Evolving text classification rules with genetic programming

HIRSCH, Laurence <http://orcid.org/0000-0002-3589-9816>, SAEEDI, Masoud and HIRSCH, Robin

Article Title: Evolving Text Classification Rules with Genetic Programming

Authors:
1. Laurence Hirsch,
School of Management
Royal Holloway University of London
Egham
Surrey
TW20 OEX
Email: Laurence.Hirsch@rhul.ac.uk

2. Masoud Saeedi
School of Management
Royal Holloway University of London
Egham
Surrey
TW20 OEX

3. Robin Hirsch
Department of Computer Science
University College London
Gower Street
London WC1E 6BT

Abbreviated Title: Evolving Text Classification Rules with GP

Evolving Text Classification Rules with Genetic Programming

Abstract.
We describe a novel method for using Genetic Programming to create compact classification rules using combinations of N-Grams (character strings). Genetic programs acquire fitness by producing rules that are effective classifiers in terms of precision and recall when evaluated against a set of training documents. We describe a set of functions and terminals and provide results from a classification task using the Reuters 21578 dataset. We also suggest that the rules may have a number of other uses beyond classification and provide a basis for text mining applications.

Key Words: text classification, Genetic Programming, N-Gram.

1 Introduction

Automatic text classification is the activity of assigning pre-defined category labels to natural language texts based on information found in a training set of labelled documents. In recent years it has been recognised as an increasingly important tool for handling the exponential growth in available online texts and we have seen the development of many techniques aimed at the extraction of features from a set of training documents, which may then be used for categorisation purposes.

In the 1980's a common approach to text classification involved humans in the construction of a classifier, which could be used to define a particular text category. Such an expert system would typically consist of a set of manually defined logical rules, one per category, of type

```
if {DNF formula} then {category}
```

A DNF ("disjunctive normal form") formula is a disjunction of conjunctive clauses; the document is classified under a category if it satisfies the formula i.e. if it satisfies at least one of the clauses. An often quoted example of this approach is the CONSTRUE system (Hayes et al. 1990), built by Carnegie Group for the Reuters news agency. A sample rule of the type used in CONSTRUE to classify documents in the 'wheat' category of the Reuters dataset is illustrated below.

```
if  ((wheat & farm) or
(wheat & commodity) or
(bushels & export)  or
(wheat & tonnes)    or
(wheat & winter & ¬ soft))
then
WHEAT  else  ¬ WHEAT
```

Such a method, sometimes referred to as 'knowledge engineering', provides accurate rules and has the additional benefit of being human understandable. That is, the definition of the category is meaningful to a human, thus producing additional uses of the rule including verification of the category. However the disadvantage is that the construction of such rules requires significant human input and the human needs some knowledge concerning the details of rule construction as well as domain knowledge (Apt´e et al. 1994).

Since the 1990's the machine learning approach to text categorisation has become the dominant one. In this case the system requires a set of pre-classified training documents and automatically produces a classifier from the documents. The domain expert is needed only to classify a set of existing documents. Such classifiers, usually built on the frequency of particular words in a document (sometimes called 'bag of words'), are based on two empirical observations regarding text:

1. the more times a word occurs in a document, the more relevant it is to the topic of the document.
2. the more times the word occurs throughout the documents in the collection the more poorly it discriminates between documents.

A well known approach for computing word weights is the term frequency inverse document frequency (tf-idf) weighting (Salton and McGill 1983) which assigns the weight to a word in a document in proportion to the number of occurrences of the word in the document and in inverse proportion to the number of documents in the collection for which the word occurs at least once, i.e.

$$a_{ik} = f_{ik} \log\left(\frac{N}{n_i}\right)$$

where a_{ik} is the weight of word i in document k, f_{ik} is the frequency of word i in document k, N the number of documents in the collection and n_i equal to the number of documents in which a_i occurs at least once. A classifier can be constructed by mapping a document to a high dimensional feature vector, where each entry of the vector represents the presence or absence of a feature (Salton et al. 1996; Joachims, 1998). In this approach, text classification can be viewed as a special case of the more general problem of identifying a category in a space of high dimensions so as to define a given set of points in that space. This is usually accompanied by some form of feature reduction such as the removal of non-informative words (stop words) and by the replacing of words by their stems, so losing inflection information. Such sparse vectors can then be used in conjunction with many learning algorithms for computing the closeness of two documents and quite sophisticated geometric systems have been devised (Bennet et al. 2000; Anthony 2003).

Although this method has produced accurate classifiers there are a number of drawbacks from the machine learning approach as compared to a rule based one.

1. All the word order information is lost; only the frequency of the terms in the document is stored.

2. The approach cannot normally identify word combinations, phrases or multi-word units e.g. ‘information processing’ (Pickens and Croft 2000).
3. If word stemming is used inflection information is also lost.
4. The classifier (the vector of weights) is not human understandable.

