (27^3) weights to optimize. This optimization problem is user dependent, so it cannot be performed a priori, and must also be able to adapt to the changing information needs of the user.

2 The evolutionary algorithm

The evolutionary algorithm is required to work incrementally. This has two important consequences. First, the fitness of a trial solution cannot be calculated, only statistically approximated over time. And, secondly, as the environment changes, the fitness of a trial solution may change too. In other words, the fitness being estimated over time will probably change over time, further complicating the task.

The elements of the population are vectors of weights. The weights are represented by integers. The fitness of an element at any given time is defined by dividing its score through its age. The age is defined
by the number of articles that have been evaluated by the element of the population up to that moment. The score is the number of times that it correctly classified an article. This means that the score is never larger than the age, which results in the welcome property that the range of the fitness values is limited to the interval between zero and one. The value is zero when the member in question has incorrectly clustered all documents processed so far, and one when the member has made no mistake in clustering as yet. This fitness function also allows the fitness of members of various ages to be meaningfully compared.

The reliability of the fitness value of an element at a particular moment depends on the age of that element. This is, however, not reflected in the fitness function. The solution adopted was to split the population into two pools, one for the new unproven members with ages below a certain value, and one for the adults. For each member of the first pool the fitness is only an estimation. This takes some time, and only if a member has reached a certain age, it is removed from this pool and put in the pool of adults. Hence one can view the first pool as the waiting room for the second pool. A fixed number of children is produced, but never more than there were members in the adult pool. They are not put directly in the adult pool, but have to wait some time until their fitness estimation is more reliable. We worked with a fixed size of the total population, so the worst adults are removed from the population in order to add the children. So both pools of children and adults can vary in size, but their combination is always reduced to the original population size.

This strategy seems to work quite well, although the question remains what the connection is between the age distribution of the population and the statistical reliability of the fitness values of the members. While advanced age certainly indicates high reliability, it is certainly not so that this can simply be reversed. This is because the children are slight mutations of the better performing adults and, therefore, can be expected to, in general, perform quite well themselves.

The only genetic operator used is mutation. We tried also 0.5-uniform crossover; no benefits were found. A single application of the mutation operator causes Gaussian noise (mean zero, unit variance) to be added to all the vector elements of the population member being mutated. Selection of a parent from the adult pool is implemented as follows. First the member with the highest fitness gets a chance to be selected. Selection is performed by choosing a random number between zero and one and comparing it with the selective pressure parameter. If it is smaller the current member is selected, otherwise the member with the next highest fitness becomes the current member and the process is repeated by again choosing a random number between zero and one. Applying the mutation operator one time to the selected parent creates a new member which is added to the child pool, and after reaching the age threshold is moved to the adult pool. The values for system parameters such as population size, selective pressure parameter, age threshold and replacement rate (the number of adults which are replaced with children for each evaluation) were chosen after trying a number of random values in test runs. The population size was 100, selective pressure parameter 0.1, age threshold 10 and replacement rate 5 (which means the population stabilizes at 50 percent adults and 50 percent children).

3 Incremental Clustering

We use a clustering algorithm that creates prototype vectors for each newsgroup. Given a weight vector \( w \), and prototype vectors for each newsgroup, a document is assigned to a newsgroup as follows. A determination is made as to which prototype vector is closest to the vector constructed from the document. The label of this prototype vector is then considered to be the newsgroup. The distance depends on the weight vector \( w \) as follows: \( d(x, y) = \sqrt{\sum_i (w_i (x_i - y_i))^2} \).

The experiments were initialized by creating the start clusters through computing document vector averages of the first \( n \) documents (values for \( n \) ranged from 10 to 30).

We keep one set of prototype vectors. These vectors change during the execution of the evolutionary algorithm as follows. After each new document, the evolutionary algorithm yields a new adult pool of weight vectors. The weight vector with the best fitness is used to change the prototype vectors. The document belongs to a newsgroup, and the closest prototype vector of that newsgroup is moved towards the document such that the distance between the document and the closest prototype vector becomes smaller. Note that the distance depends on the weight vector. This learning of cluster centers is similar to the learning of centers in Kohonen networks [5] in case of a fixed number of vectors, and is similar to learning in ART networks [2] for a variable number of vectors.

