01 Jan 2000

Adaptive Information Filtering: Evolutionary Computation and n-gram Representation

Daniel R. Tauritz
Missouri University of Science and Technology, tauritzd@mst.edu

Ida G. Sprinkhuizen-Kuyper

Recommended Citation

D. R. Tauritz and I. G. Sprinkhuizen-Kuyper, "Adaptive Information Filtering: Evolutionary Computation and n-gram Representation," *Proceedings of the 12th Belgium-Netherlands Artificial Intelligence Conference, 2000*, pp. 157-164, Jan 2000.

Adaptive Information Filtering: evolutionary computation and n-gram representation[1]

Daniel R. Tauritz [a] Ida G. Sprinkhuizen-Kuyper [b]

[a] Leiden Institute of Advanced Computer Science, Universiteit Leiden, P.O. Box 9512, 2300 RA Leiden, The Netherlands
[b] Institute of Knowledge and Agent Technology, Universiteit Maastricht, P.O. Box 616, 6200 MD Maastricht, The Netherlands
`dtauritz@liacs.nl, kuyper@cs.unimaas.nl`

Abstract

Adaptive Information Filtering (AIF) is concerned with filtering information streams 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 interests of a user can change). The research described in this paper details the progress made in a prototype AIF system based on weighted n-gram analysis and evolutionary computation. A major advance is the design and implementation of an n-gram class library allowing experimentation with different values of n instead of solely with 3-grams as in the past. The new prototype system was tested on the Reuters-21578 text categorization test collection.

1 Introduction

Information Filtering (IF) is the process of filtering data streams in such a way that only particular data are preserved, depending on certain information needs. The IF environment is the combination of data stream and information needs. When the data stream and the information needs are changing over time, the IF environment is dynamic, and an Adaptive Information Filtering (AIF) system is called for. An AIF system is capable of adapting to changes in both the data stream and the information needs. Our main goal is to build an AIF system that classifies incoming data in clusters based on the long-term needs of a specific user. Some clusters will be very important to the user, some less, and some will be discarded altogether.

One of the essential ingredients in any IF system is its ability to match a profile with the documents available for perusal. While, optimally, a semantic match should be performed, that is not currently feasible and we have to be satisfied

[1]This research is funded in part by a NWO grant for the Adaptive Information Filtering research project under the DISH theme. The project web site is `http://www.liacs.nl/home/dtauritz/aif/`. This study is also performed as part of the NWO ToKeN2000 project which studies the access and disclosure of knowledge in The Netherlands.

with a syntactic match. The most widely employed syntactic representation of textual documents is based on term indexing [4] (keywords, frequency of terms).

Another approach is based on the so-called n-gram analysis [2]. The n stands for a positive integer. The application of n-gram analysis produces an n-gram frequency vector which comprises the frequencies of all the distinct character combinations of length n. In 1-gram analysis the occurrence of single letters is determined, in 2-gram analysis that of pairs of letters, in 3-gram analysis that of triplets, etc.

The use of n-gram analysis has many advantages over term-based systems. It is more robust when dealing with spelling variations or errors and does not require linguistic preprocessing which facilitates the deployment of n-gram-based systems in multi-topic/multi-language environments [1]. However, also an n-gram-based system can potentially benefit from preprocessing, since, for example, when the stop word 'the' is removed, the 3-gram 'the' becomes of significance. A special advantage of n-grams in dynamic environments is the ease at which they can automatically be added.

A prototype AIF system based on *weighted* 3-gram analysis (each 3-gram is assigned a weight indicating its relative importance) was introduced in [7]. For $n < 3$ n-gram analysis does not provide sufficient syntactic information [5] and for $n > 3$ sparse representations are required which are employed by the new prototype system introduced in this paper.

A crucial step in working with weighted n-gram analysis is to find the right weight vector. Our first prototype AIF system introduced a novel two-pool evolutionary algorithm (EA) for optimizing weight vectors. EAs are a class of optimization algorithms which come in handy when no a-priori solutions to a specific optimization problem are available. They work by evolving a population of trial solutions using techniques inspired by evolutionary biology. For an introduction to evolutionary computation (EC) see [3].