In this paper we describe a method to evolve compact human understandable rules using only a set of training documents. The system uses genetic programming (GP)(Koza 1992) to produce a synthesis of machine learning and knowledge engineering with the intention of incorporating advantageous attributes from both. The rules produced by the GPs are based on N-Grams (sequences of N letters) and are able to use a wide variety of features including word combinations and negative information for discrimination purposes. In the next section, we introduce GP and review previous classification work with N-Grams and with phrases. We then provide information concerning the implementation of our application and the initial results we have obtained on a text classification task.

1.1 GP Background

GP is a widely used method for automatically producing computer programs based on a high level specification of a particular problem known as the fitness test. Like the Genetic Algorithm (GA) GP is based on the principle of natural selection namely variation in reproductive success correlated with variation in a trait. Unlike GAs the individuals in a GP population are in the form of executable computer programs. GP works by iteratively applying genetic transformations, such as crossover and mutation, to a population of individual programs with the intention of creating better performing individuals as measured by the fitness test in subsequent generations. Although GP has been used in a textual environment (Clack et al. 1997; Bergström et al. 2000) it has not previously been used to evolve classifiers based on evolving N-Gram patterns.

1.2 N-Grams

A character N-Gram is an N-character slice of a longer string. For example the word INFORM produces the 5-grams _INFO, INFOR, NFORM, FORM_ where the underscore represents a blank. The key benefit of N-Gram-based matching derives from its very nature: since every string is decomposed into small parts any errors that are present tend to affect only a limited number of those parts leaving the remainder intact. The N-Grams for related forms of a word (e.g., 'information', 'informative', 'informing', etc.) automatically have a lot in common. If we count N-Grams that are common to two strings, we get a measure of their similarity that is resistant to a wide variety of grammatical and typographical errors (Cavnar and Trenkle 1994; Damashek 1995). The N-Gram representation has proven a robust alternative to word stemming, having the further advantage of requiring no linguistic preparations (Biskri and Delisle 2002). Since N-Grams are simply defined as sequences of characters they constitute a valuable basis for a language-independent text classifier.

A further useful property of N-Grams is that the lexicon obtained from the analysis of a text in terms of N-Grams of characters cannot grow larger than the size of the alphabet to the power of N. Furthermore, because most of the possible sequences of N characters rarely or never occur in practice for N>2, a table of the N-Grams occurring in a given text tends to be sparse, with the majority of possible N-Grams having a frequency of zero even for very large amounts of texts. For example, (Ebert et al. 1997) found that 40 MB of text from the Wall Street Journal contained only $2.7*10^5$ different 5-grams out of a possible $7.5*10^{18,}$ based on an alphabet of 27 characters. Tauritz (2000) and later Langdon (2000) used this property to build an: adaptive information filtering system based on weighted trigram (N=3) analysis in which genetic algorithms were used to determine weight vectors. An interesting modification of N-Grams is to generalise N-Grams to substrings which need not be contiguous. Lodhi et al. (2002) define a learning algorithm that uses non-contiguous substrings of N characters, but with a penalty for any gaps occurring between the N characters.

1.3 Phrases

The notion of N-Grams of words i.e. sequences or occurrences of N contiguous and non-contiguous words (with N typically equals to 2, 3, 4 or 5) has produced good results both in language identification, speech analysis and in several areas of knowledge extraction from text (Feldman et al. 1998; Ahonen-Myka 1999; Karanikas et al. 2000; Merkel and Andersson 2000; Tan et al. 2002). Pickens and Croft (2000) make the distinction between 'adjacent phrases' where the phrase words must be adjacent and Boolean phrases where the phrase words are present anywhere in the document. They found that adjacent phrases tended to be better than Boolean phrases in terms or retrieval relevance but not in all cases. Restricting a search to only adjacent phrases means that some retrieval information is lost. The implementation described below is able to make use of both adjacent and Boolean phrases if they are found to aid discrimination between documents.

2 Implementation

When building text classifiers there are usually a variety of options regarding pre-processing of documents and particular parameters values. Examples include whether to remove stop words, to stem words to a common form, to use words or N-Grams as terms and whether to search for single terms, phrases or particular sequences of terms. Where N-Grams or phrases are used the length of the phrase or N-Gram must also be determined. Although many of these options have been researched (Berleant and Gu 2000) it is often the case that effects on the performance of the classifier will depend on the particular classifier and the particular text environment (Sebastini 2002). We have developed a GP system where many of these decisions are either made redundant or are taken by the individual GPs.

We summarise the key features below:

- The basic unit (or phrase unit) we use is an N-Gram (sequence of N characters).
- N-Gram based rules are produced by GPs and evaluate to true or false for a particular document.

