Generating Term Weighting Schemes through Genetic Programming

Ahmad Mazyad, Fabien Teytaud, Cyril Fonlupt

▶ **To cite this version:**

Ahmad Mazyad, Fabien Teytaud, Cyril Fonlupt. Generating Term Weighting Schemes through Genetic Programming. The 4th Annual Conference on machine Learning, Optimization and Data science (LOD), Sep 2018, Tuscany, Italy. pp.92-103. hal-01859657

HAL Id: hal-01859657
https://hal.science/hal-01859657v1

Submitted on 22 Aug 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Generating Term Weighting Schemes through Genetic Programming

Ahmad Mazyad, Fabien Teytaud and Cyril Fonlupt

LISIC, Université du Littoral Côte d'Opale,
50 Rue Ferdinand Buisson, 62100 Calais - France

August 22, 2018

Abstract

Term-Weighting Scheme (TWS) is an important step in text classification. It determines how documents are represented in Vector Space Model (VSM). Even though state-of-the-art TWSs exhibit good behaviors, a large number of new works propose new approaches and new TWSs that improve performances. Furthermore, it is still difficult to tell which TWS is well suited for a specific problem. In this paper, we are interested in automatically generating new TWSs with the help of evolutionary algorithms and especially genetic programming (GP). GP evolves and combines different statistical information and generates a new TWS based on the performance of the learning method. We experience the generated TWSs on three well-known benchmarks. Our study shows that even early generated formulas are quite competitive with the state-of-the-art TWSs and even in some cases outperform them.

1 Introduction

Text Classification (TC) aims to automatically assign a set of predefined categories to a text document based on their content. TC is an important machine learning problem that has been applied to numerous applications such as spam filtering [28], language identification [32], and so on. Generally, the TC approach is to learn an inductive classifier from a set of predefined categories. This approach requires that documents are represented in a suitable format such as the Vector Space Model (VSM) representation (Salton and Buckley, 1988).

In a VSM, a document d_j is represented by a term vector $d_j = (w_{1,j}, w_{2,j}, ..., w_{t,j})$ where each term is associated with a weight $w_{k,j}$.

The weight represents how much a term contributes to the semantics of a document. The method which assigns a weight to a term is called Term Weighting Scheme (TWS).

Numerous TWSs exist and we introduce the most famous in Section 2. They are generated according to human a priori and mathematical rules. TWSs are usually simple mathematical expressions. Unfortunately, depending on the application, it is not easy to know a priori which TWS will be effective.

As expression discovery may naturally be addressed by genetic programming [1], we are interested in this paper to study the effectiveness of Genetic Programming (GP) generated formulas for term-weighting and their aspects. We are also interested to know if a stochastic evolutionary process with no information about the complexity, the shape and the size of the expression can find at least competitive discriminative TWS.

The paper is organized as follows: Section 2 presents the TWSs and related works. In section 3 we present Genetic Programming and how it is applied to TWS. Section 4 presents the experiments and the results, and then we conclude in section 5.

2 Term Weighting Schemes

TC is a supervised learning task. Hence, the training data consists of a set of labeled documents $D = ((d_1, l_1), ..., (d_N, l_N))$, such that d_j is the term vector of j-th document, l_j is its label and N is the total number of training documents. As in VSM representation, a document d_j is represented by a term vector $d_j = (w_{1,j}, w_{2,j}, ..., w_{m,j})$ where $w_{i,j}$ is a weight assigned to the i-th term of the vocubulary t_i of the document d_j and determined by the TWS.

2.1 Statistical Information

Generally, a multi-labeled classification task is turned into several distinct single-label binary task, one for each label, using the binary relevance (BR) transformation strategy. That is, given the list of labels $L = \{l_1, l_2, ..., l_m\}$, the original data set is transformed into m different data sets $D = \{D_1, D_2, ..., D_m\}$. For each data set D_k, documents having the label l_k will be tagged as the positive category c_k, and the rest as the negative category $\overline{c_k}$. Weights are then computed independently for each binary data set.

