Evolving rules for document classification

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

Evolving Rules for Document Classification

Laurence Hirsch[1], Masoud Saeedi[1], and Robin Hirsch[2]

[1]School of Management, Royal Holloway University of London, Surrey, TW20 OEX, UK
[2]University College London, Gower Street, London, WC1E 6BT, UK

Abstract. We describe a novel method for using Genetic Programming to create compact classification rules based on 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 because the induced rules are meaningful to a human analyst they may have a number of other uses beyond classification and provide a basis for text mining applications.

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 [1], 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 [2].

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 [3] 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. 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 [4]. 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. 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 [5].

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' [6].
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)[7] 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 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. Although GP has been used in a textual environment [8,9] it has not previously been used to evolve compressed classifiers based on evolving N-Gram patterns.

1.1 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 [10,11,12]. A 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. Tauritz [13] and later Langdon [14] 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. [15] define a learning algorithm that uses non-contiguous substrings of N characters, but with a penalty for any gaps occurring between the N characters.

1.2 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 [16,17,18]. Pickens and Croft [6] 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 Our Genetic Programming Approach

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 [19] it is often the case that effects on the performance of the classifier will depend on the particular classifier and the particular text environment [20]. 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.
- 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 [20]. 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 and these are the categories we adopt here.

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 (where max_size can be the longest word in the collection). 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.

