A Genetic Algorithm for Text Classification Rule Induction

Adriana Pietramala[1], Veronica L. Policicchio[1], Pasquale Rullo[1,2], and Inderbir Sidhu[3]

[1] University of Calabria Rende - Italy
{a.pietramala,policicchio,rullo}@mat.unical.it
[2] Exeura s.r.l. Rende, Italy
[3] Kenetica Ltd Chicago, IL-USA
isidhu@computer.org

Abstract. This paper presents a Genetic Algorithm, called Olex-GA, for the induction of rule-based text classifiers of the form "classify document d under category c if $t_1 \in d$ or ... or $t_n \in d$ and not ($t_{n+1} \in d$ or ... or $t_{n+m} \in d$) holds", where each t_i is a term. Olex-GA relies on an efficient *several-rules-per-individual* binary representation and uses the F-measure as the fitness function. The proposed approach is tested over the standard test sets Reuters-21578 and Ohsumed and compared against several classification algorithms (namely, Naive Bayes, Ripper, C4.5, SVM). Experimental results demonstrate that it achieves very good performance on both data collections, showing to be competitive with (and indeed outperforming in some cases) the evaluated classifiers.

1 Introduction

Text Classification is the task of assigning natural language texts to one or more thematic categories on the basis of their contents.

A number of machine learning methods have been proposed in the last few years, including k-nearest neighbors (k-NN), probabilistic Bayesian, neural networks and SVMs. Overviews of these techniques can be found in [13, 20].

In a different line, rule learning algorithms, such as [2, 3, 18], have become a successful strategy for classifier induction. Rule-based classifiers provide the desirable property of being readable and, thus, easy for people to understand (and, possibly, modify).

Genetic Algorithms (GA's) are stochastic search methods inspired to the biological evolution [7, 14]. Their capability to provide good solutions for classical optimization tasks has been demonstrated by various applications, including TSP [9, 17] and Knapsack [10]. Rule induction is also one of the application fields of GA's [1, 5, 15, 16]. The basic idea is that each *individual* encodes a candidate solution (i.e., a classification rule or a classifier), and that its fitness is evaluated in terms of predictive accuracy.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part II, LNAI 5212, pp. 188–203, 2008.

In this paper we address the problem of inducing propositional text classifiers of the form

$$c \leftarrow (t_1 \in d \vee \cdots \vee t_n \in d) \wedge \neg(t_{n+1} \in d \vee \cdots \vee t_{n+m} \in d)$$

where c is a category, d a document and each t_i a term (n-gram) taken from a given vocabulary. We denote a classifier for c as above by $\mathcal{H}_c(Pos, Neg)$, where $Pos = \{t_1, \cdots, t_n\}$ and $Neg = \{t_{n+1} \cdots t_{n+m}\}$. Positive terms in Pos are used to cover the training set of c, while negative terms in Neg are used to take precision under control.

The problem of learning $\mathcal{H}_c(Pos, Neg)$ is formulated as an optimization task (hereafter referred to as MAX-F) aimed at finding the sets Pos and Neg which maximize the F-measure when $\mathcal{H}_c(Pos, Neg)$ is applied to the training set. MAX-F can be represented as a 0-1 combinatorial problem and, thus, the GA approach turns out to be a natural candidate resolution method. We call Olex-GA the genetic algorithm designed for the task of solving problem MAX-F.

Thanks to the simplicity of the hypothesis language, in Olex-GA an individual represents a candidate classifier (instead of a single rule). The fitness of an individual is expressed in terms of the F-measure attained by the corresponding classifier when applied to the training set. This *several-rules-per-individual* approach (as opposed to the *single-rule-per-individual* approach) provides the advantage that the fitness of an individual reliably indicates its quality, as it is a measure of the predictive accuracy of the encoded classifier rather than of a single rule.

Once the population of individuals has been suitably initialized, evolution takes place by iterating elitism, selection, crossover and mutation, until a predefined number of generations is created.

Unlike other rule-based systems (such as Ripper) or decision tree systems (like C4.5) the proposed method is a one-step learning algorithm which does not need any post-induction optimization to refine the induced rule set. This is clearly a notable advantage, as rule set-refinement algorithms are rather complex and time-consuming tasks.

The experimentation over the standard test sets Reuters-21578 and Ohsumed confirms the goodness of the proposed approach: on both data collections, Olex-GA showed to be highly competitive with some of the top-performing learning algorithms for text categorization, notably, Naive Bayes, C4.5, Ripper and SVM (both polynomial and rbf). Furthermore, it consistently defeated the greedy approach to problem MAX-F we reported in [19]. In addition, Olex-GA turned out to be an efficient rule induction method (e.g., faster than both C4.5 and Ripper).

In the rest of the paper, after providing a formulation of the learning optimization problem (Section 2) and proving its NP-hardness, we describe the GA approach proposed to solve it (Section 3). Then, we present the experimental results (Section 5) and provide a performance comparison with some of the top-performing learning approaches (Section 6). Finally, we briefly discuss the relation to other rule induction systems (Sections 7 and 8) and give the conclusions.

2 Basic Definitions and Problem Statement

In the following we assume the existence of:

1. a finite set $\mathcal{C}$ of categories, called *classification scheme*;
2. a finite set $\mathcal{D}$ of documents, called *corpus*; $\mathcal{D}$ is partitioned into a *training set* TS, a *validation set* and a *test set*; the training set, along with the validation set, represents the so-called *seen data*, used to induce the model, while the test set represents the *unseen data*, used to asses performance of the induced model;
3. a binary relationship which assigns each document $d \in D$ to a number of categories in $\mathcal{C}$ (*ideal classification*). We denote by TS_c the subset of TS whose documents belong to category c (the *training set of* c);
4. a set $\Phi = \{f_1, \cdots, f_k\}$ of scoring functions (or feature selection functions), such as Information Gain, Chi Square, etc. [4, 22], used for vocabulary reduction (we will hereafter often refer to them simply as "functions").