A new prototype AIF system based on the improved matching technique has been constructed. This paper describes the new system and presents the results of testing it on the Reuters-21578 text categorization test collection. Using a standard test collection will facilitate comparing these results with other case studies. The Reuters collection has embedded tags indicating common usage in text categorization tests. Unfortunately they were not suitable for our purposes and this prevents our results from being compared to previous studies which did employ those tags. However, as the collection is readily available and later in this paper we describe how we obtained the training and test sets for our research, comparison studies can be made. Another standard test collection for IF is the one employed by the TREC conference series[2].

2 Overview of the AIF system

The core of the system is the clustering cycle (see figure 1). The clustering algorithm uses a weight vector to compare the n-gram distribution vector of a docu-

[2]`http://trec.nist.gov`

ment with the prototype vectors of the clusters and decides in which cluster the document will be classified. The prototype vectors are initialized by averaging the n-gram distributions of a number of documents belonging to each cluster (class).

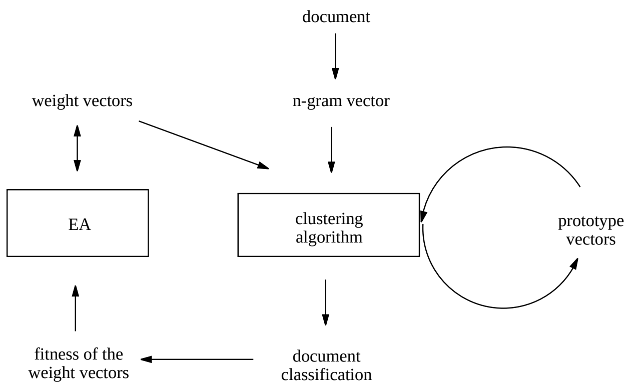

Figure 1: Schematic overview of the adaptive IF system.

The weight vector is determined by the EA. The EA works on a population of individuals, each containing a chromosome with genes composed of the components of the weight vector. The fitness of an individual is determined by dividing the number of documents it has correctly classified by the total number of documents it has classified.

3 System components

The following subsections discuss the different components of the AIF system as depicted in figure 1.

3.1 Representation and metrics

Consider two documents, A and B. We want to determine their semantic similarity. While this is not yet possible, we can estimate it by applying a metric to syntactic representations of the documents. The syntactic representation that we use is the n-gram frequency distribution[3]. Given that a is a token alphabet with size $|a|$, we define an n-gram as a sequence of n tokens belonging to a. By determining the relative frequencies at which n-grams occur in a sequence of tokens, one obtains the n-gram frequency distribution representation of that sequence.

Let $A_f = (A_{f_1}, \cdots, A_{f_t})$ and $B_f = (B_{f_1}, \cdots, B_{f_t})$ with $t = |a|^n$ be the corresponding n-gram frequency distribution vectors for these documents. While one could apply a metric directly to the normalized n-gram frequency distribution vectors of A and B to estimate their semantic similarity, a better estimate can be obtained by applying a metric to normalized weighted n-gram frequency distribution vectors [6, 7, 8, 9].

[3]For more information regarding n-grams see the n-gram clearinghouse at `http://www.liacs.nl/home/dtauritz/ngram/`.

Let $w = (w_1, \cdots, w_t)$ with $w_i \geq 0$ be the weight distribution vector giving the relative importance of the different n-grams. The weighted n-gram vectors $A_{wf} = (A_{wf_1}, \cdots, A_{wf_t})$ and $B_{wf} = (B_{wf_1}, \cdots, B_{wf_t})$ are defined as follows: $A_{wf_i} = A_{f_i} \cdot w_i$ and $B_{wf_i} = B_{f_i} \cdot w_i$ for $i = 1, \cdots, t$. Normalization is accomplished by introducing $\overline{A_{wf}} = (\overline{A_{wf_1}}, \cdots, \overline{A_{wf_t}})$ with $\overline{A_{wf_i}} = A_{wf_i} / \sum_{j=1}^{t} A_{wf_j}$ and $\overline{B_{wf}} = (\overline{B_{wf_1}}, \cdots, \overline{B_{wf_t}})$ with $\overline{B_{wf_i}} = B_{wf_i} / \sum_{j=1}^{t} B_{wf_j}$.