- Boolean functions such as EXISTS, AND, OR and NOT are included in the GP function set.
- Each GP produces a rule, which evaluates to true or false for any document (or text unit).
- A classification rule must be evolved for each category *c*. Fitness is then accrued for GPs producing classification rules which are true for training documents in *c* but are not true for documents outside *c*. Thus the documents in the training set represent the fitness cases.

2.1 Data Set

The task involved categorising documents selected from the Reuters-21578 test collection which has been a standard benchmark for the text categorisation tasks throughout the last ten years (Sebastiani 2002). Reuters-21578 is a set of 21,578 news stories which appeared in the Reuters newswire in 1987, classified according to 135 thematic categories, mostly concerning business and economy. Generally researchers have split the documents in the collection into a training set used to build a classifier and a test set used to evaluate the effectiveness of that classifier. With reference to this collection we should note that:

- The distribution of the documents across the categories is highly skewed, in the sense that some categories have very few documents classified under them while others have thousands.
- There are several semantic relations among the categories (e.g. there is a category WHEAT and a category GRAIN), but there is no explicit hierarchy defined on the categories.
- The collection is also fairly challenging for text categorisation systems since several categories have (under any possible split between training and test documents) very few training examples, making the inductive construction of a classifier a hard task.

Unfortunately, the benefits to text classification research that Reuters-21578 has brought about have been limited by the fact that different researchers have used different sub-collections and tested their systems on one of these sub-collections only. The most common direction for extracting a sub-collection out of Reuters-21578 has been that of restricting the attention to a subset of categories only. The subsets that have been most frequently used in text categorisation experimentation are:

- The set of the 10 categories with the highest number of positive training examples (often referred to as the 'Reuters top 10').
- The set of the 90 categories with at least one positive training example and one positive test example.
- The set of the 115 categories with at least one training example.

Systems that have been tested on these different Reuters-21578 subsets or with different test/train splits are thus not readily comparable (Yang and Liu 1999). In our experiments we use the "ModApt´e split", a partition of the collection into a training set and a test set that has been widely adopted by text categorisation experimenters. The top 10 categories are also widely used (for example (Lodhi et al. 2002)) and these are the categories we adopt here. As an illustration of the problems of using different subsets we found that the 'wheat rule' shown above is not an effective classifier for the wheat category when used on the top 10 subset although it was reported as a highly accurate classifier for the particular subset used by the CONSTRUE system.

2.2 Pre-Processing

Before we start the evolution of classification rules a number of pre-processing steps are made.

1. All the text in the document collection is placed in lower case.
2. Numbers are replaced by a special character and non-alphanumeric characters are replaced by a second special character.

3. All the documents in the collection are searched for N-Grams which are then stored in sets for size of N=2 to N=max_size. The size of these sets is reduced by requiring that an N-Gram occur at least 4 times before being included in a set.

The use of N-Grams as features makes word stemming unnecessary and the natural screening process provided by the fitness test means that a stop list is not required. Note that only step 3 is actually essential for the GP system to run. Including upper case letters and numbers would significantly increase the search space of the GP system but could provide useful features for discriminating between documents in particular domains. Note also that it may be useful to separate training documents in the category of the evolving classifier from all other documents in step 3. We can then specify N-Grams to be used for positive information from those to be used for negative information i.e. in conjunction with a NOT function (see below).

2.3 Fitness

GPs are set the task of assembling single letters into N-Gram strings and then combining N-Grams with Boolean functions to form a rule. The rule is then evaluated against the documents in the training set. Each rule can be tested against any document and will return a Boolean value indicating whether the rule is true for that document. An example of a rule produced by a GP evolving a classifier for the crude category of the Reuters 21578 is

```
(AND (EXISTS crude) (EXISTS (OR nerg barr)))
```

For this rule to be true for a document the string ‘crude’ AND either the string ‘nerg’ OR ‘barr’ must be substrings of that document. Note that the last two N-Grams are substrings of the words ‘energy’ and ‘barrel’.

A classification rule must be evolved for each category *c*. Each rule is actually a binary classifier, that is it will classify documents as either in the category or outside

the category. When evolving a rule for a particular category *c* the fitness depends on the number of documents in the category where the rule is true and the number of documents outside the category where the rule is true.

In information retrieval and text categorisation the F1 measure is commonly used for determining classification effectiveness and has the advantage of giving equal weight to precision and recall (Van Rijsbergen 1979). F1 is given by

$$F1(\pi, \rho) = \frac{2(\pi, \rho)}{\pi + \rho}$$

where:

Recall (π)= the number of relevant documents returned/the total number of relevant documents in the collection

Precision (ρ)= the number of relevant documents returned /the number of documents returned.

F1 also gives a natural fitness measure for an evolving classifier. The fitness of an individual GP is therefore assigned in the following way:

1. evaluate the rule produced by the GP against all documents in the training set.
2. calculate precision, recall and F1 by counting the documents where the rule is true in the category and outside the category for which the classifier is being evolved.
3. compute standardised fitness as 1 – F1 so that 0 is given to the most fit individual (a perfect classifier for that category).

