A Genetic Algorithm for Dynamic Modelling and Prediction of Activity in Document Streams

Lourdes Araujo
Dpto. Lenguajes y Sistemas Informáticos
ETSI Informática
Universidad Nacional a Distancia, Madrid 28040
lurdes@lsi.uned.es

Juan Julián Merelo
Dpto. Arquitectura y Tecnología de Computadores
ETS Ingenierías Informática y Telecomunicaciones
Universidad de Granada, Granada 18071
jmerelo@geneura.ugr.es

ABSTRACT

This paper presents an evolutionary algorithm for modeling the arrival dates of document streams, which is any time-stamped collection of documents, such as newscasts, e-mails, scientific journals archives and weblog postings. The goal is to find a frequency curve that fits the data circumventing the unavoidable noise. Classical dynamic programming algorithms are limited by memory and efficiency requirements, which can be a problem when dealing with long streams. This suggests to explore alternative search methods which although do not guarantee optimality, are far more efficient. Experiments have shown that the designed evolutionary algorithm is able to reach high quality solutions in a short time. We have also explored different approaches to infer whether new arrivals increase or decrease *interest* in the topic the document stream is about. In particular, we present a variant of the evolutionary algorithm, which is able to very quickly fit a stream extended with new data, by taking advantage of the fit obtained for the original substream. These mechanisms can be used for real time detection of changes in the trend of interest in a topic, an important application of this kind of models.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic methods

General Terms

Algorithms

Keywords

Online text streams, evolutionary algorithms, event stream modelling, buzz detection

GECCO '97, July 7-11, 2007, London, England, United Kingdom

1. INTRODUCTION

The analysis of information flow has become a critical task for many organizations. Constantly evolving websites, repositories of news, e-mails, chat logs, and scientific papers are some clear examples of streams whose interpretation highly depends on the sequence of occurrence of the documents that constitute it. All these examples have a temporal dimension that has to be taken into account when analyzing their content.

One of the first attempts to model the dynamic component of a stream of documents was the Topic Detection and Tracking (TDT) research project [2, 1, 10], which among other contributions, attempted to settle the different tasks that can be distinguished in identifying topics in documents streams. First of all, a distinction is established between *topic*, i.e. a general concept, and *event*, i.e. an occurrence of a topic in a particular time. Among the distinguished tasks are the selection of *new events*, i.e. to identify the occurrence of a document discussing a new event; the *tracking* of a detected event, i.e. to identify a collection of documents about the same event, and the *segmentation* of the large streams of documents in substreams related to different topics. Different kinds of techniques have been applied to perform these TDT tasks, which involve both, content similarity analysis and temporal analysis methods.

More recent papers have focused in identifying time segments in which the appearance of documents of a particular topic can be considered relatively stable. The frequency of occurrences can be very noisy, and this hinders the identification of intervals of similar frequency. Different statistical techniques[8, 3, 7] have been applied to analyze temporal changes in document streams.

Another approach is to focus on studying the rise and fall of frequencies to detect trends in streams by identifying changes in the frequencies along given periods of time. Charikar et al. [4] have proposed an algorithm for finding the most frequent elements in a stream, which is also adapted to find elements whose frequencies change the most.

All these papers are reviewed by Kleinberg [9], who also discusses different approaches to the problem.

In this work we are interested in identifying the trends in streams of documents in a robust and efficient way. This is a task of great interest for newsmakers, and the society at large: when large amounts of data are available, it is difficult to answer the question *What is everybody talking about?* Therefore, our problem has some common ground

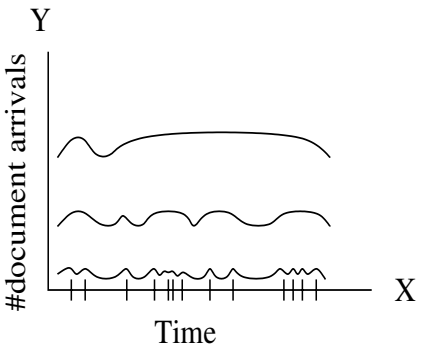

Figure 1: Some possible fitting curves for a stream of documents.

with one of those tackled in [4], although in this case we focus on the relative change of frequencies of a single and isolated topic, not different topics as above. Instead of adopting the approach of analyzing the amount of change for fixed periods of time, we model the flow in a segment of time for which enough data are available, following Kleinberg's approach [8].

The model is as follows: let us assume a stream of N documents arriving along a period of time T, and a set of S frequencies, ranging between 0 and 1. We have a double goal: determining the most appropriate frequency for each interval between one arrival and the next one and grouping together intervals with "similar" frequencies, i.e. relatively stable, in order to wash out the noise.

Figure 1 shows different possible fitting curves for the frequencies of a sequence of documents which have arrived at the instants of time marked in the X axis. If the chosen probabilistic model does not penalize changes of state, the optimum curve would be the one which assigns to each interval the most appropriate frequency, which, in general, amounts to a change of state for each arrival. In this case, the chosen fit for the arrival marked in the X axis in Figure 1 would be the lowest one. However, we want to assign the same state to consecutive intervals with similar frequency, ignoring in this way the changes produced by the randomness in the arrivals. This can be done by penalizing the change of state in different manners.