4 Results of Experiments

The success of the AIF system depends on two assumptions. First, that the articles retrieved from a specific newsgroup should show greater similarity to each other than to articles from the other newsgroups. And secondly, articles belonging to a particular newsgroup should be suited to be clustered together. To
approximate these assumptions as accurately as possible newsgroups were chosen for the experiments which were moderated. Moderated newsgroups are newsgroups for which articles cannot be posted freely, but must be approved by a human moderator who tries to filter out non-relevant postings.

Figures show the results of experiments. The graphs are offered in sets showing the average fitness and the highest fitness of the population. The x-axis indicates the number of articles being presented, the y-axis the average or highest population fitness of the adults. Contrary to the situation for most evolutionary algorithms, the evolutionary algorithm is used to approximate a function that changes over time. What we are looking for is not a best approximation at a certain point in time, but a sequence of good approximations. For the graphs in the Figures this means a situation with a sequence of high fitness values, without too many “dips”.

In Figures 1 and 2 the results of the two newsgroup experiment are shown. The task was to split the newsgroups misc.news.bosnia (moderated news about Bosnia) and misc.news.southasia (moderated news about south Asia). After presenting the training set (200 articles) about ten times the system converges to an average accuracy rate in excess of 90 percent, the highest fitness to around 95 percent (Figures 3 and 4).

Figures 5 and 6 show the results of a four newsgroup experiment. The task was to split the newsgroups misc.news.bosnia, misc.news.southasia, comp.os.os2announce, and comp.lang.java.announce (moderated announcements concerning the programming language Java). After presenting the training set (200 articles) about fifty times, the system stabilizes at accuracy rates in excess of 90 percent, with the exception of occasional dips. Notice that the time to reach this accuracy level is quite a bit higher than with the previous experiments which involved fewer
newsgroups. The final experiment was to determine if the system is capable of generalizing from what it has learned during the processing of a training set, and applying that knowledge to separating a test set. To this end the weight vector files produced by the three newsgroup experiment were taken as starting point, and then a test set (200 articles) offered, consisting almost entirely of new documents retrieved from the same newsgroups as used in the three newsgroup experiment. The results in Figure 7 show that the accuracy is comparable to that of the training set (note that Figure 7 has a higher resolution than the earlier figures).

5 Conclusion

An AIF system based on our approach is capable of accurately separating a dynamic stream of documents. Furthermore, the experiments with a varying number of clusters indicate that increasing the number of clusters only effects the time needed for the accuracy rate to stabilize, not the value at which it stabilizes. This indicates that the system is scalable. Also, the experiment with a test set seems to indicate that after sufficient training the system is capable of processing untrained documents with an accuracy rate comparable to that of processing the trained documents. This means that the system successfully generalizes.

A problem that needs investigation is the occurrence of "dips" in the accuracy level. These are probably in part due to the varying quality of the documents being processed, but randomness also seems to play a role, possibly indicating an instability due to the population getting stuck in local optima from time to time. A possible solution might be to make the amount of mutation dependent on the fitness of the member being mutated.

Further research is needed to conclusively demonstrate the scalability and generalizing power of this system. We can allow a more refined search of the solution space if the weight representation could be changed from integers to reals. Finding the right system parameters is difficult. One solution is to allow self-adaptation of the system parameters by encoding them in the trial solutions. Until now we used one prototype vector for each cluster. In the future we might extend this with more prototype vectors per cluster. It is interesting to see that trigram analysis is not restricted to text: also images and digital audio can be considered. For example, in image analysis trigrams of values of neighboring pixels can be used [4]. Hence, a major advantage of our method is that it generalizes to multi-media documents. We plan also to investigate this further.

References