The metric we will use is the Manhattan metric:

$$\rho(\overline{A_{wf}}, \overline{B_{wf}}) = \sum_{i=1}^{t} |\overline{A_{wf_i}} - \overline{B_{wf_i}}| \tag{1}$$

3.2 Evolutionary Algorithm

In our AIF system the classification of a document is dependent on the weight distribution vector being used. We determine this vector by using an evolutionary algorithm (EA). In [8, 9] we showed a detailed derivation of the classification EA (CEA) we have developed and the simplified form we implemented for our AIF system. This section will describe the algorithm as implemented.

Object space consists of the objects to classify. For short σ will stand for an object and $c(\sigma)$ for the class σ maps to. In our system the objects are textual documents. As this is a discrete temporal process in Information Filtering we indicate the current iteration with τ. As we are trying to optimize the weight vector used by the classification algorithm, the members of the CEA population are weight vectors. The population is indicated with P and the members with P_i where i is ranging from 1 to *pop_size* (the size of the population). The fitness of a member should reflect how well it does in classifying. In the best case it always classifies correctly, in the worst case always wrongly. We quantify this by assigning score and age attributes. The score of a member is the number of correctly classified documents and will be indicated with P_i^{score}. The age of a member P_i^{age} is the total number of documents it has classified. The fitness is defined as follows:

$$\mathit{FITNESS}(P_i) = \frac{P_i^{score}}{P_i^{age}} \quad . \tag{2}$$

Note that the range of the fitness is from zero to one. As the age of a member increases, so does its statistical reliability in approximating the *true* fitness of a member. If, when producing offspring, the new member's score and age are set to zero, as opposed to basing them on those of its parent(s), its statistical reliability plunges and time is needed to recover some measure of reliability. In that case it is necessary to prevent the new member from participating in the evolution process until it *matures*. This is accomplished by splitting the population of trial solutions into two pools, namely a child pool P^c and an adult pool P^a with $P = P^c \cup P^a$, $|P^c|$ the number of members in P^c, $|P^a|$ the number of members in P^a and *age_threshold* the age at which members are moved from P^c to P^a.

Two essential components of any CEA are the evaluation of all the population members and, based on that, the evolvement of the population. The evaluation component will be denoted with $EVAL(\sigma_\tau, P)$ and the evolvement component will be denoted with $EVOLVE(P)$. The evaluation component is defined as follows:

$$\begin{aligned}&EVAL(\sigma_\tau, P) : \forall P_i \in P :\\ &\quad P_i^{age} \leftarrow P_i^{age} + 1, P_i^{score} \leftarrow P_i^{score} + RESULT(\sigma_\tau, P_i)\\ &\quad \text{and compute } FITNESS(P_i)\end{aligned}$$

The result of classifying an object given a trial solution is either zero (incorrect) or one (correct). The result function is defined as follows:

$$RESULT(\sigma, P_i) = \begin{cases} 0 & \text{if } CLASSIFY(\sigma, P_i) \neq c(\sigma) \\ 1 & \text{if } CLASSIFY(\sigma, P_i) = c(\sigma) \end{cases} \tag{3}$$

where $CLASSIFY(\sigma, P_i)$ = the class σ maps to using P_i.
There are two evolvement algorithms, one with crossover (resulting in two children produced by two selected parents) and one without crossover (resulting in one child which is a copy of the selected parent). In both algorithms the generated child(ren) are mutated (see below) and the weakest adult(s) is (are) removed in favor of the generated child(ren). The form of crossover employed is uniform crossover, in which each gene of a child has an equal chance to come from either parent. Mutation is performed by adding with a certain probability Gaussian noise to the genes of a member. Parent selection is done by selecting fitter members with an exponentially higher probability; this causes selective pressure. Taking this all together we arrive at algorithm 1.