Kleinberg [8], whose method has inspired this work, has developed a framework which formalizes these ideas. He proposes a probabilistic automaton to model the frequency of appearance of documents in different intervals. The state of the automaton at a particular time determines the expected frequency of document occurrences, while transitions between states are probabilistically modeled [5]. A burst of documents for a topic can therefore be identified by the period in which the automaton has stayed in a high frequency state. Given a stream of documents related to a particular topic, the Viterbi dynamic programming algorithm [6], can be applied to determine the sequence of automaton states which optimizes the measure defined by the chosen probabilistic model. This algorithm was initially designed to find the most probable path in a Markov chain which produces a given sequence of tags, and it is now widely applied to a large range of problems [11].

The streams of document dates to model can be obtained from different environments, though in many cases they are long sequences of documents appearing along unlimited periods of time, such as chats lots, e-mails and news stories. Since a dynamic programming algorithm is limited by memory and efficiency constraints, as the length of the stream or the number of automaton states grows, it makes sense to explore alternative search methods where a degree of uncertainty is allowed in order to achieve tractability. We have designed an evolutionary algorithm to perform the search for the optimal sequence of states according to the selected model. The approximate solutions provided by evolutionary algorithms (EAs) are very appropriate for the problem since the probabilistic model used to penalize the state transitions and deal with noisy events, is just an approximation itself, which can be taken in different ways leading to different results. What is interesting for most applications, such as the detection of trends in streams of documents, is the detection of significant changes of state, i.e. the detection of clear changes in the average intensity of document arrivals, and not obtaining a very precise optimum numerical result for the cost function. Because of these reasons, this work investigates the application of evolutionary algorithms to the problem.

Besides using an alternative algorithm for fitting frequencies, in this paper we have also studied improvements of the EA to quickly obtain a fit for an extension of a previously fitted stream. The idea is to use the previous fitting curve as a seed for the EA which searches the fit of the extended stream.

The rest of the paper proceeds as follows: section 2 describes the model proposed by Kleinberg, which has inspired this work, and the variant we propose here; section 3 is devoted to describe an evolutionary algorithm used to find the optimal fit of frequency assignments; section 4 presents experiments to find the best evolutionary algorithm parameters, section 5 describes methods for dynamic detection of changes in the document stream trends, and section 6 draws the main conclusions from this work, and discusses future lines of research.

2. THE MODEL FOR THE PROBLEM

In order to model the arrival of documents in a stream we are going to use a finite state automaton (FSA) as in Kleinberg's approach [8], inspired in turn in models for network traffic in queuing theory [5] and on Hidden Markov Models [11]. According to this approach, a source of traffic emits documents at a rate which depends on the state of the FSA at a given point in time. A traffic burst begins with a transition from a state of lower rate of emission to one of higher rate. Kleinberg chooses an exponential density function $f(x) = \alpha e^{-\alpha x}, \alpha > 0$ to model the probability density of waiting times between arrivals. In the simplest case, we can distinguish between only two different states, with high and low frequency respectively. In state q_0 the automaton emits documents at a low rate, which gives rise to a probability density for the intervals $f_0(x) = \alpha_0 e^{-\alpha_0 x}$, while state q_1 has a higher emission rate, which gives rise to a probability density for the gaps given by $f_1(x) = \alpha_1 e^{-\alpha_1 x}$, with $\alpha_1 > \alpha_0$. Now the question is how to model the probability p with which the automaton changes its state between the emission of two consecutive documents. It is assumed that p is independent of previous emissions and state transitions. Assuming this probability distribution, we can compute the probability of a sequence $q = \{q_1, \cdots, q_n\}$ of state

transitions conditioned to the sequence $x = \{x_1, \cdots, x_n\}$ of gaps observed between the $n+1$ documents arrived in the stream. The state sequence which maximizes this probability $P(q|x)$ is the one which minimizes the cost function $c(q|x) = -\ln P(q|x)$. Applying this condition, Kleinberg obtains the formula

$$c(\mathbf{q}|\mathbf{x}) = b \ln \left(\frac{1-p}{p}\right) + \sum_{t=1}^{n} -\ln f_{q_t}(x_t) \quad (1)$$

where b is the number of state transitions done to emit the sequence, i.e. the number of times in which $q_t \neq q_{t+1}$, and $f_{q_t}(x) = \alpha_q e^{-\alpha_q x}$. In formula (1), we can observe that the fewer state transitions, the smaller the first term, while the better the state sequence fits the observed sequence of gaps x, the smaller the second term. Therefore, it is expected that the optimum sequence of states fits well the gap sequence with as few changes in the size of the gaps as possible, depending this inertia on the parameter b.