2.4 GP Types

We use a strongly typed tree based GP (Montana 1995) system with types shown in Table **1**:

A critical prerequisite of a GP system is that every program in the population is executable i.e. will not crash. In most GP systems type mismatch errors are avoided by requiring that all functions and terminals are of the same type (only one type is allowed in any GP). In strongly typed GP a set of types can be allowed but when a GP is created or perturbed it is necessary for the system to ensure that any function is passed values of a legal type as arguments. In our system each GP will output a rule returning a Boolean value when evaluated against a document but will include string values as part of the rule.

2.5 GP terminals

Terminals in GP are the leaf nodes of a program tree and can be thought of as functions of arity 0. For example in the program:

```
(OR (EXISTS aize) (EXISTS corn))
```

the values ‘aize’ and ‘corn’ are the GP terminals and OR and EXISTS are the functions. EXISTS takes a String argument and returns a Boolean value whilst OR takes two Boolean values and returns a Boolean value. Terminals are most commonly numeric in GP, however in our system we use the following character literals stored as string values.

26 lower case alphabetic characters (a-z).
“~” meaning the space character
“#” meaning any number.

"^" meaning any non-alphanumeric character.

Note that for particular domains it may be useful to include numbers (still stored as strings), upper case characters and other special characters although this will increase the search space of the GP system.

2.6 GP Functions

The GPs are provided with protected string handling functions for combining characters into N-Gram strings and concatenating N-Grams into a longer N-Gram. Most combinations of letters above an N-Gram size of 2 are unlikely to occur in any text, for example where N-Gram size =6 only 0.001% of possible N-Grams are found in the Reuters 21578 (see Figure 1.).

We therefore guide the GPs through the vast search space of possible N-Gram patterns by the provision of protected 'EXPAND' function. The function initially forms a new N-Gram by combining two N-Grams (N-Gram1 and N-Gram2). A new N-Gram NewN-Gram is formed by appending N-Gram2 to N-Gram1. If the NewN-Gram is of length *l* the EXPAND function checks if the NewN-Gram is in the set of N-Grams of size *l* originally extracted from all the text in the all the training documents used. If it is found in this set NewN-Gram is returned. If it is not found, i.e. the N-Gram did not occur in the documents of the training set, the next N-Gram in the set (in alphabetical order) is returned. If the set is empty the original N-Gram N-Gram1 is returned. Note that the 'EXPAND' function will not return an N-Gram consisting of a sequence of characters which never occurs in the training data. The function is summarised by the pseudo code below

```
SetOfL-Grams = the set of N-Grams of length L found in
documents from the training set


EXPAND (N-Gram1, N-Gram2)
  NewN-Gram = (concatenate N-Gram1, N-Gram2)
  L = (sizeOf NewN-Gram)

  IF (IsEmpty SetOfL-Grams)
    RETURN N-Gram1
  ELSE IF (IsMember NewN-Gram SetOfL-Grams)
    RETURN NewN-Gram
    ELSE
      RETURN Next N-Gram in SetOfL-Grams
```

We found that using an unprotected concatenation function it was quite rare for N-Grams of size greater than 2 to be evolved. However using the EXPAND function long N-Grams and words are easily and commonly evolved by combining shorter strings. For example the string ‘wheat’ could be evolved in the following way

```
(EXPAND w (EXPAND (EXPAND h e) (EXPAND a b)))
```

The function initially creates the string ‘wheab’. This string is not found in the set of N-Grams of size 5 originally extracted from the collection. The next N-Gram in the set of 5-Grams is therefore returned (‘wheat’). The size of the N-Gram sets can be reduced by extracting them only from the text of documents belonging to the category for which the library is being evolved. This will greatly reduce the search space of the GP system but some discriminating ability will be lost where a Boolean NOT function is used in a GP produced rule.

Table 2 shows a basic set of GP functions for evolving classification rules. Note that some Boolean functions take Boolean arguments whilst others take string arguments.

Although the functions ANDSTR, ORSTR, and NOTSTR are not essential as they are definable by the other operators, we include them as a way reducing tree sizes.

2.7 GP Parameters

The GP parameters used in our experiments are summarised in Table 3. All of these are fairly standard in tree based GP systems. For those not familiar with these it is worth noting the following points:

- An individual GP is selected according to fitness and can be simply copied into the then next generation (reproduction) or part of the program may be randomly changed (mutation) or most commonly parts of the program are exchanged with another selected program to create two new individuals (crossover). The probabilities of these 3 possibilities are determined by the parameters in Table 3.
- Maximum tree depth parameters are usually critical in GP systems. Without these evolution will tend to produce very large programs which will use increasing amounts of machine resources and slow the entire system.
- Elitism is sometimes used in GP and GA and ensures that the best program or programs are always copied into the next generation.
- Automatically Defined Functions (ADFs) provide a method for subroutine use in a GP (Koza 1999).

