Incorporating Temporal Information for Document Classification

Xiao Luo
Faculty of computer science
Dalhousie University
6050 University Avenue, Halifax, NS, Canada
luo@cs.dal.ca

Nur Zincir-Heywood
Faculty of computer science
Dalhousie University
6050 University Avenue, Halifax, NS, Canada
zincir@cs.dal.ca

Abstract

In this paper, we propose a novel document classification system where the Recurrent Linear Genetic Programming is employed to classify documents that are represented in encoded word sequences by Self Organizing feature Maps. The results using different feature selection techniques on Reuters 21578 data set show that the proposed system can analyze the temporal sequence patterns of a document and achieve competitive performance on classification.

1. Introduction

Automatic text document analysis and management such as classification is more and more important to the information management area. Text representation and analysis are the two necessary and important steps for this purpose. The most widely used technique to represent a text document is so called 'bag of words', which considers only term frequencies or word occurrences in documents or categories and therefore ignores the significance of the sequences in which they occur. However, understanding and classifying a document can change based on different paragraphs or even sentences. Some of these changes are based on temporal sequences of words seen through the document.

In this paper, we present a study of a hybrid learning system utilizing temporal information for document categorization and text tracking. To do so, we analyze temporal word sequences of a document as words occur one after another in time throughout the whole document. The word sequence tracking is only based on the order of words without considering the semantics. The core of the approach is to employ machine-learning techniques to automate the identification of temporal sequences within a document. This in return automates the process of word tracking and document classification. There are two major parts to the proposed system. The first part is a new architecture for document representation. To this end, a hierarchical Self-Organizing Feature Map (SOM) encoding architecture is used to first encode the characters, then words. The similarities between words are discovered by the topology of the SOM, and then the whole document is represented by the neurons of the SOM [9]. The second part is a classifier to maximize the utilization of the encoded word sequence information for word tracking, and classification. Recurrent page-based Linear Genetic Programming (RLGP) is employed for this part. There are two advantages with RLGP. First, the registers in the recurrent mechanism are used to remember and analyze the word sequences. Second, Genetic Programming creates classification rules to track word sequences and to classify documents. The overall performance results on Reuters 21578 data set, which is a standard benchmark for the text classification tasks, show that the system can analyze temporal sequences of patterns of a document and get competitive performance on classification. The word tracking mechanism works as expected on both single-labeled and multi-labeled documents as shown in section 9.

The rest of this paper is organized as follows. Section 2 presents the related work and methodologies on the text data representation, analysis and classification; Section 3 gives an overview of the proposed system. Section 4 describes the document preprocessing and different feature selection techniques that are employed. The hierarchical SOM encoding architecture is described in section 5. Section 6 details the proposed document representation based on the SOM encoding architecture. The RLGP algorithm applied to the document classification problem is given in Section 7. In Section 8, results from our experimental study are presented. Finally, conclusions are drawn and the future work is discussed in Section 9.

2 Related Work

In the automatic text classification area, a text document needs to be uniformly represented before directly interpreted by an analysis or classification algorithm. The

conventional representation is 'bag of words' where a document is represented by a vector of words. However, with this representation, all word order information is lost. Moreover, word combinations, co-occurrences or multi-word units can not be identified since the method is only based on frequencies. Thus, other representations of the documents are investigated. In particular, those representations are based on the phrases in addition to individual words. Two kinds of phrase investigation approaches exist. One is based on the syntactical phrase where the phrase is defined according to a grammar of the language [8]. The other is based on the statistical phrase where the phrase is not grammatically such, but is composed of a set/sequence of words whose patterns of contiguous occurrence in the collection are statistically significant. An example of this representation is N-gram representation [4] [7]. Although the above methods inject positional information using phrases or multi-word units or local co-occurrence statistics, none of the representations above considers the significant sequences of words or phrases within the whole document.

In the area of automatic text analysis and classification, many techniques such as decision trees, Naive Bayesian, k-Nearest Neighbor, and Support Vector Machines have been researched [10]. The tree based GP classifier was also utilized for this task in 2005 [7]. In that work, authors used N-grams as the input to their GP classifier. Several designed GP functions were used to operate on the N-grams to generate classification rules. On the other hand, in the late 90's, some word sequence analysis techniques have been introduced, such as the recurrent neural network and the word-sequence kernel based Support Vector Machine. Wermter et al. [12] introduced the recurrent neural network for text classification in 1995 and 1999 respectively. In their system, a word is represented by a vector, which includes the information of the word to each category. This could mislead the classification process according to the category sequences instead of the actual word sequences. In 2003, Nicola Cancedda et al. [3] invented the word-sequence kernel based Support Vector Machine. Using this kernel, the similarity is assessed by the number of (possibly non-contiguous) matching sub-sequences shared by two word sequences. In that work, there is a predefined length for the sub-sequences. Thus, the classifier does not work on the word sequence of a document as a whole where the length is dynamic.