Afterwards, Kleinberg extends this simple two-state model to one of infinite states, providing a different one for each possible intensity of emission. Kleinberg proposes an automaton with an initial state q_0 whose corresponding density function $\alpha_0 e^{-\alpha_0 x}$ is assigned an emission rate $\alpha_0 = n/T$, where n is the number of gaps between documents emissions and T is the total length of the considered period of time; i.e., α_0 corresponds to a perfectly uniform event emission. For the remaining states $q > 0$, the assigned emission rate is $\alpha_q = \alpha_0 s^q$, where $s > 1$ is a scaling parameter, i.e. the smaller the gap, the greater the intensity. Then, by analogy with the two-state model, the cost function which the selected sequence of states must minimize, is

$$c(\mathbf{q}|\mathbf{x}) = \sum_{t=0}^{n-1} \tau(q_t, q_{t+1}) + \sum_{t=1}^{n} -\ln f_{q_t}(x_t) \quad (2)$$

where $\tau(q, q')$ represents the cost of a state transition from the state q at a given time t to the state q' at time $t+1$. Kleinberg, which considers that the selection of $\tau(q, q')$ is very flexible, chooses $\tau(q, q')$ in such a way that the cost of changing from a lower intensity state to a higher intensity one is proportional to the number of involved states, while there is no cost for changing for higher to lower intensity states. Specifically, the cost associated to change from state q to q', where $q > q'$, is defined as $(q' - q)\gamma \ln n$, γ being a parameter of the model. $\mathcal{A}^*_{s,\gamma}$ denotes the automaton of infinite states with parameters s and γ. The parameter s controls the scale for the rate values of the states, while γ, which Kleinberg sets to 1 in his experiments, controls the resistance to changing state.

Kleinberg shows that computing an optimal state sequence in an automata $\mathcal{A}^*_{s,\gamma}$ with infinite states is equivalent to compute q in one of its finite restrictions $\mathcal{A}^k_{s,\gamma}$, obtained by deleting from the automaton all states but the first k of them. This result allows establishing algorithms to compute the sequence of states for the minimum cost. For this purpose, Kleinberg adopts the standard dynamic programming algorithm used for hidden Markov Models.

However, this penalty function is not the only possible one. We have studied other penalization functions and concluded that the following measure:

$$\begin{cases} (q' - q)\gamma / \ln E & \text{if } q' > q \\ 0, & \text{if } q \leq q' \end{cases} \quad (3)$$

in which E is the total number of automaton states, avoids the dependency with the number of states and performs better than the one proposed by Kleinberg. Accordingly, is the one used in the experiments of this work.

3. THE EVOLUTIONARY ALGORITHM

Individuals represent sequences of state transitions in the automaton. The fitness of individuals is the cost function associated to the sequence of state transitions, and depends on the chosen probabilistic model.

3.1 Individual representation

Let E the number of states chosen for the automaton. Let us assume a stream of $n+1$ documents arriving along a period of time T. Then, the individuals of our evolutionary algorithm could be represented as the list of automaton states corresponding to each arrival of a document. Accordingly, an individual would be a list of n genes g_i, where $g_i \in \{0, \cdots, E\}$ is the state q in which the automaton is after the arrival of document i.

q_{t_1}	q_{t_2}	$\cdots$	q_{t_n}

However, the arrival of a new document does not produce a state transition in many cases. Therefore, the sequence of transitions can be represented in a more compact manner. Thus, an individual is a variable length list, in which each position, or "gene", represents the arrival of a subsequence of documents which do not lead to a change of state. Each gene is composed of an automaton state and of an identifier of the last document in the subsequence. Therefore, the individuals of our evolutionary algorithm could be represented as the list of state transitions in the automaton caused by the arrival of the documents.

g_1	g_2	$\cdots$	g_f
q_{t_1}, t_{k_1}	$q_{t_{k_1}+1}, t_{k_2}$	$\cdots$	q_{t_f}, t_n

3.2 The Fitness Function

For the fitness function we take, quite naturally, the cost function which defines the chosen statistical model. Thus the goal of our evolutionary algorithm is to find the sequence of state transitions $\mathbf{q} = (q_{i_1}, \cdots, q_{i_n})$ which minimizes the function

$$c(\mathbf{q}|\mathbf{x}) = \sum_{t=0}^{n-1} \tau(q_t, q_{t+1}) + \sum_{t=1}^{n} -\ln \alpha_{q_t} e^{-\alpha_{q_t} x_t},$$

with the penalty cost $\tau(q, q')$ and the state parameter, α_q, of the chosen statistical model. In particular, $\tau(q, q')$ is given by equation 3. In order to compute this function, the implicit automaton underlying the model must be completely defined, i.e. we have to assign values to each α_q. We assume that the number of automaton states has previously been fixed to E. Following Kleinberg's approach, we establish a *uniform state* $q = 0$, with a document arrival rate $\alpha_0 = n/T$, which corresponds to uniform document arrivals. For the remaining states, $q > 0$, the arrival rate is $\alpha_q = \alpha_0 s^q$, where $s > 1$ is a scaling parameter, i.e. the arrival intensity increases geometrically with q.