3 Experiments

3.1 Objectives

The objective of our experiments were two fold:

1. To evolve effective classifiers against the text dataset (Reuters 21578 top 10).

2. To automatically produce compact human understandable rules with minimal features.

An alternative method using a library of rules tested on the 20 Newsgroup dataset is described in (Hirsch et al. 2004).

3.2 Evolution

In generation 0 of any GP run we have a set of (800) randomly created rules and these are all measured for fitness by testing them against the documents in the training set. Although we are unlikely to see a useful rule produced at this stage there will be some differentiation in performance between individuals in the population. These differences are exploited so that better programs are more likely to participate in the creation of the next generation. **Figure 2** shows a fairly typical pattern of evolution and in this case we see the emergence of a useful rule after approximately 20 generations. Precision is very high during the early evolution but is reduced as recall improves. In other cases we see recall starting very high and reducing as precision improves. In general we will see an improvement in F1 as measured against the training set and a corresponding but lower F1 as measured against the test set.

3.3 Example of Rule Induction: Reuters 21578 Category Crude

In this section we included an example of the evolution of a rule for the Reuters 21578 category "crude". Some good results can be evolved quite easily on this category e.g. the individual

```
(EXISTS barre)
```

often appears in generation 0 and achieves an F1 measure of 0.65 against the test data. The individual representing the rule

```
(ORSTR cru barre)
```

has been evolved in generation 4 and achieves an F1 measure of 0.729 when measured against the test data. The highly compressed rule using only 6 characters (disjunction of 2 tri-grams) appeared after 15 generations and was able to achieve and F1 measure of 0.802

```
(ORSTR rgy rud)
```

Note that 'rgy' is a substring of 'energy' and 'rud' is a substing of 'crude'. The best performing rule we have evolved in our experiments achieves a fitness (F1 measure) of 0.826 is shown below.

(OR (OR (OR (ORSTR arrels~ rude~) (EXISTS opec~) (EXISTS energy) (EXISTS oleum))))

3.4 Results

A classification rule was evolved for each category by using 4 GP runs and selecting the best rule to emerge from the 4 runs. GP contains a number of random elements and the use of multiple runs is commonly applied to improve the chances of finding a useful solution. The rule produced by the best individual for each category is shown in Table 4 together with the F1 measure (against the test set). Functions are shown in upper case and N-Grams are shown in lower case. The blank character is indicated by '~'.

The global macro-average F1 is 0.717 which compares favourably with other classifiers such as (Tan et al. 2002) although we should note that this is not a strictly controlled comparison. Indeed our intention at this point is not to produce the best classifier in terms of accuracy but to produce a good classifier which is based on a small number of features in a human understandable form. As can be seen some rules

are much easier to understand than others. However we argue that in contrast to other methods of automatic classification such as Support Vector Machines even the more complex rules are somewhat comprehensible to a human analyst.

GP's have a tendency to 'bloat' and to produce long forms of equivalent shorter programs by including redundant sections. For example the rule

```
(OR (ORSTR ~mone dollar~)(ORSTR ~mone dollar~))
```

is equivalent to the rule

```
(ORSTR ~mone dollar~)
```

One of the key objectives of our system was comprehensibility. We therefore applied a simple form of parsimony pressure such that where the F1 fitness of two programs was found to be equal the shorter program was assigned a better fitness value. With this method we were able to evolve the rules shown in Table 4 although they are still not necessarily in the most compact form. Comprehensibility may be improved by using other forms of parsimony pressure on the GP evolution and by favouring longer N-Grams or words.

Performance is also an important issue. In the experiments we described here to evolve a classifier for a single category requires that 800 rules be evaluated against all documents in the training set for each generation of every run. For our experiments we were using fairly modest machine resources (processor speed of 800Mhz and memory of 128MB) and we found that the average time to complete a run for one category was just under 20 minutes.

4 Discussion

Previous text classification systems have used various sets of features including words, word combinations and N-Grams. The system described here is capable of including any or all of these where they are found to be useful for classification

purposes. In addition the system can easily make use of negative information via the inclusion of Boolean NOT functions in the rule. The rule produced is in a form which can be easily manipulated so that it can be fed directly into a search engine such as a database or Internet search to retrieve similar texts. The rule is produced automatically but is somewhat similar to rules produced by knowledge engineering systems using human experts. For example the following rule was evolved for the Reuters Trade category happened to be in DNF form although it was not the most effective classifier (F1 0.692).

```
(OR (OR (OR (ANDSTR llion export) (OR (ANDSTR
llion surpl) (ANDSTR ~trad mport) (ANDSTR ~trad
vis) (ANDSTR ~trad yeutt)))
```