Algorithm 1 AIF two pool

```
$\tau \leftarrow 1$, initialize prototype vectors
initialize $P^c$
repeat forever
    $EVAL(\sigma_\tau, P)$
    if ($|P^a| > 0$) $EVOLVE(P^a)$
    $\forall P_i \in P^c$: if ($P_i^{age} = age\_threshold$) move $P_i$ from $P^c$ to $P^a$
    $\tau \leftarrow \tau + 1$
end
```

3.3 Classification

In order to concentrate on the effect the CEA had on the weight vectors, we used a very simple classification algorithm without any parameters. The algorithm calculates the distance between a document and all the classes and classifies the document as belonging to whichever class is closest. The classes are represented by their class center. The prototype vectors representing the class centers are initialized by calculating for each the average of a certain number of n-gram frequency distribution vectors known to belong to that class. The distance is calculated by applying equation 1 to the normalized weighted n-gram frequency distributions (see subsection 3.1) representing the document and the classes.

4 System parameters

Weights are implemented using doubles ranging from 0 to 1. System performance is expressed in correct classifications per document and ranges from zero for all documents classified incorrectly to one for a perfect classification record. In order to accurately measure the performance of the system thousands of documents need to be classified. The $c(\sigma)$'s should be provided via user feedback. Until the system is ready for trial deployment, however, it will be necessary to simulate this user feedback. One way this can be accomplished is by employing a test set of documents for which the $c(\sigma)$'s are known. The initialization of the population is done by setting the scores and ages to zero. The weights are either initialized to a user specified value or assigned randomly (either using a specified seed or deriving the seed from the hardware timer). The definable experiment parameters for the AIF system are as follows. For the CEA a researcher can specify the size of the population, the age threshold, the number of adults to replace after each evaluation, the selective pressure rate (a value between 0 and 1; e.g. 0.1 means a 0.1 chance to select the fittest member, a $0.9 * 0.1$ chance to select the second fittest member, a $0.9^2 * 0.1$ chance for the third fittest member, etc.), crossover (enabled/disabled), the chance that a gene will be mutated and the amount of Gaussian noise used during mutation. Note that after $2 * \mathit{age\_threshold}$ generations, the size of the child pool is $\mathit{age\_threshold} * \mathit{offspring\_size}$, under the condition that that is not larger than *pop_size*. So, for example, if the size of the population is 100, the age threshold 10 and the *offspring_size* is 4, then after 20 generations the child pool will stabilize at size 40 and the adult pool at size 60. For each experiment the user can further specify the number of vectors used for averaging during the initialization of the prototype vectors, the class topics, the number of passes for the trainingset and the size of both the training and the test sets.

5 Experiments

As mentioned in section 4 we are currently using a simulated environment to test our system. Note that while our system is fully designed to be adaptive, and when classifying a document no knowledge about future documents is used, our simulated environment is currently static to simplify testing.

We experimented using the Reuters-21578 text categorization collection. The documents in this collection appeared on the Reuters newswire in 1987. The collection is downloadable from David D. Lewis' professional home page[4]. The documents are in SGML format and tagged for the purpose of splitting into training and test sets as used in published studies concerning text classification. For our purposes a subset of the collection was needed. First of all it was required that a document be indexed with only one topic. This limited the subset to 9494 documents. And, secondly, it was required that the document be a regular text document. This further limited the subset to 8654 documents. From that subset only those documents belonging to the ten most frequently occuring topics in the

[4]`http://www.research.att.com/home/lewis`

Table 1: Subset of Reuters-21578 used in experiments

tag	size
acq	2125
coffee	114
crude	355
earn	3735
interest	211
money-fx	259
money-supply	97
ship	156
sugar	135
trade	333

Table 2: Test set system results (percentage correctly classified)

Topics	Unweighted	Weighted
Coffee, trade	99.0	98.0
+ crude	94.7	97.6
+ money-fx	90.5	94.0
+ sugar	90.4	94.2
+ money-supply	84.3	89.5
+ ship	81.9	87.4
+ interest	79.8	84.8

subset, as listed in Table 1, were employed.