If we knew the value for the rate α_q of a particular state, we could obtain the value for s from:

$$\alpha_q = \frac{n}{T} s^q.$$

By assigning to α_E, the maximum arrival rate in the automaton, a particular value, such as 1, we obtain a value for s:

$$s = \exp\{(\ln \alpha_E - \ln n + \ln T)/E\}$$

3.3 Initial Population

Individuals of the initial population represent sequences of state transitions randomly generated. The simplest way of creating one such sequence is to choose a few documents at random and use them to split the whole document stream into intervals, each of which is assigned a random state. However, some preliminary experiments we have performed have shown that such a simple strategy gives rise to a search space that is too large for the algorithm to be efficient. Accordingly, we choose for a state transition only those documents for which the gaps with the previous document and with the following one are sufficiently different (the size of one at least 50 % longer than the other). The interval before the first transition is assigned a random state, and this state is increased or decreased in successive intervals according to whether the gaps on the left and right of the partition points increase or decrease in length, respectively. The size of this change is randomly chosen.

3.4 Crossover Operator

We have implemented the classic one point crossover, which creates two offspring by combining two individuals in such a way that the first part of one parent up to a crossover point is combined with the second part of the other parent and vice versa. Afterwards, the best offspring substitutes the worst parent. This is a steady state, elitist strategy.

The steps to apply this operator are the following:

- In order to select the crossover point, we randomly select one of the dates of document arrival in the stream. Then, we search in both parents the gene which contains this document.
- Then, the genes on the right-hand side of the selected gene in both parents are exchanged.
- For the gene containing the crossover point, we must decide if the substream of documents —which, in general, is different in both parents— is going to be joined or split. Experiments have shown that taking this decision at random produces bad results. Accordingly, if the document arrival gaps to the left and to the right of the crossover point are comparable (they differ less then 50%), the substream is assigned a single gene whose state is randomly selected from one of the parents. Otherwise, the substream is split at the crossover point in two genes, each taking the state from one parent.

3.5 Mutation Operator

The mutation operator is applied to every individual of the population with a probability given by the mutation rate. Different variants of mutation have been implemented, selecting at random the one to apply in each case:

- One of these mutation operators amounts to choosing a gene at random and randomly increment or decrement its state by one unit.

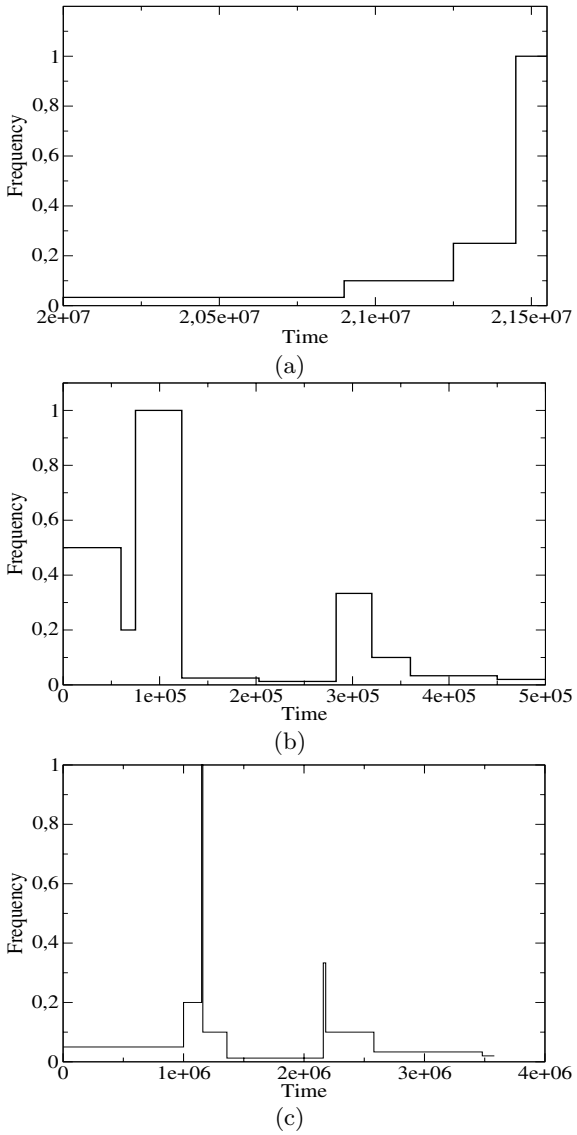

Figure 2: Artificial streams generated to evaluate the EA.

- Other mutation operator joins two consecutive genes to produce a single one. The state of the new gene is randomly taken from one of the original genes.
- The last mutation operator splits a gene in two; each one is assigned a different state: one of them is given the state of the original gene and the other one is given the previous state plus or minus one (plus if the gap on the left of the partition point is longer than the one on the right, and minus otherwise). This operator is only applied if the gaps in both sides of the partition are different.