3. Overview of the Proposed System

As discussed earlier, the core of our approach is to employ machine learning techniques to automatically analyze word co-occurrences of a document in order to automate the process of word tracking by capturing temporal relationships. Words are the basic units of a document, however to start with, an encoding system, which can encode the symbolic representation of a word to a numerical representation is needed. In the world of automatic information processing, there are many different algorithms to achieve this objective. For example, in 'bag of words' representation, words can be encoded into 1 or 0 based on their occurrence (presence or absence) in the document. In our approach, the encoding system makes use of the mechanism of unsupervised learning to organize the words according to their occurrences, frequencies and similarities between each other from the perspectives of characters and their positions in the words. Specifically, the encoding system is a hierarchical SOM architecture that is used to first encode the characters, then words. The first level SOM is trained to discover the pattern distribution of characters within the whole corpus. We call this first level SOM as character encoding. Different SOMs are trained at the second level for each category to discover patterns in the distribution of words within a specific category. The second level SOMs are referred as word encoding. After encoding the words from their symbolic representation to their numerical representation, the proposed new document representation is constructed.

The second part of the proposed system, classification part, works on this new document representation to classify the documents to their corresponding categories. Since in our approach, the proposed document representation employs a temporal representation as opposed to a vector space mode (bag of words), a classification algorithm that is capable of working with this temporal information is required. Thus, a RLGP based classifier is developed and employed for this purpose. RLGP is based on the general page based linear genetic programming, which is derived from Genetic Algorithms.

Figure 1 shows the overview of the proposed system. The details of the encoding system, document representation, and the classification algorithm are given in the following sections.

4. Pre-Processing and Feature Selection

Indeed, as in other approaches, pre-processing and feature selection are employed to the input data of hierarchical SOM encoding architecture. The pre-processing steps include the removal of all the markup tags (such as <title>, <body>, etc.) and non-textual data (such as digital, special signs) as well as the removal of the stop words in the stop words list [1].

It is worth to mention that the stemming process is not employed, since all the words that have the same base form can be grouped together on the second level SOMs. The details of this process are given in Section 5.

In order to evaluate the proposed system and compare

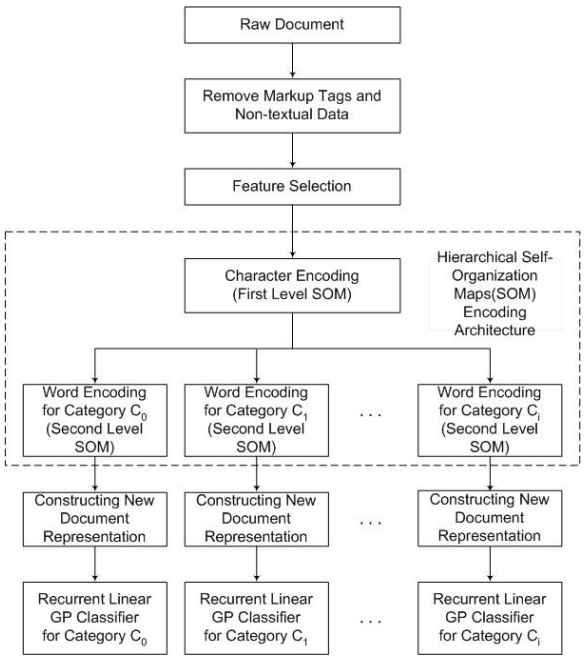

Figure 1. An overview of the system

with previous approaches, four different feature selection methods from are employed. The details of the feature selection methods are described below:

- Document Frequency (DF)

  Document Frequency can be formally defined as follows. Given a corpus X of t features by n documents, the DF value of feature f, DF_f, is defined as the number of documents in which f occurs at least once among the n documents. The assumption is that the words with small values of DF appear in a few documents. Therefore features with higher DF values are chosen as important features. In spite of its simplicity, DF has been demonstrated to be as effective as more advanced feature selection techniques in text categorization [11].

- Information Gain (IG)

  Information Gain measures the decrease in entropy due to the presence or absence of a word. It is reported that Information Gain performed best in some multi-class benchmarks [11]. Given a corpus which has $|C|$ categories $\{C_1...C_{|C|}\}$, the information gain of a feature f is defined in Equation 1:

$$IG(f) = -\sum_{j=1}^{j=m} P(C_j)\log P(C_j) + P(f)\sum_{j=1}^{j=m} P(C_j|f)\log P(C_j|f)$$

$$+P(\overline{f})\sum_{j=1}^{j=m} P(C_j|\overline{f})\log P(C_j|\overline{f}) \quad (1)$$

  In summary, if the information gain value of a feature is high, that feature is considered important and informative for a topic, otherwise it is ignored.