Based on the BR transformation, given a term t_i and a category c_k, TWS could be expressed using statistical information a, b, c and d obtained from the training data:

- a is the number of documents that contain the term t_i and belong to the positive category c_k.
- b is the number of documents that don't contain t_i and belong to c_k.
- c is the number of documents that contain t_i and don't belong to c_k.
- d is the number of documents that don't contain t_i and don't belong to c_k.

Besides the statistics described above, Table 1 shows different statistical information that could be extracted from the training data.

Table 1: Statistical information (Terminals) used to evolve a TWS.

Label	Description
N	# documents
C	# categories
C_t	# categories that contain the term t
N_t	# doc that contain t
$\overline{N_t}$	# doc that do not contain t
N_{cat}	# doc in the positive category cat
$\overline{N_{cat}}$	# doc that do not belong to cat

Table 2: Six traditional CF factors.

CF	Defined by
χ^2	$\frac{N*(a*d-b*c)*(a*d-b*c)}{(a+c)*(b+d)*(a+b)*(c+d)}$
or	$\log(2+\frac{a*d}{b*c})$
rf	$\log(2+\frac{a}{\max(1,c)})$
icf	$\log(\frac{C}{C_t})$
ig	$(\frac{a}{N}\times\log\frac{a\times N}{(a+b)(a+c)})$
	$+(\frac{c}{N}\times\log\frac{c\times N}{(c+d)(a+c)})$
	$+(\frac{b}{N}\times\log\frac{b\times N}{(a+b)(b+d)})$
	$+(\frac{d}{N}\times\log\frac{d\times N}{(c+d)(b+d)})$

2.2 Term Weighting Schemes

Generally, TWSs combines two of three factors pointed out by Salton et al. in [26] that are believed to improve both recall and precision:

- *Term Frequency (TF) factor*: The TF factor is used to capture the relative importance of terms in a document.
- *Collection Frequency (CF) factor*: Also called term discrimination. The importance of words in a document (TF factor) does not provide enough discrimination ability. A common word like 'The' is frequent in almost all documents, and then it could not separate a group of documents from the remainder of the collection. Hence a discrimination factor is needed to favor those terms that are concentrated in a few documents of the collection. Main known CF factors are presented in Table 2.

TWSs could be divided into two sets depending on whether they make use of available information on document membership (Supervised TWSs) or not (Unsupervised TWSs).

Unsupervised TWSs are generally borrowed from Information Retrieval domain [26] and adopted for TC [22, 7, 23].

Term Frequency-Inverse Document Frequency (TF-IDF) is the most famous term weighting method. This method combines the TF factor and the CF factor and can be formally defined as $w_{i,j} = tf_{i,j} \times \log\frac{N}{N_t}$ where $w_{i,j}$ is the weight of the term t_i in the document d_j, $tf_{i,j} = f_{i,j}$ is the term frequency represented by the raw count of t_i in d_j, and $\log\frac{N}{N_t}$ is the inverse document frequency (idf). Besides the raw count ($f_{t,d}$) representation of tf, there exist numerous other variants such as binary representation ($w_{i,j} = 1$ if the term t_i occurs in the document d_j and 0 otherwise), $log(f_{i,j}) + 1$, $f_{i,j}/\sum_{t'\in d} f_{t',d}$. All these variants

are also used as TWS on their own [26, 22, 7, 8]. The inverse document frequency has also a number of variants such as $\log(N/N_t) + 1$, $\log((N - N_t)/N_t)$ [26].