4. EVOLUTIONARY ALGORITHM PARAMETERS

We have performed experiments to investigate the range of values for the EA parameters which provides best results. We have used the artificial streams (a), (b) and (c) depicted

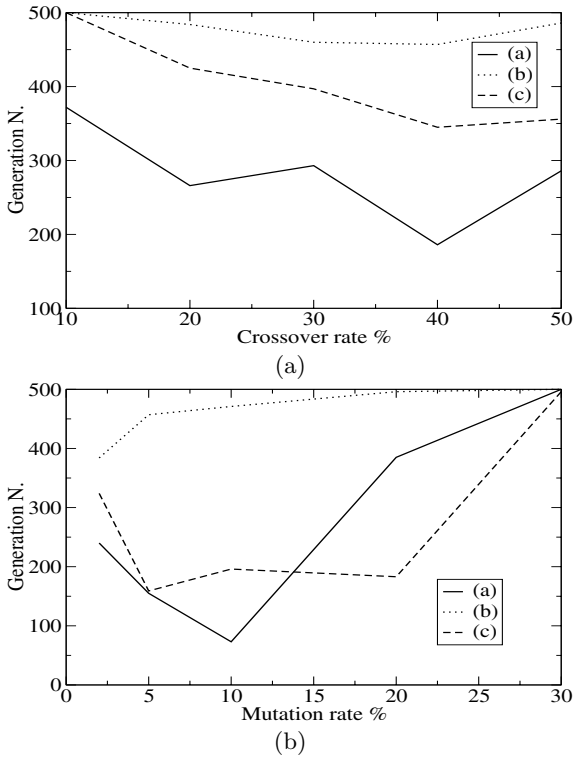

Figure 3: Number of generations required to reach convergence (as defined in the text) with different rates of crossover and mutation. Results correspond to the streams in Figure 2 (a, b and c).

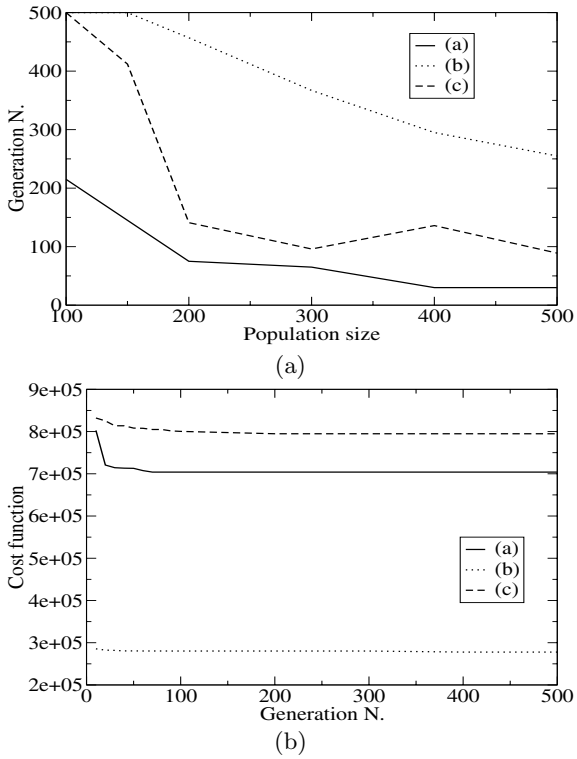

Figure 4: Number of generations required to reach convergence with different population sizes (a) and evolution of the cost function with the number of generation (b). Results correspond to the streams in Figure 2 (a, b and c).

in Figure 2 to study the best parameter settings for the EA. They are streams for which we know the emission frequencies that generated it. The first one (a) is characterized by ascending steps. Its length is 220000 dates of document arrivals. Stream (b) presents both, ascending and descending steps and has a length of 129000 dates. Stream (c), also presents both kinds of steps, but in this case, the length of the steps is very different. Figures 3(a) and 3(b) show the number of iterations required for the EA to reach convergence for the streams (a), (b) and (c) of Figure 2, with different rates of application of crossover and mutation, respectively. Convergence is achieved if the difference between the average fitness value of successive generations lies below a threshold for a number of generations. We can observe that intermediate values of these parameters, such as a crossover rate of 40 % and a mutation rate of %10, are enough to quickly reach convergence, which makes the EA perform efficiently.

Figure 4(a) presents the number of iterations required to reach the optimum value for the stream depicted in Figure 2 using different population sizes. We can observe that a population size of 200 individuals is enough. Larger sizes, although are also valid, increase the execution time. Figure 4(b) shows the value of the cost function as the number of iterations increases. This chart shows that convergence can be reached very quickly.

Tables 1 and 2 show the values obtained for the cost function and the execution time when using a dynamic programming algorithm and the EA (best result of five runs, average and standard deviation). It can be observed that the EA always yields the shortest running time. In some cases the value obtained for the cost function with both algorithms and the number of states in the automaton is the same. There are other cases in which the value provided by Viterbi is slightly better. However, in all these cases the fitting curve obtained with both algorithms is the same and the only difference is the absolute state number assigned to different steps; the relative change of state, and therefore of frequency, is maintained.