The rule created may also be used for purposes beyond classification such as text mining. For example, the regular occurrence of synonyms (different words with the same meaning) and homonyms (words with the same spelling but with distinct meanings) are key problems in the analysis of text data: in the language of relational databases this is a classic many-to-many relationship. There is some evidence that the rules evolved in our current system are using synonyms to improve the effectiveness of a rule, e.g.:

```
(ORSTR aize~ corn~)
```

Furthermore we suggest that homonyms are best discriminated by the use of contextual evidence, i.e. by an analysis of nearby strings in the text. We believe that much of this contextual evidence can be detected simply by the use of the Boolean operators AND, OR and NOT, though it may be that additional operators that impose constraints on the relative positions of two N-Grams in the text will allow an improved discrimination.

5 Future Work

We are investigating the usefulness of new GP functions:

- Special functions for identifying word order. For example FOLLOWS X Y (Bergström 2000) indicates that the word matched by N-Grams Y must follow the word matched by N-Gram X in the text of a document.
- Kleene's star (*) could be included as a marker for an arbitrary sequence of characters, e.g. a*t matches any of "at", "ant" or "agony aunt" within an N-Gram. We will also investigate the use of full regular expressions for the rules evolved by the GPs.
- Functions for identifying words that are ADJACENT in the text or NEAR one another.
- New functions together with numeric terminals for identifying frequency information may be introduced. Functions such as '>' return a Boolean value based on the frequency of a particular N-Gram in comparison to an integer terminal. This frequency could be a simple count of the occurrence of an N-Gram in a document or a more sophisticated measure such as the term frequency inverse document frequency (tf-idf) described above.

We believe that the system described here may be of particularly value when used in conjunction with other classification systems in a classification committee (Sebastiani 2002) because the method of producing the classifier is quite different to other automatic classifiers based on vectors of weights.

6 Conclusion

We have produced a system capable of discovering rules based on a rich and varied set of features which are useful to the task of discriminating between text documents. We suggest that there may a number of areas within automatic text analysis where the basic technology described here may be of use.

References

Ahonen-Myka, H. 1999. Finding All Maximal Frequent Sequences in Text. In *Proceedings of the 16th International Conference in Machine Learning* ICML Bled, Slovenia,

Apt´e, C., F. J. Damerau, and S. M. Weiss. 1994. Automated learning of decision rules for text categorization. *ACM Trans. on Inform. Syst. 12*, 3, 233–251.ATTARDI

Anthony N. 2003. Generalization error bounds for threshold decision lists. *Technical report*, LSE CDAM-LSE-2003-09.

Bennet K., J. Shawe-Taylor , D. Wu. 2000. Enlarging the margins in perceptron decision trees. *Machine Learning* 41, pp 295-313

Bergström, A., P. Jaksetic, P. Nordin. 2000. Enhancing Information Retrieval by Automatic Acquisition of Textual Relations Using Genetic Programming. In *Proceedings of the 2000 International Conference on Intelligent User Interfaces* (IUI-00), pp. 29-32, ACM Press.

Berleant, D., Z. Gu. 2000. Hash table sizes for storing n-grams for text processing, *Technical Report 10-00a*, Software Research Lab, 3215 Coover Hall, Dept. of Electrical and Computer Engineering, Iowa State University.

Biskri I., S. Delisle. 2002. Text Classification and Multilinguism: Getting at Words via N-grams of Characters. In *Proceedings of the 6th World Multiconference on Systemics*, Cybernetics and Informatics (SCI-2002), Orlando (Florida, USA), Volume V, 110-115.

Cavnar, W., J. Trenkle. 1994. N-Gram-Based Text Categorization In *Proceedings of SDAIR*-94, 3rd Annual Symposium on Document Analysis and Information Retrieval

Clack, C., J. Farrington., P. Lidwell. and T. Yu. 1997. Autonomous Document Classification for Business, in *Proceedings of The ACM Agents Conference.*

Damashek, M. 1995. Gauging similarity with n-grams: Language-independent categorization of text, *Science*, 267 pp. 843 . 848.

Ebert D., D. Shaw , A. Zwa , E. Miller, D. Roberts. 1997. Interactive Volumetric Information Visualization for Document Corpus Management, *Proceedings of Graphics Interface* .97, Kelowna, B.C.,121-128.

Feldman R., M. Fresko, Y. Kinar, O. Lindell,. M. Liphstat, Y. Rajman, O. Schler, O. Zamir. 1998. Text mining at the term level. In *Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 65--73, Nantes, France.

Hayes, P. J., P. M. Andersen, I. B. Nirenburg, and L.M. Schmandt. 1990. Tcs: a shell for content-based text categorization. In *Proceedings of CAIA-90, 6th IEEE Conference on Artificial Intelligence Applications* (Santa Barbara,CA, 1990), 320–326.