A *vocabulary* $V(k, f)$ over the training set TS is a set of terms (n-grams) defined as follows: let $V_c(k, f)$ be the set of the k terms occurring in the documents of TS_c that score highest according to the scoring function $f \in \Phi$; then, $V(k, f) = \cup_{c \in \mathcal{C}} V_c(k, f)$, that is, $V(k, f)$ is the union of the k best terms, according to f, of each category of the classification scheme.

We consider a hypothesis language of the form

$$c \leftarrow (t_1 \in d \vee \cdots \vee t_n \in d) \wedge \neg(t_{n+1} \in d \vee \cdots \vee t_{n+m} \in d) \tag{1}$$

where each t_i is a term taken from a given vocabulary. In particular, each term in $Pos = \{t_1, \cdots, t_n\}$ is a *positive* term, while each term in $Neg = \{t_{n+1} \cdots t_{n+m}\}$ is a *negative* term. A classifier as above, denoted $\mathcal{H}_c(Pos, Neg)$, states the condition "if *any* of the terms $t_1, \cdots, t_n$ occurs in d and *none* of the terms $t_{n+1}, \cdots, t_{n+m}$ occurs in d then classify d under category c"[1]. That is, the occurrence of a positive term in a document d requires the contextual *absence* of the (possibly empty) set of negative terms in order for d be classified under c[2].

To assess the accuracy of a classifier for c, we use the classical notions of Precision Pr_c, Recall Re_c and F-measure $F_{c,\alpha}$, defined as follows:

$$Pr_c = \frac{a}{a+b}; \quad Re_c = \frac{a}{a+e}; \quad F_{c,\alpha} = \frac{Pr_c \cdot Re_c}{(1-\alpha)Pr_c + \alpha Re_c} \tag{2}$$

Here a is the number of *true positive* documents w.r.t. c (i.e., the number of documents of the test set that have correctly been classified under c), b the number of *false positive* documents w.r.t. c and e the number of *false negative* documents w.r.t. c. Further, in the definition of $F_{c,\alpha}$, the parameter $\alpha \in [0 \cdots 1]$

[1] It is immediate to recognize that $c \leftarrow (t_1 \in d \vee \cdots \vee t_n \in d) \wedge \neg(t_{n+1} \in d \vee \cdots \vee t_{n+m} \in d)$ is equivalent to the following set of classification rules: $\{c \leftarrow t_1 \in d \wedge \neg(t_{n+1} \in d) \wedge \cdots \wedge \neg(t_{n+m} \in d), \cdots, c \leftarrow t_n \in d \wedge \neg(t_{n+1} \in d) \wedge \cdots \wedge \neg(t_{n+m} \in d)\}$.

[2] d may "satisfy" more then one; thus, it may be assigned to multiple categories.

is the relative degree of importance given to Precision and Recall; notably, if $\alpha = 1$ then $F_{c,\alpha}$ coincides with Pr_c, and if $\alpha = 0$ then $F_{c,\alpha}$ coincides with Re_c (a value of $\alpha = 0.5$ attributes the same importance to Pr_c and Re_c)[3].

Now we are in a position to state the following optimization problem:

PROBLEM MAX-F. Let a category $c \in \mathcal{C}$ and a vocabulary $V(k, f)$ over the training set TS be given. Then, find two subsets of $V(k, f)$, namely, $Pos = \{t_1, \cdots, t_n\}$ and $Neg = \{t_{n+1}, \cdots, t_{n+m}\}$, with $Pos \neq \emptyset$, such that $\mathcal{H}_c(Pos, Neg)$ applied to TS yields a maximum value of $F_{c,\alpha}$ (over TS), for a given α.

Proposition 1. Problem *MAX-F* is NP-hard.

Proof. We next show that the decision version of problem *MAX-F* is NP-complete (i.e., it requires time exponential in the size of the vocabulary, unless $P = NP$). To this end, we restrict our attention to the subset of the hypothesis language where each rule consists of only positive terms, i.e, $c \leftarrow t_1 \in d \vee \cdots \vee t_n \in d$ (that is, classifiers are of the form $\mathcal{H}_c(Pos, \emptyset)$). Let us call *MAX-F+* this special case of problem *MAX-F*.

Given a generic set $Pos \subseteq V(k, f)$, we denote by $S \subseteq \{1, \cdots, p\}$ the set of indices of the elements of $V(k, f) = \{t_1, \cdots, t_q\}$ that are in Pos, and by $\Delta(t_i)$ the set of documents of the training set TS where t_i occurs. Thus, the set of documents classifiable under c by $\mathcal{H}_c(Pos, \emptyset)$ is $D(Pos) = \cup_{i \in S} \Delta(t_i)$. The F-measure of this classification is

$$F_{c,\alpha}(Pos) = \frac{|D(Pos) \cap TS_c|}{(1-\alpha)|TS_c| + \alpha|D(Pos)|}. \tag{3}$$

Now, problem *MAX-F+*, in its decision version, can be formulated as follows: "$\exists$ $Pos \subseteq V(k, f)$ such that $F_{c,\alpha}(Pos) \geq K$?", where K is a constant (let us call it *MAX-F+(D)*).

Membership. Since $Pos \subseteq V(k, f)$, we can clearly verify a YES answer of *MAX-F+(D)*, using equation 3, in time polynomial in the size $|V(k, f)|$ of the vocabulary.