- Mutual Information (MI)

  Mutual information is a basic concept in information theory. It also involves the probabilities regarding the categories. Since this method is widely used in the area of text processing [6] [5], we used it to evaluate the proposed system as well. Mutual information is a measure of general interdependence between random variables. In the scenario of categorization, it is a measure of interdependence between feature and category. Specifically, given a feature f and a category C_j. The mutual information $MI(f, C_j)$ of a feature f corresponding to a category C_j is computed by using Equation 2.

$$MI(f, C_j) = \sum_{\{\overline{f},f\}}\sum_{\{\overline{C_j},C_j\}} P(f, C_j)\log\frac{P(f, C_j)}{P(f)P(C_j)} \quad (2)$$

  The feature f with higher mutual information value implies it is informative for category C_j, otherwise it is ignored.

- Frequent Nouns

  In terms of natural languages, nouns are more informative than verbs, adverbs, adjectives, prepositions and so on. Frequent nouns of each category could be the most informative features, so the Part-of-Speech(POS) tagging algorithm [2] is employed to select them. POS tagging is the process of marking up the words in a text with their corresponding parts of speech, such as noun, verb, adjective, preposition, pronoun, adverb, conjunction, and interjection. Common nouns and their plurals are marked as 'NNS' and 'NN' by the POS tagging algorithm. We assume that the frequent nouns are more informative than those infrequent ones. So all nouns of each category are ranked according to their frequency. Only those with higher frequency values are selected.

The number of features selected for each feature selection methods is summarized in table 1.

5. SOM Encoding Architecture

In this work, we step beyond the traditional vector space model representation and proposed a new one. The new representation is indeed based on a hierarchical SOM encoding architecture that is to organize the characters and words on

Table 1. Number of Selected Features for Each Feature Selection Method

Document Frequency [11]	1000 (whole corpus)
Information Gain [11] [14]	1000 (whole corpus)
Mutual Information [6] [5]	300 (per category)
Frequent Nouns	100 (per category)

the SOMs and then to encode them to their numerical representations. Specifically, it is a two-level hierarchical architecture. The first level of the SOM hierarchy is employed to discover patterns in the distribution of characters and the second level is employed to discover patterns in the distribution of words. The hierarchical nature of the model is shown in Figure 2.

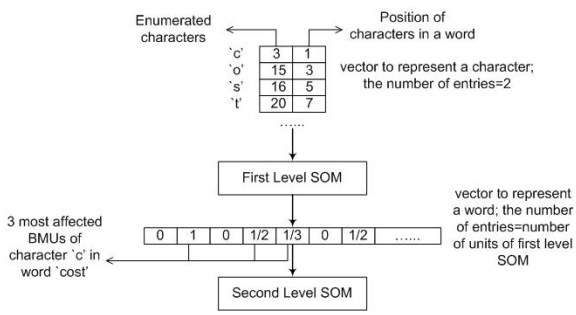

Figure 2. An overview of the hierarchical SOM encoding architecture

- First level SOM

  The assumption at this level is that the SOM forms a code-book for the patterns of characters that occur in a specific document corpus. In order to train an SOM to recognize patterns of characters, the document data must be enumerated in such a way as to distinguish characters and highlight the relationships between them. Characters can easily be represented by their ASCII representations. However, for simplicity, we enumerated them by the numbers 1 to 26, i.e. no differentiation between upper and lower case. The relationships between characters are represented by a character's position, or time index, in a word. For example, in the word '*cost*': '*c*' appears at time index 1, '*o*' appears at time index 2, '*s*' appears at time index 3, etc. So in all, each character is represented by a 2-dimensional vector. First dimension corresponds the numerical representation of the character itself, while the second dimension corresponds the numerical representation of the position of the character in the word. The inputs to the first level SOM are all the characters of words in the training data. It should be noted that it is important to repeat the characters as many times as they occur in the corpus, so that the neurons on the SOM will be more excited by those frequent characters and the data density will be more accurately represented by SOM. The overall process to construct 2-dimensional inputs for the first-level SOM are therefore:

  - Construct first dimension: represent characters in words by a number between 1 and 26.
  - Construct second dimension: represent the positions of characters in words by numbers equals to their time indexes in words times 2 and then minus 1

  The reason for not using the real time index of characters in words to construct the second dimension is that the maximum length of a word is usually about 13, so that by means of multiplying 2 to the real index of character and then deducting 1 makes the values of both dimensions to spread over a similar range. Thus, when training the SOM on these 2-dimensional inputs, there is less or no bias on either of the dimensions. Based on the observation of average weight change (AWC) [9] the size we used for the first level SOM is 7 by 13.

- Second Level SOM

  To construct the inputs to the second level SOM, three most affected BMUs from the 1st level SOM is selected to represent each character of a word. These three most affected BMUs correspond to the three closest units on the map to each input character in terms of Euclidian distance. The first most affected BMU means the Euclidian distance from it to the input character is the shortest. Thus, based on the trained map of the first level SOM, we construct a vector to represent a word by using the three most affected BMUs of each character in the word. The assumption at this level is that the SOM forms a code-book for the patterns in words that occur in a particular category. Specifically, three steps are used to construct a vector for a word:

  - Form a vector of size equal to the number of units(neurons) on the first level SOM (as mentioned, the size of first level SOM is 7 by 13, so the size of the vector is 91), where each entry of the vector corresponds to a unit/neuron on the first level SOM.
  - Initialize all entries of the vector to 0.