Hirsch, L., M. Saeedi, R. Hirsch 2004. Evolving Text Classifiers with Genetic Programming, In *Proceedings of the Euro-GP conference 2004*, Portugal, Springer.

Joachims, T. 1998. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of the l0th European Conference on Machine Learning* (ECML98) , pp 137-142.

Karanikas, H., C. Tjortjis, B. Theodoulidis. 2000 An Approach to Text Mining using Information Extraction *PKDD 2000 Conference*: Papers and Presentations, Springer-Verlag Publisher

Koza, J.R. 1992. Genetic Programming: On the Programming of Computers by Means of Natural Selection. *The MIT Press*, Cambridge MA

Koza, J.R., Bennet F, Andre D, Kean M, 1999 Genetic Programming III Darwinian Invention *and Problem Solving*, Morgan Kaufman Publishers

Langdon,W.B., 2000. Natural Language Text Classification and Filtering with Trigrams and Evolutionary Classifiers, *Late Breaking Papers at the 2000*

Genetic and Evolutionary Computation Conference, Las Vegas, Nevada, USA, editor Darrell Whitley, pages 210—217.

Lodhi H., J. Shawe-Taylor, N. Cristianini, C. Watkins 2001. Text classification using string kernels. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems* 13, pages 563--569. MIT Press

Merkel, M., M. Andersson. 2000. Knowledge-lite extraction of multi-word units with language filters and entropy thresholds. In *Proceedings of the 2000 Conf. User-Oriented Content-Based Text and Image Handling* (RIAO'00), pages 737--746, Paris, France.

Montana, D. 1995. Strongly Typed Genetic Programming. *In Evolutionary Computation*. 3:2, 199--230. The MIT Press, Cambridge MA.

Pickens, J., W.B. Croft. 2000. An Exploratory Analysis of Phrases in Text Retrieval. In *Proceedings of RIAO 2000 Conference*, Paris, France.

Salton, G., M.J. McGill M.J.1983. An Introduction to Modern Information Retrieval, *McGraw-Hil*l.

Salton,G., S. Singhal., C. Buckley, M. Mitra 1996. Automatic Text Decomposition Using Text Segments and Text Themes. In *Proceedings of the hypertext '96 Conference*, Washington D.C. USA.

Sebastiani, F. 2002. Machine learning in automated text categorization, *ACM Computing Surveys*, 34(1), pp. 1-47.

Tan, C.M., Y. F. Wang and C. D. Lee. 2002. The use of bigrams to enhance text categorization In *Information Processing and Management: an International Journal*, Vol 38, Number 4 Pages 529-546

Tauritz D.R., J.N. Kok, I.G. Sprinkhuizen-Kuyper. 2000. Adaptive information filtering using evolutionary computation, *Information Sciences*, vol.122/2-4, pp.121-140**.**

Van Rijsbergen, C.J. 1979. Information Retrieval, 2nd edition, Department of Computer Science, University of Glasgow

Yang, Y. and X. Liu. 1999. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, 42-49.

Table 1: GP Types

GP Type	Description
String	A sequence of one or more characters.
Boolean	True/False: the return type of all GPs

Table 2: GP Functions

Function Name	Number of Arguments	Type of Arguments	Return Type	Description
EXPAND	2	String	String	Concatenate 2 N-Grams and return the nearest N-Gram of the same length extracted from the training data. If found in the set of N-Grams extracted from the training data return that N-Gram else return the next N-Gram in the set.
EXISTS	1	String	Boolean	IF the N-Gram is found in a document return TRUE ELSE return FALSE
AND	2	Boolean	Boolean	Return arg1 AND arg2
OR	2	Boolean	Boolean	Return arg1 OR arg2
NOT	1	Boolean	Boolean	Return NOT arg1
ANDSTR	2	String	Boolean	IF arg1 AND arg2 are found in the document return TRUE ELSE return FALSE
ORSTR	2	String	Boolean	IF arg1 OR arg2 are found in the document return TRUE ELSE return FALSE
NOTSTR	1	String	Boolean	IF arg1 is NOT found in the document return TRUE ELSE return FALSE.