Supervised TWSs makes use of available information on the membership of training documents by replacing the unsupervised idf component in TF-IDF by another supervised component. Debole et al. and Deng et al. in [8, 7] are the first to take advantage of such information by combining the unsupervised TF component with different supervised term discrimination component: χ^2 (TF-CHI), which makes a test of independence between a term and a category. χ^2 alongside with other supervised feature selection metrics, has been tested in several papers, as a term weighting methods for text categorization. For example, Deng et al. in [8], replaces the idf factor with χ^2 factor, claiming that TF-CHI is more efficient than TF-IDF. In contrast, in a similar test, Debole et al. in [7], compare TF-IDF with three supervised term weightings, namely, TF-CHI, Odds Ratio (TF-OR) and Information Gain (TF-IG). The authors have found no consistent superiority of these new term weighting methods over TF-IDF; Information Gain TF-IG [2] which measures the amount of information obtained for category prediction by knowing the presence or absence of a term in a document [31, 7, 8]; Gain Ratio (TF-GR) first used in a feature selection method defined as the ratio between the information gain of two variables and the entropy of one of them [7]; Odds Ratio (TF-OR) was first used as a feature selection method by Mladeni'c et al. [24]. It is a measure that describes the strength of association between two random variables. A comparative study on term-weighting for TC is made by Deng et al. in [8]. The study shows a good performance of TF-OR but is outperformed by TF-GR; Relevance frequency (TF-RF) proposed in [20], measures the distribution of a term between the positive and the negative category, and favors those terms that are more concentrated in positive category than in negative categories; Inverse Category Frequency (TF-ICF) is a new supervised TWS proposed by Wang et al. in [30]. The measure aims to favor those terms that appear in fewer categories. More similar methods have appeared in [12, 13, 15]. Several comparative studies on these TWSs for both term-weighting and feature selection has been reported in [31, 24, 8, 22]. A new approach for term-weighting based on (TF-IG) have been proposed for multi-labeled classification task in [23]. The method computes a score based on all categories and then subtracts it from the original TF-IG weight. The idea is to take into consideration the weights of terms not only in terms of positive and negative categories but also in terms of every single category. Similar approaches have been proposed to learn TWSs via GP in [4, 6, 5, 29, 25, 11], however, these studies have focused on information retrieval problem. For TC, a similar approach proposed by Escalante et al. in [9]. However our study differs in two ways: first, Escalante et al. try to generate new TWSs by combining existing TWS, and secondly, they learn a single TWS for each data set whereas we learn a TWS for each category in a data set. In our work, we generate TWSs by combining statistical information at a microscopic level to evolve new TWSs. We also extend the study on the thematic TC. We hope this leads into more robust non human based TWSs.

3 Genetic Programming

Evolutionary computing is based on Darwin's theory of "survival of the fittest". The main scheme of evolutionary algorithms is to evolve a population of individuals that are randomly generated. Each individual represents a candidate solution that undergoes a set of genetic operators that allow to mix and alter partial solutions. One of the key features of evolutionary algorithms is that they are stochastic schemes.

3.1 Introduction

GP belongs to the family of evolutionary algorithms. It was first proposed by Cramer [3] and then popularized by Koza [19]. Unlike genetic algorithms where the aim is to discover a solution, the goal of GP is to find out a computer program that is able to solve a problem.

In GP, a set of random expressions that usually represent computer programs are generated. As in all evolutionary computation algorithms, this set of programs will evolve and change dynamically during the evolution. What makes GP suitable for a number of different applications is that these computer programs can represent many different structures, such as mathematical expressions for symbolic regression [27], decision trees [17], programs that control a robot [18, 21] to fulfill a certain task or programs that are able to predict defibrillation success in patients and so on.

The quality of a candidate solution (i.e. a program) is usually assessed by confronting it with a set of fitness cases. This step is usually the most time-consuming step as the programs may get huge and several thousands of candidate programs are usually evaluated at each generation. These computer programs will undergo one or several evolutionary operators that will alter in a hopefully beneficial way. The most classical evolutionary operators are usually the crossover operator that allows the exchange of genetic material (in our case subtrees) and the mutation operator that allows a small alteration to the program.