– For each character of word, increase entries in the vector corresponding to the 1st, 2nd and 3rd most affected BMUs by 1, 1/2, and 1/3, respectively.

Hence, each vector represents a word through the sum of all values of the BMUs' entries of its characters. It is worth to mention that we train different second level SOMs for each category. That is to say the inputs to a second level SOM of one category are the words of all documents of that category. Similar to the process of training the first level SOM, words are input to the 2nd level SOM as many times as they occur in the category (and in the same order), so that those frequent words and the density can be more accurately represented by the units and their distribution on the SOM. Based on the observation of AWC [9], the size we used for the second level SOM is 8 by 8. After training this level SOM, it can be visualized that those words have similar characters on close positions are projected to the same BMU or close BMUs, as shown in figure 3.

6. Proposed Document Representation

In this section, we introduce how the hierarchical SOM encoding architecture is used for the representation of a document.

6.1. Mapping Words to Units on the Second Level SOM

The assumption of the proposed system is that the documents from the same categories have some similar co-occurring words. Thus, by tracking those words, documents can be categorized to their corresponding category or categories. After training the second level SOMs, the words can be projected (or mapped) to their BMUs on the map, so that a document (after pre-processing and feature selection), which is consisted of a set of ordered words can be transferred into a set of ordered BMUs. If we use the index of the units on the map to represent the BMUs, the ordered BMUs can be represented as $\{8 \rightarrow 1 \rightarrow 43 \rightarrow 62 \rightarrow 56 \rightarrow 17 \rightarrow 64 \rightarrow 56 \rightarrow 62 \rightarrow 43 \rightarrow 8 \rightarrow 1 \rightarrow 16 \rightarrow 38 \rightarrow 63 \rightarrow 16\}$ shown in Figure 3. Thus, if documents are transferred into their corresponding sets of ordered BMUs, it is easy to visualize that two documents have common parts in their respective ordered BMUs if they belong to the same category.

6.2. Word and Document Representation

The output of a second level SOM is used to represent a document. Thus, instead of using all the BMUs of a second level SOM, we first select the most informative BMUs by analyzing the hit histogram of a second level SOM. We

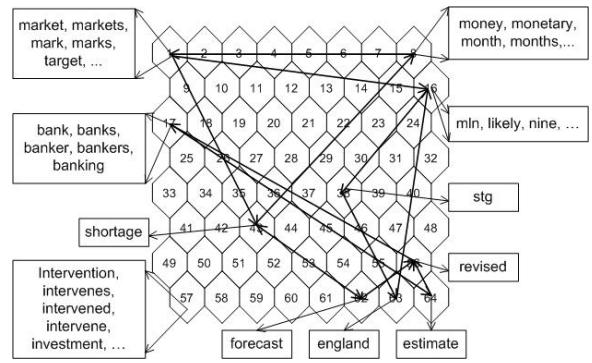

Figure 3. Mapping the ordered words to their BMUs

assume that BMUs that receive more hits are more informative to the category than the others. Thus, those units are used to represent words. BMUs that receive less hits are ignored. So, the volume of words to be input to the categorization system are reduced. This in return decreases the computational cost for classification. The number of BMUs to be chosen is based on the heuristic that each document in a category should at least have one word affecting (hitting) the selected BMUs, so that every document can be represented by the selected BMUs.

Then, we build Gaussian membership functions on those selected BMUs to differentiate the words that affect the same BMU. Gaussian kernel is employed for the training process of an SOM, so that each unit on the SOM is the Gaussian center of the words that affect it. Since each second level SOM is trained for category C_i, a word that is more closer to the Gaussian center is more likely to be a member word of that category C_i. Thus, a Gaussian membership function, Equation 3, of BMU W_i is used to determine if a word that affects BMU W_i is a member word of category C_i or not.

$$G(X_j, W_i) = \frac{1}{\sigma\sqrt{2\Pi}} e^{\frac{-(X_j - M)^2}{(2\sigma^2)}} \tag{3}$$

where M and σ^2denote the mean and variance of all words that affects BMU W_i, respectively. The algorithm to calculate σ^2 and M are shown in figure 4

If the Gaussian membership value of word $X_j$is $G(X_j, W_i) \geq \min_{i=1}^{N} G(X_i, W_i)$, where N is the total number of words that affect BMU W_i, then X_j is a member word of C_i. Otherwise, X_j is not a member word of C_i. Thus, we propose a word representation as an input to the classifier by using a two dimensional vector. The first dimension is the normalized index of its BMU and the second dimension is the Gaussian membership value of it to its BMU.

for i =1 to M

1. $A=\{X_0,X_1,\ldots,X_N\}$ X: word which neuron i is its BMU
2. for j=1 to N
   2.1 $d_j=\|X_j-W_i\|$
3. $mean(i)=W_i$
4. $variance(i)=\frac{1}{N-1}\sum_{j=1}^{N} d_j$

Figure 4. Constructing Gaussian membership functions for BMUs

7. Document Categorization and Analysis

As discussed earlier in the paper, an RLGP based classifier is developed in order to work on the proposed document representation for document categorization and analysis. To the best of our knowledge, this is the first attempt to utilize such a technique for document classification based on temporal representation. In this section, we first introduce the background knowledge for Linear Genetic Programming, then we detail the developed RLGP based classifier and the dynamic subset selection algorithm, which is employed to speed up the training process of the RLGP.