Hardness. We define a partition of $D(Pos)$ into the following subsets: (1) $\Psi(Pos) = D(Pos) \cap TS_c$, i.e., the set of documents classifiable under c by $\mathcal{H}_c(Pos, \emptyset)$ that belong to the training set TS_c (true classifiable); (2) $\Omega(Pos) = D(Pos) \setminus TS_c$, i.e., the set of documents classifiable under c by $\mathcal{H}_c(Pos, \emptyset)$ that do not belong to TS_c (false classifiable).

Now, it is straightforward to see that the F-measure, given by equation 3, is proportional to the size $\psi(Pos)$ of $\Psi(Pos)$ and inversely proportional to the size $\omega(Pos)$ of $\Omega(Pos)$ (just replace in equation 3 the quantities $|D(Pos) \cap TS_c|$ by $\psi(Pos)$ and $|D(Pos)|$ by $\psi(Pos) + \omega(Pos)$). Therefore, problem *MAX-F+(D)* can equivalently be stated as follows: "$\exists$ $Pos \subseteq V(k, f)$ such that $\psi(Pos) \geq V$ and $\omega(Pos) \leq C$?", where V and C are constants.

[3] Since $\alpha \in [0 \cdots 1]$ is a parameter of our model, we find convenient using $F_{c,\alpha}$ rather than the equivalent $F_\beta = \frac{(\beta^2+1) Pr_c \cdot Re_c}{\beta^2 Pr_c + Re_c}$, where $\beta \in [0 \cdots \infty]$ has no upper bound.

Now, let us restrict *MAX-F+(D)* to the simpler case in which terms in Pos are pairwise disjoint, i.e., $\Delta(t_i) \cap \Delta(t_j) = \emptyset$ for all $t_i, t_j \in V(k, f)$ (let us call $\mathcal{DSJ}$ this assumption). Next we show that, under $\mathcal{DSJ}$, *MAX-F+(D)* coincides with the Knapsack problem. To this end, we associate with each $t_i \in V(k, f)$ two constants, v_i (the *value* of t_i) and a c_i (the *cost* of t_i) as follows:

$$v_i = |\Delta(t_i) \cap TS_c|, \quad c_i = |\Delta(t_i) \setminus TS_c|.$$

That is, the value v_i (resp. cost c_i) of a term t_i is the number of documents containing t_i that are true classifiable (resp., false classifiable).

Now we prove that, under $\mathcal{DSJ}$, the equality $\Sigma_{i \in S} v_i = \psi(Pos)$ holds, for any $Pos \subseteq V(k, f)$. Indeed:

$$\Sigma_{i \in S} v_i = \Sigma_{i \in S} |\Delta(t_i) \cap TS_c| \; = \; |(\cup_{i \in S} \Delta(t_i)) \cap TS_c| = |(D(Pos) \cap TS_c| = \psi(Pos).$$

To get the first equality above we apply the definition of v_i; for the second, we exploit assumption $\mathcal{DSJ}$; for the third and the fourth we apply the definitions of $D(Pos)$ and $\psi(Pos)$, respectively.

In the same way as for v_i, the equality $\Sigma_{i \in S} c_i = \omega(Pos)$ can be easily drawn.

Therefore, by replacing $\psi(Pos)$ and $\omega(Pos)$ in our decision problem, we get the following new formulation, valid under $\mathcal{DSJ}$: "$\exists$ $Pos \subseteq V(k, f)$ such that $\Sigma_{i \in S} v_i \geq V$ and $\Sigma_{i \in S} c_i \leq C$?". That is, under $\mathcal{DSJ}$, *MAX-F+(D)* is the Knapsack problem – a well known NP-complete problem. Therefore, *MAX-F+(D)* under $\mathcal{DSJ}$ is NP-complete. It turns out that (the general case of) *MAX-F+(D)* (which is at least as complex as *MAX-F+(D)* under $\mathcal{DSJ}$) is NP-hard and, thus, the decision version of problem *MAX-F+* is NP-hard as well. It turns out that the decision version of problem *MAX-F* is NP-hard.

Having proved both membership (in NP) and hardness, we conclude that the decision version of problem *MAX-F* is NP-complete.

A solution of problem MAX-F is a best classifier for c *over the training set* TS, for a given vocabulary $V(f, k)$. We assume that categories in $\mathcal{C}$ are mutually independent, i.e., the classification results of a given category are not affected by the results of any other. It turns out that we can solve problem MAX-F for that category independently on the others. For this reason, in the following, we will concentrate on a single category $c \in \mathcal{C}$.

3 Olex-GA: A Genetic Algorithm to Solve MAX-F

Problem MAX-F is a combinatorial optimization problem aimed at finding a best combination of terms taken from a given vocabulary. That is, MAX-F is a typical problem for which GA's are known to be a good candidate resolution method.

A GA can be regarded as composed of three basic elements: (1) A *population*, i.e., a set of candidate solutions, called *individuals* or *chromosomes*, that will evolve during a number of iterations (*generations*); (2) a *fitness function* used to

assign a score to each individual of the population; (3) an evolution mechanism based on operators such as *selection*, *crossover* and *mutation*. A comprehensive description of GA can be found in [7].

Next we describe our choices concerning the above points.

3.1 Population Encoding

In the various GA-based approaches to rule induction used in the literature (e.g., [5, 15, 16]), an individual of the population may either represent a single rule or a rule set. The former approach (single-rule-per-individual) makes the individual encoding simpler, but the fitness of an individual may not be a meaningful indicator of the quality of the rule. On the other hand, the several-rules-per-individual approach, where an individual may represent an entire classifier, requires a more sophisticated encoding of individuals, but the fitness provides a reliable indicator. So, in general, there is a trade-off between simplicity of encoding and effectiveness of the fitness function.