In the most conventional GP approach, programs are usually depicted by trees. In GP terminology, the set of nodes are split into two sets, inner nodes of the tree are drawn from a set of functions while the terminal nodes (leaves) are drawn from a so-called terminal set. Depending on the problem, the set of functions can be mathematical functions, boolean functions, control flow functions (if,...), or any functions that may be suitable to solve the given problem. The terminal set is usually the set of inputs of the problem, e.g., parameters and constants for symbolic regression problems, sensors for robot planning, etc.

When the stopping criterion is reached, the best individual is returned, otherwise, the loop continues and the best individuals are selected (according to their fitness). There exist numerous ways for selecting the population, the mutation and the crossover operators. This is beyond the scope of this paper and the reader can refer to [19, 16] for more information.

3.2 Evolving Term Weighting Scheme using Genetic Programming

A CF factor is a combination of statistical information. It is intended to measure the discriminative power of a term, i.e. it tells how much a term is related to a certain category. These statistics combined by means of mathematical operators and functions.

We are interested in automatically evolving a CF factor (an individual) using GP. In our approach, the learned CF factor combined to the TF factor forms a term weighting method.

In our context of automatically evolving term weighting methods, an individual is a combination of the function set that is built with simple arithmetical operators (+,-,*,/,log,...) and the terminal set (constant values and inputs to our problem).

Table 1 shows the statistical information used as terminal set for generating formulas which represent CF factors. As it can be seen, the function set is made of very simple arithmetical functions while the terminal set includes to the best of our knowledge all the statistical information used to build a TWS.

As previously mentioned, programs (generated TWS) are depicted as trees. In this problem, the terminal nodes consist of statistical information extracted from training data, while the inner nodes are a set of defined operators that combines the statistical information to form a new TWS.

Table 3: Parameters used in our genetic program.

Parameter	Value
Population Size	100
Initial Individual Size	20
Number of generations	100
Function set	$+$, $-$, $/$, $*$, $\sqrt{x}$, $log1(x)$, $log2(x)$
Terminal set	a, b, c, d, N, N_t, $\overline{N_t}$, N_{cat}, $\overline{N_{cat}}$, C, C_t
Mutation	OnePointMutation ($P = 1/individual\ size$)
CrossOver	SubtreeCrossover ($P = 0.85$)

3.2.1 Terminals and Function Set

In this study, we try to generate new TWS by evolving the CF factor and then combines it with the TF factor. The CF factor is a combination of constants, statistical information (N, N_t,...), and mathematical operators. Hence we define the terminals as the statistical information shown in Table 1. Regarding the mathematical operators, they are defined as one of the following ($+$, $-$, $/$, $*$, $\sqrt{x}$, $log1(x) = \log(1 + x)$ and $log2(x) = \log(2 + x)$).

We should note that the statistical information has different types (single value, vector, and matrix). For instance, the number of documents in the training data N is a constant (single value), the number of documents that contains a term t is a vector containing the number of documents for each term and

finally, the number of documents that belongs to a category *cat* and contains a term t is a matrix. Operations on these different types of statistical information are taken care of by Eigen[1] library using element-wise transformations.

3.2.2 Genetic Operators

In GP, a set of individuals is initialized and then evolved according to a set of genetic operators. At first, we randomly generate a random size individuals with a max size of twenty genes (the max size could be overpassed during the cross-over operation). As for genetic operators, we use the elite selection and re-insertion, a subtree crossover with a probability of 0.85 and one point mutation with a probability of 1/size of the individual.

3.2.3 Fitness Function

Generally, the performance of a TWS is assessed on known benchmarks by evaluating a classification model on VSM representation of this TWS. Numerous evaluation metrics exist that evaluate the classification model such as f_1 measure. Evaluating the classification model is a vital step that affects the performance of the GP. However, it could be very time-consuming. Hence, it is important to choose a good and fast machine learning algorithm. LibLinear [10] is an open source library for large-scale linear classification. It supports linear support vector machines.