7.1. Linear Genetic Programming (LGP)

LGP is based on a representation closely related to that employed by Genetic Algorithms. Specifically, individuals are constructed from a (linear) sequence of integers each of which has to be decoded into a valid instruction (syntactic closure). The decoding process effectively translates each integer into an equivalent binary string, separates the string into a series of fields based on the addressing mode and maps each field into a valid value. Typical fields include mode, opcode, source and destination. The mode bit distinguishes between different instruction types, for example instructions detailing a constant or an operation performed on a register or on an input. The source and destination fields detail specific registers or input ports. Programs now take the form of a register level transfer language in which all operations operate on general purpose registers or read values from input ports (features from the current example). In this work, a 2-address instruction format is employed e.g.,$R1 = R1 + IP3$, where R1 denotes a register with index '1' and IP3 is a reference to the 3rd feature of the current pattern.

As with the case of Tree structured GP many instances of LGP have been developed over a considerable period of time [13]. The specific form of LGP employed by this work utilizes the page-based LGP developed in an earlier work [13]. Such a scheme enforces a fixed length representation (crossover only exchanges an equal number of instructions), the basic components of which are defined as follows.

- *Representation*: Individuals take the form of a 2-address instruction format. Individuals are described in terms of a (uniform) randomly selected number of pages, where each page has the same fixed number of instructions.

- *Initialization*: Individuals are described in terms of the number of pages and instructions, where instructions are selected from a valid set of integers denoting the instruction set. The number of pages per individual is determined through uniform selection over the interval [1, . . ., maxPages]. That is to say the initial population is initialized over the entire range of program lengths. Defining an instruction is a two-stage process in which the mode bit is first defined (instruction type) using a roulette wheel (user specifies the proportions of the three instruction types). Secondly the content of the remaining fields is completed with uniform probability. Such a scheme is necessary in order to avoid half of the instructions denoting constants i.e. effect of the mode field.

- *Selection Operators*: A steady-state tournament is employed. In this case all such tournaments are conducted with 4 individuals selected from the population with uniform probability. The two fittest individuals are retained and reproduce. The children over-write the worst two individuals from the same tournament using their respective position in the population.

- *Variation Operators*: Three variation operators are utilized, each with a corresponding probability of application, where such tests are applied additively (i.e., the resulting children might be the result of all three variation operators). Crossover selects one page from each offspring and swaps them. The pages need not be aligned, but always consist of the same number of instructions. Mutation has two forms. The first case - hereafter referred to as 'Mutation' - merely Ex-OR's an instruction with a new instruction. No benefits were observed in making such a mutation operator 'field specific', where this is undoubtedly a factor of the addressing format [13]. The second mutation operator - hereafter denoted 'Swap' - identifies two instructions with uniform probability in the same individual and interchanges them. The basic motivation being that an individual might possess the correct instruction mix, but have the instruction order incorrect.

This represents the basic page-based LGP scheme. However, the selection of page size is problem specific. As a consequence the Dynamic Page based LGP algorithm was introduced to modify the number of instructions per page

dynamically during the course of the training cycle [13]. In this case, the user merely defines the maximum page size as an order of 2. The page size is then doubled for each plateau in the fitness function, beginning with a page size of 1 and finishing at 'max page size' and returning to a page size of 1 once a plateau following 'max page size' is encountered. Plateaus are defined in terms of consecutive non-overlapping windows of 10 tournaments. For each of the 10 tournaments the (tournament's) best-case individual's fitness is summed. If the total over both windows is the same then a plateau is 'defined'. Such a scheme was demonstrated to be much more robust than that of a fixed page size over a range of benchmark problems (2 boxes, 6 parity, UCI classification benchmarks) [13]. In this work, we employed the dynamic page-based LGP.

7.2. Recurrent Linear Genetic Programming (RLGP)

RLGP is based on the standard page based LGP, the only modification necessary to change a standard page based LGP model into a recurrent model is to retain register values between sequential pattern presentations. Thus within the context of a prediction problem, the registers are never reset. The prediction is read from the predefined 'output' register(s) [13].

In the case of document categorization and analysis problem, the objective is to predict the class label for each document. Since a document can be represented by an encoded word sequence, the document classification can be treated as a word sequence's label prediction problem. Encoded words of a document are input into RLGP one after another, registers are never reset until the last word of a document is input to RLGP. The prediction of the class label for a document is then read from the predefined 'output' register after the last word is input.