Thanks to the structural simplicity of the hypothesis language, in our model an individual encodes a classifier in a very natural way, thus combining the advantages of the two aforementioned approaches. In fact, an individual is simply a binary representation of the sets Pos and Neg of a classifier $\mathcal{H}_c(Pos, Neg)$.

Our initial approach was that of representing $\mathcal{H}_c(Pos, Neg)$ through an individual consisting of $2 \cdot |V(k, f)|$ bits, half for the terms in Pos, and half for the terms in Neg (recall that both Pos and Neg are subsets of $V(k, f)$). This however proved to be very ineffective (and inefficient), as local solutions representing classifiers with hundreds of terms were generated.

More effective classifiers were obtained by restricting the search of both positive and negative terms, based on the following simple observations: (1) scoring functions are used to assess goodness of terms w.r.t. a given category; hence, it is quite reasonable searching positive terms for c within $V_c(k, f)$, i.e., among the terms that score highest for c according to a given function f; (2) the role played by negative terms in the training phase is that of avoiding classification of documents (of the training set) containing some positive term, but falling outside the training set TS_c of c.

Based on the above remarks, we define the following subsets $Pos*$ and $Neg*$ of $V(k, f)$:

- $Pos* = V_c(k, f)$, i.e., $Pos*$ is the subset of $V(k, f)$ consisting of the k terms occurring in the documents of the training set TS_c of c that score highest according to scoring function f; we say that $t_i \in Pos*$ is a *candidate positive term* of c over $V(k, f)$;
- $Neg* = \{t \in V(k, f) \mid (\cup_{t_j \in Pos*} \Delta(t_j) \setminus TS_c) \cap \Delta(t) \neq \emptyset\}$, where $\Delta(t_j) \subseteq TS$ (resp. $\Delta(t) \subseteq TS$) is the set of documents of TS containing term t_j (resp. t); we say that $t_i \in Neg*$ is a *candidate negative term* of c over $V(k, f)$. Intuitively, a candidate negative term is one which occurs in a document containing some candidate positive term and not belonging to the training set TS_c of c.

Now, we represent a classifier $\mathcal{H}_c(Pos, Neg)$ over $V(k, f)$ as a chromosome K made of $|Pos*| + |Neg*|$ bits. The first $|Pos*|$ bits of K, denoted K^+, are used to represent the terms in Pos, and the remaining $|Neg*|$, denoted K^-, to represent the terms in Neg. We denote the bit in K^+ (resp. K^-) representing the candidate positive (resp. negative) term t_i as $K^+[t_i]$ (resp. $K^-[t_i]$). Thus, the value 0-1 of $K^+[t_i]$ (resp. $K^-[t_i]$) denotes whether $t_i \in Pos*$ (resp. $t_i \in Neg*$) is included in Pos (resp. Neg) or not.

A chromosome K is *legal* if there is no term $t_i \in Pos* \cap Neg*$ such that $K^+[t_i] = K^-[t_i] = 1$ (that is, t_i cannot be at the same time a positive and a negative term)[4].

3.2 Fitness Function

The fitness of a chromosome K, representing $\mathcal{H}_c(Pos, Neg)$, is the value of the F-measure resulting from applying $\mathcal{H}_c(Pos, Neg)$ to the training set TS. This choice naturally follows from the formulation of problem MAX-F. Now, if we denote by $D(K) \subseteq TS$ the set of all documents containing any positive term in Pos and no negative term in Neg, i.e.,

$$D(K) = \cup_{t \in Pos} \Delta(t) \setminus \cup_{t \in Neg} \Delta(t),$$

starting from the definition of $F_{c,\alpha}$, after some algebra, we obtain the following formula for $F_{c,\alpha}$:

$$F_{c,\alpha}(K) = \frac{|D(K) \cap TS_c|}{(1-\alpha)|TS_c| + \alpha|D(K)|}.$$

3.3 Evolutionary Operators

We perform selection via the roulette-wheel method, and crossover by the uniform crossover scheme. Mutation consists in the flipping of each single bit with a given (low) probability. In order not to lose good chromosomes, we apply elitism, thus ensuring that the best individuals of the current generation are passed to the next one without being altered by a genetic operator. All the above operators are described in [7].

3.4 The Genetic Algorithm

A description of Olex-GA is sketched in Figure 1. First, the sets $Pos*$ and $Neg*$ of candidate positive and negative terms, respectively, are computed from the input vocabulary $V(k, f)$. After population has been initialized, evolution takes place by iterating elitism, selection, crossover and mutation, until a pre-defined number n of generations is created. At each step, a repair operator ρ, aimed at correcting possible illegal individuals generated by crossover/mutation, is applied. In particular, if K is an illegal individual with $K^+[t] = K^-[t] = 1$,

[4] Note that the classifier encoded by an illegal individual is simply redundant, as it contains a "dummy" rule of the form $c \leftarrow t \in d, \cdots \neg(t \in d), \cdots$.