5. DYNAMIC DETECTION OF CHANGES OF INTEREST IN DOCUMENT STREAMS

In this section, we show how to apply the EA to dynamically model the data streams as new data arrive. Let us

State n.	Viterbi	
	Ex. time	Cost
15	2319.36	277402
20	3117.28	277306
25	3835.37	277260

Table 1: Execution time in seconds and value of the cost function for the stream of Figure 2(b) when applying the Viterbi algorithm (the classic dynamic programming one).

State n.	Evo. Alg	
	Ex. time	Cost (Av. Cost, Std. dev.)
15	1678.61	277712 (279385.6, 980.11)
20	2182.12	277528 (278980.4, 1114.91)
25	2033.81	277270 (279472.6, 1116.03)

Table 2: Execution time in seconds and value of the cost function for the stream of Figure 2(b) when applying the Evolutionary algorithms to find the optimal sequence of states in automata with different number of states. The EA has been run with a population size of 200 individuals, a maximum number of 200 iterations, a crossover rate of 40% and a mutation rate of 10%. The values presented for are the execution time and the cost (best of the five runnings), with the average and the standard deviation appearing in brackets.

assume the fit for a stream has been found, new documents arrive, and we want to detect possible changes in the trends of the corresponding topic (or the document stream, in general). Clearly, the most accurate way of doing this is to apply again the algorithm for finding the best fit for the extended stream, but this solution is costly in terms of time, and obviously not suitable for real-time applications on the web, which is its major field of application. That is why we have investigated ways to predict the trend of the next arrival, without fitting the whole stream until there is time to do it.

5.1 Approximation to predict the state of a new document

Previous substream	A. T.	Old s.	New s.	Trend
$\cdots$ 38 38 39 41 49 49	52	12	0	$\downarrow$
$\cdots$ 41 49 49 52 68 69	69	3	4	$\uparrow$
$\cdots$ 88 89 90 90 91 92	95	0	0	$\rightarrow$

Table 3: Results of applying a local approximation to detect changes in the trend of a stream. *A.T.* stands for arrival time, *Old s.* for old state and *New s.* for new state. The first column indicates the arrival times of the previous documents.

If we need to approximate a new arrival very quickly, we can perform an approximation based on the idea that the current trend of the stream is maintained. This can be done by searching the minimum cost for the new gap x according to the last state of the previous fit. Thus, if the fit obtained for the previous stream finished at state q_i, we look for the state q_j which minimizes the cost

$$arg\ min_{q'} P(q'|q,x) = arg\ min_{q'}(\tau(q,q') + \ln\ \alpha_{q'} e^{-\alpha_{q'} x}),$$

equivalent to a single step of equation 2.

We have chosen one real world stream in order to check if this "local" approximation is meaningful, the one tracking the term *gmail*. It has been obtained from the Blogalia, weblog hosting site[1], by doing a database search on the word **gmail**. We have used several partitions of this document stream to perform the experiments. At each partition point we assume that we have received the documents preceding

[1] http://www.blogalia.com

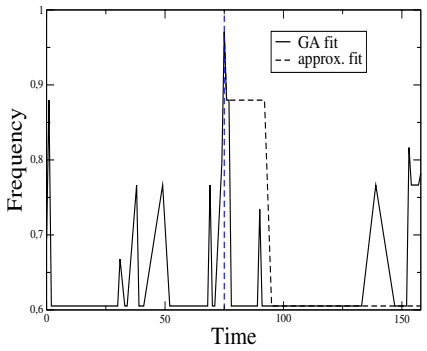

Figure 5: Fitting curves obtained with the EA and the local approximation for the stream of documents called *gmail*. The results of the local approximation start at the date whose integer representation is 76.

the partition date and we have found the fitting curve for them. Then, the remaining documents arrive and we want to quickly fit them.

Table 3 shows the results on some of these partitions. This mechanism, which allows us to immediately detect changes in the trends of a stream, has properly worked in all these cases, yielding the state that had previously been found by the fitting algorithms. The problem arises when this approximation is applied to subsequent dates, a is shown in Table 4, which shows some states corresponding to the fitting curve obtained by the EA, as well as those resulting the approximation proposed herein starting at the date 2004-06-15, which corresponds to the integer representation of 75. We can see that only the first point has been correctly approximated. Moreover, Figure 5, which presents the whole fit for the stream *gmail*, shows that the divergence of the results of both methods increases with time. These results reveal the need of finding a solution that fits, in as little

date	GA	approx.
0(2004-04-02)	7(0.694669)	
$\cdots$	$\cdots$	
69(2004-06-10)	12(0.766507)	
70(2004-06-11)	0(0.605263)	
71(2004-06-12)	0(0.556962)	
74(2004-06-15)	14(0.797281)	
75(2004-06-16)	24(0.970706)	
76(2004-06-17)	19(0.87973)	
77(2004-06-18)	19(0.87973)	19(0.87973)
78(2004-06-19)	0(0.605263)	19(0.87973)
79(2004-06-20)	0(0.605263)	19(0.87973)
80(2004-06-21)	0(0.605263)	19(0.87973)
81(2004-06-22)	0(0.605263)	19(0.87973)
82(2004-06-23)	0(0.605263)	19(0.87973)

Table 4: Results of applying a local approximation repeatedly, starting at the date whose integer representation is 76. The first column corresponds to the integer representation of the date, the second one to the states assigned by the EA that fits the whole stream (the corresponding frequency in brackets), and the third one to the state assigned applying the approximation.

time as possible, new arrivals. In the next section we tackle the problem of fitting the extension of a stream with an EA which takes advantage of the previous fit.