7.3. Dynamic Subset Selection (DSS)

In this work, we expect the RLGP algorithm to work on a large data set, such as Reuters-21578 which could have over 6000 documents. This expectation inspired us to identify a solution to the problem of efficiently training RLGP with a large document set. The basic idea of the approach is based on the concept of the sampling theory where instead of using all of the training data during training, we dynamically select a subset of the training data for a certain duration of the training process [13]. This subset selection algorithm is called DSS. Because of space limitations, details of the algorithm and DSS parameters used in our experiments can be found in [13].

7.4. RLGP for Document Categorization and Analysis

- Inputs and outputs of RLGP

  The number of inputs to the RLGP classifier is the number of values used to represent the words. In this case, a word is represented by a two-dimensional vector, so the number of inputs to RLGP is 2. The output of RLGP at the end of a document is the classification label of the input document. We can track the association between each word and category by tracking the 'output' register value after each input word. Binary classifiers are built up for each category. The RLGP classifier classifies documents to in class (1) or out class (-1). Hence, the output of RLGP is one value corresponding to the label. Thus, the 'output' register value is projected into the data range [-1,1] by using Equation 4:

$$GPoutNew = \frac{2}{1 + e^{-GPout}} - 1 \qquad (4)$$

- Fitness function and parameters of RLGP

  RLGP generates a classification rule by using a set of instructions operating on registers and inputs by using the function set in Table 2. The classification rule is then evaluated against the documents in the training set. Each rule can be tested against any document and will return a value indicating whether the rule is true for that document or not. A classification rule must be evolved for each category C (Figure 1). When evolving a rule for a particular set, the fitness depends on the number of documents classified to the correct category or not. In this work, we simply used the sum square error of all in class and out class documents as the fitness function shown in Equation 5:

$$fitness = \sum_{p=1}^{P} (CorrectLabel(p) - GPout(p))^2 \qquad (5)$$

  where P is the number of total examples in the training set. All the RLGP parameters used in our experiments are summarized in Table 2

8. Experiments and Results

In this work, Reuters 21578 collection is used as the benchmarking data set. There are a total of 12612 news stories in this collection. These stories are in English, where 9603 of them are in the training set, and 3299 are in the test set. We used the top 10 categories of the collection,

Table 2. GP Parameters

Parameter	Value
Selection type	Tournament
Tournament size	4
Functional Set	+, -, ×, ÷
Instruction Type(Ratio)	Constants (0), Internal(4), External(1)
Node Limit	256
Population Size	125
Generations	48000
Number of Register	8
P(Xover)	0.9
P(Mutate)	0.5
P(Swap)	0.9

which are widely used in the area of document analysis and classification. There are two main objectives in our experiments: (i) To evolve effective classifiers to categorize the documents in the data set; (ii) To track word sequences in order to detect the context changes within a document.

8.1. Document categorization

A classification rule for each category is evolved using 20 different RLGP initializations. We selected the best rule as the final categorization rule for each case. An example of a rule evolved by the RLGP classifier for category 'Earn' of the Reuters 21578 data set is shown below:

R1=R1-I1; R0=R0×I1; R1=R1-I1; R0=R0+I1; R1=R1-I1; R0=R0-R1; R0=R0-I0; R1=R1-I1; R0=R0-R1; R0=R0-R1; R0=R0-I0; R0=R0÷I1; R0=R0-I0; R0=R0+I1; R1=R1÷I1;

In the example above, the predefined 'output' register of the classifier is Register0, which gives the label of a document after the last word has been input. 'Input0' and 'Input1' are associated with two dimensions of the word representation. In this work, a label given by 'output' register of RLGP to a document is a value between -1 and 1. Thresholds need to be chosen to separate the in class and out class examples for each category. In these experiments, a threshold (T) is calculated as the median between the median value of in class and out class examples of the training data, shown in Equation 6. If the 'output' register's value of RLGP of a document is greater than the threshold, it is classified to be in class, otherwise it is classified to be out class. For each document, we run it through all classifiers which are built up for each category, so that we can identify multi-labeled documents as well.

$$T = median(median(inClass), median(outClass)) \quad (6)$$

Table 3. Information retrieval measurements

Recall	Precision	F1-Measure
$\frac{TP}{TP+FN}$	$\frac{TP}{TP+FP}$	$\frac{2\times R\times P}{R+P}$

Table 4. Performance on Reuters on four different feature selections

Category	DF	IG	Nouns	MI
Earn	0.90	0.97	0.95	0.98
Acq	0.73	0.64	0.71	0.69
Money	0.48	0.51	0.39	0.38
Grain	0.79	0.79	0.84	0.85
Crude	0.99	0.81	0.84	0.80
Trade	0.72	0.75	0.73	0.74
Interest	0.48	0.60	0.62	0.41
Wheat	0.95	0.91	1.00	0.93
Ship	0.52	0.66	0.43	0.48
Corn	0.61	0.58	0.65	0.35
Macro Ave.	0.72	0.72	0.72	0.66
Micro Ave.	0.79	0.79	0.79	0.78

In information retrieval and text categorization, the classical performance measurements are Recall (R), Precision (P) and F1-measure (F1), table 3, where TP means in class documents, classified to be in class; FN means in class documents, classified to be out class; and FP means out class documents, classified to be in class.