Algorithm Olex-GA

Input: vocabulary $V(f,k)$ over the training set TS; number n of generations;
Output: "best" classifier $\mathcal{H}_c(Pos, Neg)$ of c over TS;

- **begin**
- Evaluate the sets of candidate positive and negative terms from $V(f,k)$;
- Create the population *oldPop* and initialize each chromosome;
- **Repeat** n times
- Evaluate the fitness of each chromosome in *oldPop*;
- $newPop = \emptyset$;
- Copy in $NewPop$ the best r chromosomes of *oldPop* (elitism - r is determined on the basis of the elitism percentage)
- **While** $size(newPop) < size(oldPop)$
- select *parent1* and *parent2* in *oldPop* via roulette wheel
- generate *kid1*, *kid2* through $crossover(parent1, parent2)$
- apply mutation, i.e., $kid1 = mut(kid1)$ and $kid2 = mut(kid2)$
- apply the repair operator ρ to both $kid1$ and $kid2$;
- add *kid1* and *kid2* to *newPop*;
- **end-while**
- $oldPop = newPop$;
- **end-repeat**;
- Select the best chromosome K in *oldPop*;
- Eliminate redundancies from K;
- **return** the classifier $\mathcal{H}_c(Pos, Neg)$ associated with K.

Fig. 1. Evolutionary Process for category c

the application of ρ to K simply set $K^+[t] = 0$ (thus, the redundant rule $c \leftarrow t \in d, \cdots \neg(t \in d), \cdots$ is removed from the encoded classifier). After iteration completes, the elimination of *redundant* bits/terms from the best chromosome/classifier K is performed. A bit $K[t] = 1$ is redundant in a chromosome K, representing $\mathcal{H}_c(Pos, Neg)$, if the chromosome K' obtained from K by setting $K[t] = 0$ is such that $F_{c,\alpha}(K) = F_{c,\alpha}(K')$. This may happen if either $t \in Pos$ and $\Delta(t) \subseteq \cup_{t' \in (Pos - \{t\})} \Delta(t')$ (i.e., t does not provide any contribution to the classification of new documents) or $t \in Neg$ and $\Delta(t) \cap \cup_{t' \in Pos} \Delta(t') = \emptyset$ (i.e., t does not contribute to remove any misclassified document).

4 Learning Process

While Olex-GA is the search of a "best" classifier for category c *over the training set*, for *a given* input vocabulary, the learning process is the search of a "best" classifier for c *over the validation set*, for *all* input vocabularies. More precisely, the learning process proceeds as follows:

- *Learning*: for each input vocabulary
  - *Evolutionary Process*: execute a predefined number of runs of Olex-GA over the *training set*, according to Figure 1;
  - *Validation*: run over the *validation set* the best chromosome/classifier generated by Olex-GA;
- *Testing*: After all runs have been executed, pick up the best classifier (over the validation set), and assess its accuracy over the *test set* (unseen data).

The output of the learning process is assumed to be the "best classifier" of c.

5 Experimentation

5.1 Benchmark Corpora

We have experimentally evaluated our method using both the REUTERS-21578 [12] and and the OHSUMED [8] test collections.

The REUTERS-21578 consists of 12,902 documents. We have used the Mod-Apté split, in which 9,603 documents are selected for training (*seen data*) and the other 3,299 form the test set (*unseen data*). Of the 135 categories of the TOPICS group, we have considered the 10 with the highest number of positive training examples (in the following, we will refer to this subset as R10). We remark that we used *all* 9603 documents of the training corpus for the learning phase, and performed the test using *all* 3299 documents of the test set (including those not belonging to any category in R10).

The second data set we considered is OHSUMED, in particular, the collection consisting of the first 20,000 documents from the 50,216 medical abstracts of the year 1991. Of the 20,000 documents of the corpus, the first 10,000 were used as *seen data* and the second 10,000 as *unseen data*. The classification scheme consisted of the 23 MeSH *diseases*.

5.2 Document Pre-processing

Preliminarily, documents were subjected to the following pre-processing steps: (1) First, we removed all words occurring in a list of common stopwords, as well as punctuation marks and numbers; (2) then, we extracted all n-grams, defined as sequences of maximum three words consecutively occurring within a document (after stopword removal)[5]; (3) at this point we have randomly split the set of *seen data* into a *training set* (70%), on which to run the GA, and a *validation set* (30%), on which tuning the model parameters. We performed the split in such a way that each category was proportionally represented in both sets (stratified holdout); (4) finally, for each category $c \in \mathcal{C}$, we scored all n-grams occurring in the documents of the training set TS_c by each scoring function $f \in \{CHI, IG\}$, where IG stands for Information Gain and CHI for Chi Square [4, 22].

[5] Preliminary experiments showed that n-grams of length ranging between 1 and 3 perform slightly better than single words.

Table 1. Best classifiers for categories in R10 and micro-averaged performance

Cat.	Input ($\alpha = 0.5$)		Training (GA)	Test		Classifier	
	f	k	F-meas	F-meas	BEP	#pos	#neg
earn	CHI	80	93.91	95.34	95.34	36	9
acq	CHI	60	83.44	87.15	87.49	22	35
mon-fx	CHI	150	79.62	66.47	66.66	33	26
grain	IG	60	91.05	91.61	91.75	9	8
crude	CHI	70	85.08	77.01	77.18	26	14
trade	IG	80	76.77	61.80	61.81	20	11
interest	CHI	150	73.77	64.20	64.59	44	23
wheat	IG	50	92.46	89.47	89.86	8	6
ship	CHI	30	87.88	74.07	74.81	17	33
corn	IG	30	93.94	90.76	91.07	4	12
μ-BEP				86.35	86.40		

5.3 Experimental Setting

We conducted a number of experiments for inducing the best classifiers for both REUTERS-21578 and OHSUMED according to the learning process described in Section 4. To this end, we used different input vocabularies $V(k, f)$, with $k \in \{10, 20, \cdots, 90, 100, 150, 200\}$ and $f \in \{CHI, IG\}$. For each input vocabulary, we ran Olex-GA three times. The parameter α of the fitness function $F_{c,\alpha}$ was set to 0.5 for all the experiments (thus attributing equal importance to Precision and Recall).