5.2 A Fast Genetic Algorithm to model new states

Subst. len.	New Subs. len.	T. w/out seed	T. w/ seed
219900	100		141.45 (79.09)
219000	1000	3895.28	144.75 (81.96)
210000	10000		166.73 (79.32)

Table 5: Time (seconds) spent without seed and using the previous fit as a seed for some substream taken from the (artificially generated) stream of Figure 2(c). Results correspond to the best of five runs. The first column indicates the length of the previously fitted substream, and the second one the length of the stream of documents which has to be added to the previous one. The third column presents the time needed to fit the whole stream, while the last one presents the time spent to reach convergence by using the previous fit as seed, with a population size of 200 individuals, a crossover rate of 40 % and a mutation rate of 10%. The result for a population size of 100 individuals appears in parentheses.

In order to quickly obtain the fit for an extension of a previously fitted stream, we can use the previous fitting curve as a seed for the EA which searches the fit of the extended stream. To implement this mechanism, we have modified the way in which the EA creates the initial population. A *seed individual* is created, and *extended* with a set of genes corresponding to the new substream of document dates. The initial population is created by applying the mutation operator to this seed individual, but in such a way that the last gene has a higher probability to undergo mutation. Once the initial population has been created, the EA proceeds as explained in section 3. Experiments have shown that this mechanism can save a lot of execution time. Table 5 shows the time required to fit the stream of Figure 2(c) if a part of it, whose length appears in the first column, has already been fitted. We can observe that the time required is very small.

In order to further evaluate this approach we have tested it to fit a sequence from the real world formed by the comments sent to all blogs hosted in Blogalia (http://blogalia.com) during the period January 2002-January 2006. Figure 6 shows the fitting curves obtained by applying the EA to the whole sequence (Figure 6(a)) and using as seed the fitting curve of the subsequence which lacks the last 1000 dates (Figure 6(b)). We can see that the curves are very similar, showing the same intervals of higher interest in the topic. Furthermore, we can observe that the area corresponding to the last 1000 dates is cleaner in the fit obtained from the seed. This is probably because in this case, due to the way in which the individuals are created, the search is centered in the area of the new dates, providing more precise results for it.

Finally, since we are interested in obtaining solutions in as little a time as possible, several experiments have been made varying the population size with the sight on striking a balance between model accuracy (high fitness) and running time. Figure 7 shows the result of this set of experiments; it plots the evolution of the fitness with the generations for different population sizes. We can see that even for small populations of 500 individuals, the evolution is fast; no further achievement is obtained for populations bigger than that; thus, we can conclude that this population size is more than enough for models of the size we are dealing with in this paper.

Subst. Len.	New Subs. len.	T. w/out seed	T. w/ seed
3032	100	5048.49	54.6
2632	500		92.247
2132	1000		294.97
1132	2000		570.41

Table 6: Time (seconds) spent fitting the whole stream using an evolutionary algorithm (*w/o seed* column) and using the previous fit as a seed for some substream taken from the time sequence of comments including the word 'blog' during the January 2002-January 2006 period. Results correspond to the best of five runs. The first column indicates the length of the previously fitted substream, and the second one the length of the stream of documents which has to be added to the previous one. The third column presents the time needed to fit the whole stream (with a population size of 1000 individual, 10000 iterations, crossover rate of 40%, and mutation rate of 10%), while the last one presents the time spent to reach convergence by using the previous fit as seed, with a population size of 1000 individuals, a number of iterations equal to the size of the new substream, a crossover rate of 40 % and a mutation rate of 10%.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented the design of a system devoted to the dynamic detection of changes on the trends of the topics of a stream of documents, such as newscasts, e-mails, IRC conversations, scientific journals or weblogs. It is based on modeling the assignment of frequencies to intervals of document arrivals and obtaining an optimal fit to the data. We have designed an evolutionary algorithm to implement the model, which allows us to deal with very large sequences of documents in a reasonable time, obtaining fitting curves with a similar shape to those provided by classic dynamic algorithms.

We have also designed a version of the evolutionary algorithm which dramatically reduces the time required to find the optimal fit to a stream which is an extension of a previously fitted substream. This version of the evolutionary algorithm uses the previous fit as a seed to generate the initial population, which can quickly converge if most of the stream has been previously fitted. In this way, our system can be applied to dynamically model the document stream, and thus detect changes on the trends of the corresponding topic in real time. Besides, the fitting curves produced by the system for a stream of documents can also be useful for other applications: the fit obtained for streams corresponding to different topics can help to detect correlations between these topics, to study how a topic affects others, etc.