Table 4 shows the classification results (in F1 measure) of the proposed system on all feature selection methods discussed in Section 3. These results show that Mutual Information technique seems to perform worst than the other three techniques when employed by the proposed system on this data set. Moreover, we compared the performance of the proposed system (ProSys) to other systems, tables 5 and 6. It should be noted here that in these comparisons the only common attribute among the different systems is the feature selection method employed, the document representation methods (L-SVM, DT and NB use "bag of words"; T-GP uses n-grams) and classifiers are actually quiet different. Table 5 shows the comparison of categorization performance (F1 measure) of the proposed system with other systems (Tree based GP (T-GP) [7], Linear SVM (L-SVM) [5], Decision Tree (DT) [5] and Naive Bayesian (NB) [5]) using mutual information as the feature selection technique. In this case, the proposed system performs better than the T-GP and NB classifiers but not as good as the DT and L-SVM classifiers. However, it should be noted that based on the results of table 4, the proposed system does not perform that well when mutual information is used. Having said this, it can perform very competitive on some categories such as Earn, Grain and Wheat. However, for some categories, such

Table 5. Comparison: Mutual Information

Category	ProSys	T-GP	L-SVM	DT	NB
Earn	0.98	0.86	0.98	0.95	0.93
Acq	0.69	0.76	0.93	0.85	0.86
Money	0.38	0.61	0.78	0.58	0.55
Grain	0.85	0.55	0.93	0.88	0.72
Crude	0.80	0.83	0.86	0.84	0.62
Trade	0.74	0.76	0.70	0.68	0.43
Interest	0.41	0.57	0.75	0.52	0.51
Wheat	0.93	0.66	0.86	0.90	0.65
Ship	0.48	0.75	0.85	0.73	0.75
Corn	0.35	0.84	0.84	0.92	0.55
Macro Ave.	0.66	0.72	0.85	0.78	0.65
Micro Ave.	0.78	0.77	0.91	0.85	0.79

Table 6. Comparison: Information Gain

Category	ProSys	NB	Rocchio
Earn	0.97	0.97	0.97
Acq	0.64	0.92	0.91
Money	0.51	0.59	0.53
Grain	0.79	0.58	0.58
Crude	0.81	0.71	0.62
Trade	0.75	0.41	0.34
Interest	0.60	0.54	0.48
Wheat	0.91	0.59	0.55
Ship	0.66	0.36	0.36
Corn	0.58	0.31	0.28
Macro Ave.	0.72	0.60	0.56
Micro Ave.	0.79	0.74	0.69

as Money and Interest, it does not work well. Our analysis shows that word co-occurrences in these two categories are heavily overlapped, so the proposed system consistently categorize them into one category. On the other hand, Table 6 shows the comparison of categorization performance (F1 Measure) between the proposed system and Naive Bayesian (NB) [14] and Rocchio [14] classifiers using information gain as the feature selection technique. In this case, the overall performance of the proposed system outperforms the Naive Bayesian (NB) and Rocchio based classifiers on all categories in terms of both Micro and Macro averages.

8.2. Document word tracking

As discussed earlier our objective is to develop a classifier, which can analyze word sequences of documents while tracking the context changes associated with words and categories. Indeed, the proposed classifier can achieve this objective for both single-labeled and multi-labeled documents.

In the case of single-labeled documents, obviously a document is only classified by one classifier (even though it will be input to all the classifiers in parallel). For example, given a single-labeled document from category Earn (in this case, there are 19 words left after Mutual Information is employed), Figure 5 shows the change in the output register of RLGP as words of the document are input to the classifier. When the output register value reduces, it implies that the context is moving towards the out class, when the output register value increases, it implies that the context is moving towards the in class.

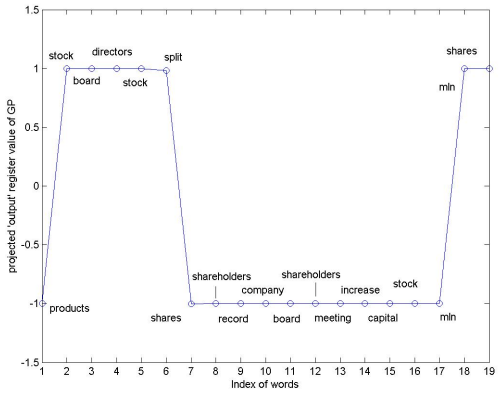

Figure 5. Classification label changes for a single-labeled document

Whereas in the case of a multi-labeled document, it will be classified by multiple classifiers, each classifier will track the words that as they are input to the system. Figure 6 shows a document that belongs to categories Grain, Wheat, and Trade. The underlined words (in different colors) correspond to different classifiers that have the output register value as 1, when that word is input to RLGP. We could see that classifiers can track the context changing. In summary, the proposed system can successfully track the context changes associated with words and categories on both single-labeled and multi-labeled documents. Thus, we believe that such a mechanism can also be successful for tasks such as Topic Detection and Tracking (TDT).