In all cases, the population size has been held at 500 individuals, the number of generations at 200, the crossover rate at 1.0, the mutation rate at 0.001 and the elitism probability at 0.2.

For each chromosome K in the population, we initialized K^+ at random, while we set $K^-[t] = 0$, for each $t \in Neg*$ (thus, K initially encodes a classifier $\mathcal{H}_c(Pos, \emptyset)$ with no negative terms).

Initialization, as well as all the above mentioned GA parameters, have been empirically set based on a preliminary experimentation.

5.4 Experimental Results

The first data set we considered is REUTERS-21578 . Table 1 shows the performance of the best classifiers for the categories of R10. Here, for each classifier, we report: (1) the values of f and k of the input vocabulary $V(f, k)$; (2) the F-measure over the training set (i.e., the fitness of the best individual); (3) F-measure and Break Even Point (BEP) over the test set ("Test"); (4) the number of positive and negative terms occurring in the classifier ("Classifier").

From results of Table 1 it clearly appears that the evolutionary model is sensitive to the input parameters, as each category achieves its maximum performance with different values of k and f. For an instance, the best classifier of category

"earn" is learned from a vocabulary $V(k, f)$ where $k = 80$ and $f = CHI$; the respective F-measure and breakeven point are both 95.34. Looking at the last row, we see that the micro-averaged values of F-measure and BEP over the test set are equal to 86.35 and 86.40, respectively. Finally, looking at the last two columns labelled "classifier", we see that the induced classifiers are rather compact: the maximum number of positive terms is 44 ("interest"), and the minimum is 4 ("corn"); likewise, the maximum number of negative terms is 35 ("acq") and the minimum is 6 ("wheat").

As an example, the classifier $\mathcal{H}_c(Pos, Neg)$ for "corn" has

$$Pos = \{\text{“}corn\text{”}, \text{“}maize\text{”}, \text{“}tonnes\ maize\text{”}, \text{“}tonnes\ corn\text{”}\}$$

$$Neg = \{\text{“}jan\text{”}, \text{“}qtr\text{”}, \text{“}central\ bank\text{”}, \text{“}profit\text{”}, \text{“}4th\text{”}, \text{“}bonds\text{”}, \text{“}pact\text{”}, \text{“}offering\text{”}, \text{“}monetary\text{”}, \text{“}international\text{”}, \text{“}money\text{”}, \text{“}petroleum\text{”}\}.$$

Thus, $\mathcal{H}_c(Pos, Neg)$ is of the following form:

$$c \leftarrow \text{“}corn\text{”} \vee \cdots \vee \text{“}tonnes\ corn\text{”} \ \wedge \neg\ (\text{“}jan\text{”} \vee \cdots \vee \text{“}petroleum\text{”}).$$

To show the sensitivity of the model to parameter settings, in Table 2 we report the F-measure values (on the validation set) for category "corn" obtained by using different values for ϕ and ν. As we can see, the performance of the induced classifiers varies from a minimum of 84.91 to a maximum of 93.58 (note that, for both functions, the F-measure starts decreasing for $\nu > 50$, i.e., a reduction of the vocabulary size provides a benefit in terms of performance [22]).

Table 2. Effect of varying ϕ and ν on the F-measure for category "corn"

	ν				
ϕ	10	50	100	150	200
CHI	90.74	91.00	88.7	88.68	84.91
IG	92.04	93.58	91.59	91.59	89.09

The second data set we considered is OHSUMED . In Table 3 we provide the best performance for the ten most frequent Mesh categories and micro-averaged performance over all 23. In particular, micro-averaged F-measure and BEP (over the test set) are both equal to 62.30. Also in this case, with a few exceptions, classifiers are rather compact.

5.5 Time Efficiency

Experiments have been run on a 2.33 GHz Xeon 4 Gb RAM. The average execution time of the evolutionary process of Figure 1 is around 10 seconds per category for both data sets (recall that in both cases the population is made of 500 individuals which evolve for 200 generations).

Table 3. Best classifiers for the ten most frequent MeSH "diseases" categories of Ohsumed and micro-averaged performance over all 23

Cat.	Input ($\alpha = 0.5$)		Training (GA)	Test		Classifier	
name	f	k	F-meas	F-meas	BEP	#pos	#neg
C23	CHI	100	54.93	50.67	51.39	25	13
C14	CHI	70	79.67	75.46	75.61	32	7
C04	CHI	100	80.79	80.18	80.36	33	7
C10	CHI	70	61.63	54.60	54.65	24	25
C06	IG	100	73.51	68.69	68.80	95	3
C21	CHI	200	79.17	62.11	62.22	113	6
C20	IG	80	75.64	70.98	71.12	35	4
C12	CHI	100	81.59	73.60	74.28	55	7
C08	IG	30	69.63	63.99	64.07	18	24
C01	CHI	100	76.14	64.32	64.46	93	3
avg-BEP (top 10)				66.46	66.96		
μ-BEP (all 23)				62.30	62.30		

6 Performance Comparisons

In this section we compare Olex-GA with four classical learning algorithms: SVM (both, polynomial and radial basis function - rbf), Ripper (with two optimization steps), C4.5 and Naive Bayes (NB). Further, we make a comparison with the greedy approach to problem MAX-F we presented in our previous work [19] (hereafter called Olex-Greedy).

To determine the performance of the above algorithms (apart from Olex-Greedy) we utilized the Weka library of machine learning algorithms, version 3.5.6 [21].