For the future we plan to study correlations among document streams, to automatically detect the occurrence of

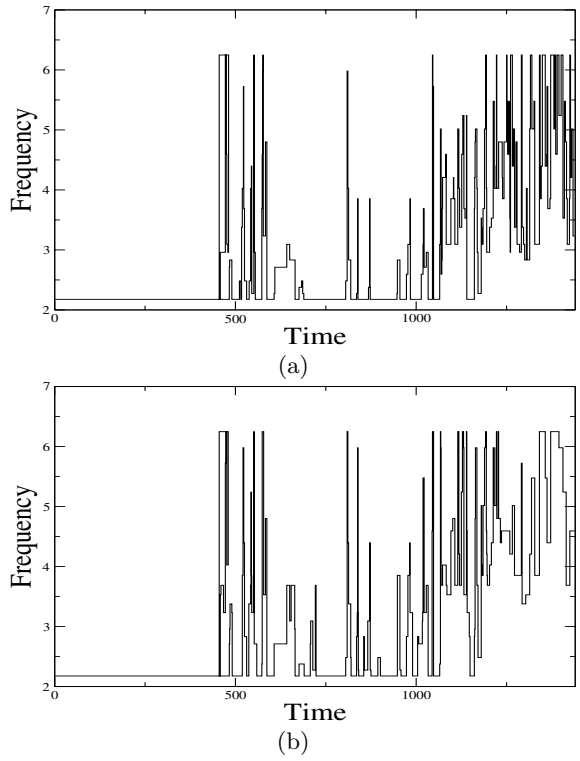

Figure 6: Fits obtained for the sequence of blog comments including the word *blog*. In Figure (a) (top) the EA has been used to fit the whole sequence, with a population size of 1000 individuals, 10000 generations, a crossover rate of 40% and a mutation rate of 10%. Figure (b) shows the results when the EA is applied to a part of the sequence missing the last 1000 dates (corresponding to the third row of table 6), and the result of this fit is used as seed for another EA which produces the fit of the whole sequence. In this case the algorithm has been run with a population size of 1000 individuals, 1000 generations, a crossover rate of 40% and a mutation rate of 10%.

new topics composed of multi-word concepts. This can also be helped by other techniques, as well as optimization for real-time operation, including parallelization of the algorithms. We will also perform experiments with more document streams, in order to find out which parameters are the most adequate for each situation, and whether the algorithm scales and how for larger document streams.

Acknowledgments

This work has been supported by the Ministry of Science and Technology TIC2003-09481-C04. The first author has also been partially supported by Ingeniería del Software e Inteligencia Artificial group, ref. 910494 and by the Regional Government of Madrid under the Research Network MAVIR (S-0505/TIC-0267).

7. REFERENCES

[1] J. Allan. *Topic Detection and Tracking: Event-Based Information Organization*. Kluwer Academic Publishers, 2002.

[2] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study, 1998.

[3] E. Bingham, A. Kabán, and M. Girolami. Topic identification in dynamical text by complexity pursuit. *Neural Process. Lett.*, 17(1):69–83, 2003.

[4] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In Proceedings of the 29th International Colloquium on Automata, Languages, and Programming, 2002.*, 2002.

[5] A. I. Elwalid and D. Mitra. Effective bandwidth of general markovian traffic sources and admission control of high speed networks. *IEEE/ACM Trans. Netw.*, 1(3):329–343, 1993.

[6] G. D. Forney. The Viterbi algorithm. *Proceedings of The IEEE*, 61(3):268–278, 1973.

[7] M. Girolami and A. Kaban. Simplicial mixtures of Markov chains: Distributed modelling of dynamic user profiles. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[8] J. Kleinberg. Bursty and hierarchical structure in streams. In *Proc. 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pages 91–101. ACM, 2002.

[9] J. Kleinberg. Temporal dynamics of on-line information streams. In M. Garofalakis, J. Gehrke, and R. Rastogi, editors, *Data Stream Management: Processing High-Speed Data Streams (to appear)*. Springer, 2005.

[10] R. Papka. *On-line New Event Detection, Clustering and Tracking*. PhD thesis, Department of Computer Science, University of Massachusetts, 1999.

[11] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Readings in speech recognition*, pages 267–296. Morgan Kaufmann Publishers Inc., 1990.

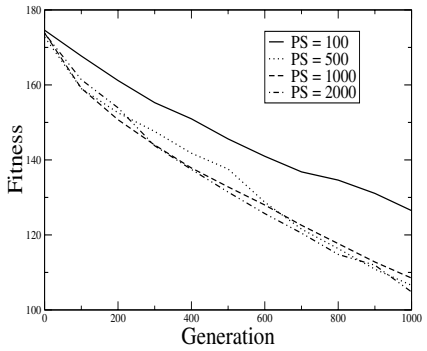

Figure 7: Fitness evolution with different population sizes when fitting a new substream of 1000 dates, using a crossover rate of 40% and a mutation rate of 10%. As it can be observed, no substantial improvement is observed for population bigger than 500 individuals.