9. Conclusions

We have proposed a system capable of encoding documents into co-occurances/sequences and discovering rules by utilizing these sequences, i.e. employing temporal information. The encoding process is based on a hierarchical SOM architecture, whereas RLGP is used at the stage of document classification. The returned categorization results are competitive with other systems. However, to the best of our knowledge, the proposed system is the first system

<TITLE>TUNISIA BUYS 150,000 TONNES FRENCH WHEAT</TITLE>

<BODY>Tunisia has bought 150,000 tonnes of French soft wheat for August to December shipment at 76.70 dlrs per tonne fob with COFACE export credit, trade sources said.

This is the second French soft wheat export sale for the 1987/88 season which starts on July 1 and follows a Brazilian purchase of 150,000 tonnes.

France sold around 300,000 tonnes of soft wheat to Tunisia during the 1986/87 season.</BODY>

Figure 6. Classification label changes for a multi-labeled document

that directly analyzes word sequences of a document without giving any weight or probability value to an individual word. The rules generated by RLGP is relatively simple and can be easily stored in a database or embedded in programs. Thus, it is useful for document analysis and word tracking associated with categories and also useful for document structure analysis where there can be tags to define the structure of a document such as HTML files. The idea of using a hierarchical SOM architecture to encode temporal information of a document and apply RLGP as a classifier to analyze this temporal information is new for text analysis, the future work will investigate the development of other fitness functions that can incorporate information retrieval measures (such as F1 measure) into the fitness calculation for RLGP. Moreover, different forms of the DSS algorithm where subset is selected based on the nature of a category instead of age and difficulty values will be explored. We believe that the proposed system can make a difference when used for data analysis applications where temporal information is important. Thus, we are going to test the proposed system on topic detection and tracking data sets as the next step.

Acknowledgments

This work was supported by NSERC Discovery and CFI New Opportunities grants of Canada.

References

[1] http://www.cs.dal.ca/ luo/stopwords.txt.

[2] Brill, E. "A simple rule-based part of speech tagge," *Proc. 3rd Conference on Applied Natural Language Processing*, 1992, pp. 152-155.

[3] N. Cancedda, E. Gaussier, C. Goutte, J. Renders, "Word-sequence Kernels," Journal of Machine Learning Research, vol. 3, 2003, pp. 1059-1082.

[4] M. F. Caropreso, S. Matwin, and F. Sebastiani, "A Learner-Independent Evaluation of the Usefulness of Statistical Phrases for Automated Text Categorization," Text Databases and Document Management: Theory and Practice, 2001, pp. 78-102.

[5] S. Dumais, P. N. Bennett, and E. Horvitz, "The combination of text classifiers using reliability indicators", Information Retrieval, 2005, vol. 8, no. 1, pp. 67-100.

[6] S. Dumais, J. Platt, and D. Heckerman, "Inductive Learning Algorithms and Representations for Text Categorization," *Proc. 7th International Conference on Information and Knowledge Management*, 1998, pp. 148-155.

[7] L. Hirsch, M. Saeedi, R. Hirsch, "Evolving Rules for Document Classification," *Proc. 8th European Conference, EuroGP 2005*, Switzerland, 2005, pp. 85-95.

[8] D. D. Lewis, "An Evaluation of Phrasal and Clustered Representations on a Text Categorization Task," *Proc. 15th International ACM Conference on Research and Development in Information Retrieval*, 1992, pp. 37–50.

[9] X. Luo, A. N. Zincir-Heywood, "Analyzing the Temporal Sequences for Text Categorization," *Proc. the 8th International conference on Knowledge-Based Intelligent Information and Engineering Systems*, Springer-Verlag LNAI 3215, 2004, pp. 498-505.

[10] F. Sebastiani, "Machine Learning in Automated Text Categorization," ACM Computing Surveys, vol. 34, no. 1, 2002, pp. 1-47.

[11] Y. Yang and J.O. Pedersen, "A comparative study on feature selection in text categorization," *Proc. ACM Proceedings of the Fourteenth International Conference on Machine Learning*, 1997, pp. 412-420.

[12] S. Wermter, G. Arevian and C. Panchev, "Recurrent Neural Network Learning for Text Routing," *Proc. 9th International Conference on Artificial Neural Networks*, 1999, pp. 470-475.

[13] X. Luo, A. N. Zincir-Heywood, "Evolving Recurrent Linear-GP for Document Classification," *Proc. IEEE Congress on Evolutionary Computation*, 2006, pp. 8605-8612.

[14] H. Wu, T. H. Phang, B. Liu, X. Li, "A Refinement Approach to Handling Model Misfit in Text Categorization", *Proc. 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002, pp. 207-216.