In order to make results really comparable, documents of both corpora were preliminarily pre-processed exactly as described in Subsection 5.2. Then, for each of the above learners, we carried out a number of experiments over the training set, using the validation set for tuning the model parameters. In particular, we ran all methods with vocabularies consisting of 500, 1000, 2000, 5000 terms. For each vocabulary, polynomial SVM was run with degree d ranging from 1 to 5, while rbf SVM was executed with variance $\gamma \in \{0.2, 0.4, 0.6, 0.8, 1.0, 1.2\}$. Once completed all experiments, we selected the best classifier (over the validation set) of *each* category, i.e., the one with the maximum value of the F-measure. Finally, we used the test set to evaluate the performance of the best classifiers and the micro-averaged accuracy measures[6].

Reuters-21578 . The performance results of the tested algorithms are reported in Table 4. In particular, for each algorithm, we provide the BEP over

[6] To attain more reliable estimations of the differences in predictive accuracy of different methods, n-fold cross-validation along with statistical significance tests should be performed. We envision this task as part of our future work.

Table 4. Recall/Precision breakeven points on R10

Category	NB	C4.5	Ripper	SVM		Olex	
				poly	rbf	Greedy	GA
earn	96.61	95.77	95.31	97.32	96.57	93.13	**95.34**
acq	90.29	85.59	86.63	90.37	90.83	84.32	**87.49**
money	56.67	63.08	62.94	72.89	68.22	68.01	**66.66**
grain	77.82	89.69	89.93	92.47	88.94	91.28	**91.75**
crude	78.84	82.43	81.07	87.82	86.17	80.84	**77.18**
trade	57.90	70.04	75.82	77.77	74.14	64.28	**61.81**
interest	61.71	52.93	63.15	68.16	58.71	55.96	**64.59**
wheat	71.77	91.46	90.66	86.13	89.25	91.46	**89.86**
ship	68.65	71.92	75.91	82.66	80.40	78.49	**74.81**
corn	59.41	86.73	91.79	87.16	84.74	89.38	**91.07**
μ-BEP	82.52	85.82	86.71	89.91	88.80	84.80	**86.40**
learning times (min)	0.02	425	800	46	696	2.30	**116**

the test set of each category in R10, the micro-avg BEP and the overall learning time. Concerning predictive accuracy, with a μ-BEP of 86.40, our method surpasses Naive Bayes (82.52), which shows the worst behavior, and Olex-Greedy (84.80); is competitive with C4.5 (85.82) and Ripper (86.71), while performs worse than both SVM's (poly = 89.91, rbf = 88.80). Concerning efficiency, NB (0.02 min) and Olex-Greedy (2.30 min) are by far the fastest methods. Then poly SVM (46 min) and Olex-GA (116 min), followed at some distance by C4.5 (425 min), rbf SVM (696 min) and Ripper (800).

Ohsumed . As we can see from Table 5, with a μ-BEP = 62.30, the proposed method is the top-performer. On the other side, C4.5 shows to be the worst

Table 5. Recall/Precision breakeven points on the ten most frequent MeSH diseases categories of Ohsumed and micro-averaged performance over all 23

Category	NB	C4.5	Ripper	SVM		Olex	
				poly	rbf	Greedy	GA
C23	47.59	41.93	35.01	45.00	44.21	47.32	**51.39**
C14	77.15	73.79	74.16	73.81	75.34	74.52	**75.61**
C04	75.71	76.22	80.05	78.18	76.65	77.78	**80.36**
C10	45.96	44.88	49.73	52.22	51.54	54.72	**54.65**
C06	65.19	57.47	64.99	63.18	65.10	63.25	**68.80**
C21	54.92	61.68	61.42	64.95	62.59	61.62	**62.22**
C20	68.09	64.72	71.99	70.23	66.39	67.81	**71.12**
C12	63.04	65.42	70.06	72.29	64.78	67.82	**74.28**
C08	57.70	54.29	63.86	60.40	55.33	61.57	**64.07**
C01	58.36	48.89	56.05	43.05	52.09	55.59	**64.46**
avg-BEP (top 10)	61.37	58.92	62.73	62.33	61.40	62.08	**66.69**
μ-BEP (all 23)	57.75	55.14	59.65	60.24	59.57	59.38	**62.30**
learning times (min)	0.04	805	1615	89	1100	6	**249**

performer (55.14) (so confirming the findings of [11]). Then, in the order, Naive Bayes (57.75), rbf SVM (59.57), Ripper (59.65), polynomial SVM (60.24) and Olex-Greedy (61.57). As for time efficiency, the OHSUMED results essentially confirm the hierarchy coming out from the REUTERS-21578 .

7 Olex-GA vs. Olex-Greedy

One point that is noteworthy is the relationship between Olex-Greedy and Olex-GA, in terms of both predictive accuracy and time efficiency.

Concerning the former, we have seen that Olex-GA consistently beats Olex-Greedy on both data sets. This confirms Freitas' findings [6], according to which effectiveness of GA's in rule induction is a consequence of their inherent ability to cope with *attribute interaction* as, thanks to their global search approach, more attributes at a time are modified and evaluated as a whole. This in contrast with the local, one-condition-at-a-time greedy rule generation approach.

On the other hand, concerning time efficiency, Olex-Greedy showed to be much faster than Olex-GA. This should not be surprising, as the greedy approach, unlike GA's, provides a search strategy which straight leads to a suboptimal solution.

8 Relation to Other Inductive Rule Learners

Because of the computational complexity of the learning problem, all real systems employ heuristic search strategies which prunes vast parts of the hypothesis space. Conventional inductive rule learners (e.g, RIPPER [3]) usually adopt, as their general search method, a covering approach based on a separate-and-conquer strategy. Starting from an empty rule set, they learn a set of rules, one by one. Different learners essentially differ in how they find a single rule. In RIPPER, the construction of a single rule is a two-stage process: a greedy heuristics constructs an initial rule set (IREP*) and, then, one or more optimization phases improve compactness and accuracy of the rule set.