Table 3: GP Parameters

Parameter	**Value**
Population	800
Generations	50
Typing	Strongly typed
Creation Method	Ramped half and half
GP format	Tree Based
Selection type	Tournament
Tournament size	7
Mutation probability	0.1
Reproduction probability	0.1
Crossover probability	0.8
Elitism	No
ADF	No
Maximum tree depth at creation	9
Maximum tree depth	17
Maximum tree depth for mutation	4

Table 4: Rules evolved for Reuters top 10 categories

Name	F1	The Rule
Crude	0.826	(OR (OR (OR (ORSTR arrels~ rude~) (EXISTS opec~) (EXISTS energy) (EXISTS oleum))))
Corn	0.835	(ORSTR aize~ corn~)
Earn	0.857	(OR (ORSTR shr~ qt) (EXISTS ividend))
Grain	0.550	(OR (ORSTR ulture~ crop~) (EXISTS nnes~))
Interest	0.569	(OR (OR (AND (ORSTR engla deposit) (OR (NOTSTR vity) (EXISTS ny) (OR (AND (ORSTR lending epurcha) (ORSTR ~fut cut) (AND (OR (ANDSTR g-t ~l) (ORSTR ederal~ ~money~) (EXISTS further) (OR (AND (ORSTR epurc sbank) (NOT (EXISTS ny) (AND (OR (ANDSTR g-t bl) (ORSTR ngland~ ~money~) (NOT (EXISTS ny))))
money-fx	0.612	(ORSTR ~mone dollar~)
Ship	0.745	(OR (OR (ORSTR trike hips~) (ORSTR vesse river) EXISTS ipping~)))
Trade	0.761	(AND (ORSTR kore rade~) (OR (OR (AND (ORSTR ~yeu rade~) (ORSTR oods ficit) (ORSTR ~yeu domes) (ORSTR ~bil rplus)))
Wheat	0.663	(AND (NOTSTR prio) (AND (NOTSTR opme) EXISTS wheat))
Acq	0.755	(ORSTR cqui hares)

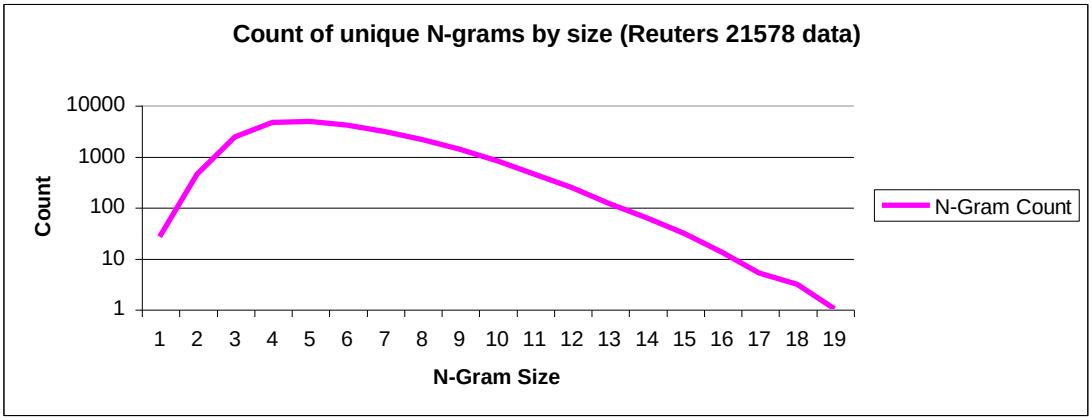

Figure 1. Count of unique N-Grams by size.

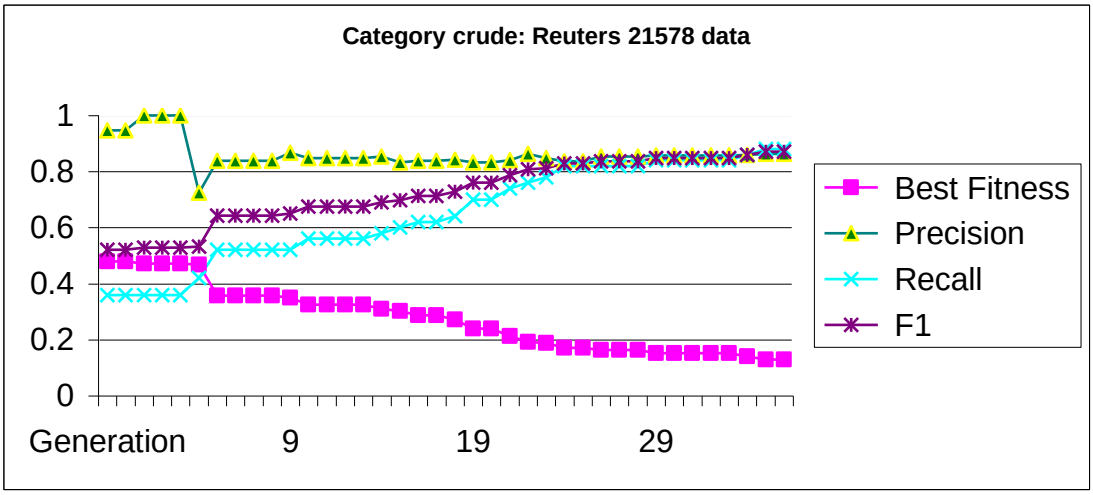

Figure 2: Evolution of a rule for the Reuters 21578 Crude category