For the purpose of n-gram analysis, a document is treated as a string of characters. Letters are handled case-insensitive and all other characters are interpreted as the space character. Any sequence of spaces is replaced by a single space. Therefore the token alphabet consists of 27 characters, namely 'a' through 'z' and the space delimeter. So for $n = 3$ the number of distinct n-grams is $27^3 = 19683$. As n increases the number of distinct n-grams increases exponentially.

We did experiments using a growing number of the topics in Table 1 and employing the following parameters: $n = 3$, documents to average = 30, trainingset size = 30, testset size = 30, number of passes = 20, crossover = true, random seed: timer, $pop\_size = 200$, $age\_threshold = 25$, $offspring\_size = 2$, selective pressure = 0.1, gene mutation chance = 0.5 and deviation = 0.2. A small keyword stop list was also employed. The results are given in Table 2. The first column lists the test set system results without using weights and the second column the test set system results with weights averaged over five runs. The results show that for a small number of topics even unweighted n-gram analysis performs reasonably well, but when the number of topics is increased, the superiority of weighted n-gram analysis is clearly demonstrated. Comparing these results with those reported in [8, 9] shows that the use of a keyword stop list definitely improves the effectiveness of n-gram analysis.

6 Conclusions

In this paper we described a complete revision of the prototype AIF system introduced in [8, 9]. From the results presented in section 5 we can draw a number of conclusions. First of all, while maintaining the accuracy of the old system, the performance has been greatly improved by the use of advanced sparse representations and optimized associated functions. This has also significantly reduced the memory requirements, allowing experiments with both more topics and larger EA populations to be run. And as part of the rewrite, the system can now handle researcher defined values of n instead of being hardcoded for 3-grams.

The major problem that needs to be tackled now is generalization. The trainingset results (not listed) are very encouraging, but the testset results are a bit disappointing. Especially the lack of improvement in the testset results when the number of trainingset runs is increased needs to be addressed, before we can move to a dynamic simulation environment.

Further down the line we plan to investigate other values of n, user defined token alphabets and more advanced clustering algorithms which will be able to add new clusters and in which each cluster would have an independent radius. Also, we hope to be able to adapt our system for use with the test set provided by the TREC conference series.

References

[1] William B. Cavnar. N-gram-based text filtering for trec-2. In D.K. Harman, editor, *Overview of the Second Text REtrieval Conference (TREC-2)*, volume 500-215 of *Special Publications*, pages 171–179. National Institute of Standards and Technology (NIST), 1994.

[2] T. de Heer. The application of the concept of homeosemy to natural language information retrieval. *Information Processing & Management*, 5(18):229–236, 1982.

[3] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, third edition, June 1996.

[4] C.J. van Rijsbergen. *Information Retrieval.* Butterworths, second edition, 1979.

[5] S. Schmidt and B. Teufel. Full text retrieval based on syntactic similarities. *Information Systems*, 13(1):65–70, 1988.

[6] Johannes Cornelis Scholtes. *Neural Networks in Natural Language Processing and Information Retrieval.* PhD thesis, Universiteit van Amsterdam, 1993.

[7] Daniel R. Tauritz, Joost N. Kok, and Ida G. Sprinkhuizen-Kuyper. Adaptive information filtering using evolutionary computation. *Information Sciences*, 122(2–4):121–140, February 2000.

[8] Daniel R. Tauritz and Ida G. Sprinkhuizen-Kuyper. Adaptive information filtering algorithms. In David J. Hand, Joost N. Kok, and Michael R. Berthold, editors, *Advances in Intelligent Data Analysis, Third International Symposium, IDA-99*, volume 1642 of *Lecture Notes in Computer Science*, pages 513–524. Springer-Verlag, 1999.

[9] Daniel R. Tauritz and Ida G. Sprinkhuizen-Kuyper. Adaptive information filtering: improvement of the matching technique and derivation of the evolutionary algorithm. Technical Report 99-04, Leiden University, 1999.