Also decision tree techniques, e.g., C4.5 [18], rely on a two-stage process. After the decision tree has been transformed into a rule set, C4.5 implements a pruning stage which requires more steps to produce the final rule set - a rather complex and time consuming task.

In contrast, Olex-GA, like Olex-Greedy and other GA-based approaches (e.g., [5]), relies on a single-step process whereby an "optimal" classifier, i.e., one consisting of few high-quality rules, is learned. Thus, no pruning strategy is needed, with a great advantage in terms of efficiency. This may actually account for the time results of Tables 4 and 5, which show the superiority of Olex-GA w.r.t. both C4.5 and Ripper.

9 Conclusions

We have presented a Genetic Algorithm, Olex-GA, for inducing rule-based text classifiers of the form "if a document d includes either term t_1 or ... or term t_n, but not term t_{n+1} and ... and not term t_{n+m}, then classify d under category c".

Olex-GA relies on a simple binary representation of classifiers (several-rules-per-individual approach) and uses the F-measure as the fitness function. One design aspect related to the encoding of a classifier $\mathcal{H}_c(Pos, Neg)$ was concerned with the choice of the length of individuals (i.e., the size of the search space). Based on preliminary experiments, we have restricted the search of Pos and Neg to suitable subsets of the vocabulary (instead of taking it entirely), thus getting more effective and compact classifiers.

The experimental results obtained on the standard data collections REUTERS-21578 and OHSUMED show that Olex-GA quickly converges to very accurate classifiers. In particular, in the case of OHSUMED , it defeats all the other evaluated algorithms. Further, on both data sets, Olex-GA consistently beats Olex-Greedy. As for time efficiency, Olex-GA is slower than Olex-Greedy but faster than the other rule learning methods (i.e., Ripper and C4.5).

We conclude by remarking that we consider the experiments reported in this paper somewhat preliminary, and feel that performance can further be improved through a fine-tuning of the GA parameters.

References

1. Alvarez, J.L., Mata, J., Riquelme, J.C.: Cg03: An oblique classification system using an evolutionary algorithm and c4.5. International Journal of Computer, Systems and Signals 2(1), 1–15 (2001)
2. Apté, C., Damerau, F.J., Weiss, S.M.: Automated learning of decision rules for text categorization. ACM Transactions on Information Systems 12(3), 233–251 (1994)
3. Cohen, W.W., Singer, Y.: Context-sensitive learning methods for text categorization. ACM Transactions on Information Systems 17(2), 141–173 (1999)
4. Forman, G.: An extensive empirical study of feature selection metrics for text classification. Journal of Machine Learning Research 3, 1289–1305 (2003)
5. Freitas, A.A.: A genetic algorithm for generalized rule induction. In: Advances in Soft Computing-Engineering Design and Manufacturing, pp. 340–353. Springer, Heidelberg (1999)
6. Freitas, A.A.: In: Klosgen, W., Zytkow, J. (eds.) Handbook of Data Mining and Knowledge Discovery, ch. 32, pp. 698–706. Oxford University Press, Oxford (2002)
7. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
8. Hersh, W., Buckley, C., Leone, T., Hickman, D.: Ohsumed: an interactive retrieval evaluation and new large text collection for research. In: Croft, W.B., van Rijsbergen, C.J. (eds.) Proceedings of SIGIR-1994, 17th ACM International Conference on Research and Development in Information Retrieval, Dublin, IE, pp. 192–201. Springer, Heidelberg (1994)
9. Homaifar, A., Guan, S., Liepins, G.E.: Schema analysis of the traveling salesman problem using genetic algorithms. Complex Systems 6(2), 183–217 (1992)

10. Hristakeva, M., Shrestha, D.: Solving the 0/1 Knapsack Problem with Genetic Algorithms. In: Midwest Instruction and Computing Symposium 2004 Proceedings (2004)
11. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 137–142. Springer, Heidelberg (1998)
12. Lewis, D.D.: Reuters-21578 text categorization test collection. Distribution 1.0 (1997)
13. Lewis, D.D., Hayes, P.J.: Guest editors' introduction to the special issue on text categorization. ACM Transactions on Information Systems 12(3), 231 (1994)
14. Michalewicz, Z.: Genetic Algorithms+Data Structures=Evolution Programs, 3rd edn. Springer, Heidelberg (1999)
15. Noda, E., Freitas, A.A., Lopes, H.S.: Discovering interesting prediction rules with a genetic algorithm. In: Proc. Congress on Evolutionary Computation (CEC-1999), July 1999. IEEE, Washington (1999)
16. Pei, M., Goodman, E.D., Punch, W.F.: Pattern discovery from data using genetic algorithms. In: Proc. 1st Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD-1997), Febuary 1997. World Scientific, Singapore (1997)
17. Jung, Y., Jog, S.P., van Gucht, D.: Parallel genetic algorithms applied to the traveling salesman problem. SIAM Journal of Optimization 1(4), 515–529 (1991)
18. Quinlan, J.R.: Generating production rules from decision trees. In: Proc. of IJCAI-1987, pp. 304–307 (1987)
19. Rullo, P., Cumbo, C., Policicchio, V.L.: Learning rules with negation for text categorization. In: Proc. of SAC - Symposium on Applied Computing, Seoul, Korea, March 11-15 2007, pp. 409–416. ACM, New York (2007)
20. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys 34(1), 1–47 (2002)
21. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
22. Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. In: Fisher, D.H. (ed.) Proceedings of ICML-97, 14th International Conference on Machine Learning, Nashville, US, pp. 412–420. Morgan Kaufmann Publishers, San Francisco (1997)