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)))
```

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 [21]. F1 is given by

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

where:

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

Precision ( )= 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 a perfect classifier for that category.

2.4 GP Types

We use a strongly typed tree based GP [22] system with types shown in Table 1.

Table 1: GP Types

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

2.5 GP Terminals

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, with the majority of possible N-Grams having a frequency of zero even for very large amounts of texts. For example, 40 MB of text from the Wall Street Journal were found to contain only $2.7*10^5$ different 5-grams out of a possible $7.5*10^{18,}$ based on an alphabet of 27 characters [23]. We 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 appending one N-Gram to another. The EXPAND function checks if the new N-Gram is in the set of N-Grams of size *N* originally extracted from all the text in the all the training documents. If it is found the new N-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.

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').

Table 2. shows a basic set of GP functions for evolving classification rules. 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.

Table 2: GP Functions

Function Name	**No of Args**	**Type of Args**	**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.

2.7 GP Parameters

The GP parameters used in our experiments are summarised in Table 3.

Table 3: GP Parameters

Parameter	**Value**
Population	800
Generations	40
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

3 Experiments and Results

The objective of our experiments were two fold:

1. To evolve effective classifiers against the text dataset.
2. To automatically produce compact human understandable rules with minimal features.

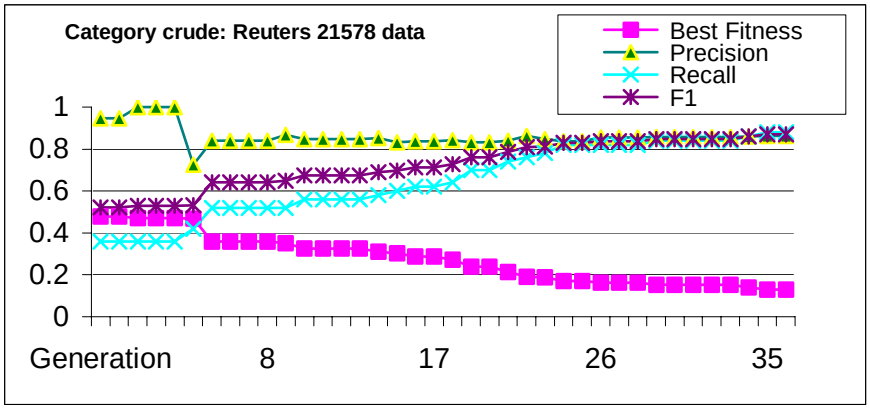

Fig. 1: Evolution of a rule for the Reuters 21578 Crude category

Fig. 1 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.

A classification rule was evolved for each category by using 4 GP runs and selecting the best rule to emerge from the 4 runs. 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 '~'.

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)

The global macro-average F1 is 0.717 which compares favourably with other classifiers such as [18] 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. Comprehensibility may be improved by using various forms of parsimony pressure on the GP evolution and by favouring longer N-Grams or words.

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 can be reformulated and fed directly into a database or Internet search engine 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 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. 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 Conclusions and Future Work

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.

We are investigating the usefulness of new GP functions:

- Special functions for identifying word order. For example FOLLOWS X Y [9] 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 [8]. 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 [20] because the method of producing the classifier is quite different to other automatic classifiers based on vectors of weights.

6 References

1. Hayes, P. J., Andersen, P.M., Nirenburg, I.B., Schmandt, L.M.: 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.
2. Apt´e, C., Damerau, F.J., Weiss, S.M.: Automated learning of decision rules for text categorization. ACM Trans. on Inform. Syst. 12, 3, 233–251.ATTARDI (1994)
3. Salton, G., McGill M.J.: An Introduction to Modern Information Retrieval, McGraw-Hill. (1983)
4. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In Proceedings of the l0th European Conference on Machine Learning (ECML 1998), pp 137-142.
5. Bennet K., Shawe-Taylor, J., Wu. D.: Enlarging the margins in perceptron decision trees. Machine Learning 41 (2000), pp 295-313
6. Pickens, J., Croft, W.B.: An Exploratory Analysis of Phrases in Text Retrieval. In Proceedings of RIAO Conference, Paris, France (2000).
7. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge MA (1992).
8. Clack, C., Farrington, J., Lidwell, P., Yu, T.: Autonomous Document Classification for Business, in Proceedings of The ACM Agents Conference (1997).
9. Bergström, A., Jaksetic, P. Nordin, P.: Enhancing Information Retrieval by Automatic Acquisition of Textual Relations Using Genetic Programming. In Proceedings of the 2000 International Conference on Intelligent User Interfaces, ACM Press. (2000) pp. 29-32,
10. Cavnar, W., Trenkle, J.: N-Gram-Based Text Categorization In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval (1994).
11. Damashek, M.: Gauging similarity with n-grams: Language-independent categorization of text, Science, 267 (1995) pp. 843 . 848.
12. Biskri I., Delisle, S. 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, (2002) 110-115.
13. Tauritz D.R., Kok, J.N., Sprinkhuizen-Kuyper I.G.: Adaptive information filtering using evolutionary computation, Information Sciences, vol.122/2-4, (2000) pp.121-140.
14. Langdon, W.B., 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, (2000) pages 210—217.

15. Lodhi H., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. In Leen, T.K., Dietterich, T.G., Tresp, V. editors, Advances in Neural Information Processing Systems 13, pages 563--569. MIT Press (2001).
16. Feldman R., Fresko M., Kinar Y., Lindell, O., Liphstat, M., Rajman, Y., Schler, O., Zamir, O.: 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 (1998).
17. Ahonen-Myka, H.: Finding All Maximal Frequent Sequences in Text. In Proceedings of the 16th International Conference in Machine Learning ICML Bled, Slovenia, (1999).
18. Tan, C.M., Wang, Y. F., Lee, C.D.: The use of bigrams to enhance text categorization In Information Processing and Management: an International Journal, Vol 38, Number 4 (2002) Pages 529-546
19. Berleant, D., Gu, Z.: 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. (2000).
20. Sebastiani, F.: Machine learning in automated text categorization, ACM Computing Surveys, 34(1), (2000), pp. 1-47.
21. Van Rijsbergen, C.J.:Information Retrieval, 2nd edition, Department of Computer Science, University of Glasgow (1979).
22. Montana, D.: Strongly Typed Genetic Programming. In Evolutionary Computation. 3:2, 199--230. The MIT Press, Cambridge MA. (1995).
23. Ebert, D., Shaw, D., Zwa, A. Miller, E. Roberts, D., Interactive Volumetric Information Visualization for Document Corpus Management, Proceedings of Graphics Interface .97, Kelowna, B.C., May 1997, 121-128