In our study, once a new individual is generated, we perform a 3-fold cross-validation on the training data which generates three disjoint subsets. We use two subsets as the training set and one subset as the test set. The process is repeated three times using each time different subset for testing. The performance is measured using the f_1 measure. The average classification performance is used as the fitness function. The f_1 measure considers both precision p (true positive over true positive plus false positive) and recall r (true positive over true positive plus false negative) and can be formally defined as $f_1(p, r) = \frac{2rp}{r+p}$.

4 Experiments and Results

This section presents an empirical evaluation of the proposed approach. The goal of this study is to assess the effectiveness of the generated TWSs and compare their performances to standard TWSs. The souce code of the implentation needed for our experiments could be found in a public repository[2].

4.1 Experimental setup

In our experiments, we have used three widely well-known benchmarks in TC: Reuters-21578 Benchmark Corpus[3], Oshumed Benchmark Corpus[2] and the 4

[1] http://eigen.tuxfamily.org/
[2] https://bitbucket.org/mazyad/eigennlp
[3] http://disi.unitn.it/moschitti/corpora.htm

Universities data set also called Webkb[4]. The Reuters-21578 data set is one of the most used test collection for TC research. We use the well-known "Apte-Mod" split [14]. This version of the data set contains ninety categories, however, in our experiments, we report results only for the largest ten categories. Oshumed dataset is extracted from the Oshumed[1] collection compiled by William Hersh. It includes 13,929 medical abstracts from the MeSH categories of the year 1991. Each document in this data set belongs to one or more categories from 23 cardiovascular diseases categories. Webkb data set contains WWW-pages collected from computer science departments of various universities in January 1997 by the World Wide Knowledge Base (Webkb) project of the CMU text learning group. In this experiment, we kept only the four largest categories ("student", "faculty", "course" and "project"), and we split it into three random folds where two folds are used for the training set and one fold for the test set.

For all three data sets considered in the experiments, a default list of stop words, punctuation and numbers are removed, lower case transformation and Porter's stemming are performed.

Furthermore, for each experiment, a binary transformation is applied. That leads to multiple distinct single-label binary task, one for each label (see Section 2.1). Each task could be treated as an independent experiment with its own data set.

As mentioned above, each data set has been split into training and test subsets. Table 4 shows, for each data set, the number of documents in the training and test subsets, the number of classes, the number of terms, the size of smallest category and the size of the largest category.

Table 4: Statistics on the selected data sets used for our experiments (training/test).

	Reuters	Oshumed	Webkb
number of documents	7769/3019	6286/7643	2803/1396
number of classes	90	23	4
number of terms	26000	30198	7890
size of the smallest category	1/1	65/70	336/168
size of the largest category	2877/1087	1799/2153	1097/544

TWSs are evolved using the training subset (see Section 3.2.3). Finally, the test subset is used to evaluate the performance of the generated TWS. And finally, for each data set, we report the f_1 measure (see Section 3.2.3).

In order to obtain more reliable results, we have performed 20 runs on each task. After having evaluated the generated TWSs, we report the performance average and standard deviation over the 20 runs. In addition, we report the maximum and minimum f_1 score obtained across the 20 runs (for each run, only the last generated TWS is taken into account).

[4]http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/

Tables 5,6 and 7 show the results obtained by the generated TWSs and the best baseline using linearSVM. Table 8 shows the average classification performance of the generated TWSs on the test subset of the training data (Validation) and the performance on the test data (Test). The goal of this experiment is to assess the learning ability and to warn us of eventual overfitting. Table 9 shows the average classification performance of a random learned TWS for a single-label binary task on the complete data set. This is important in order to know whether our GP-Based TWS has good generalization performances.

4.2 Results

First, a fast study of the Tables 5,6 and 7 shows that the best baseline TWS is different for each binary task. Therefore, a multi-labeled task requires different TWSs for each category. Using different TWSs could lead to better results. However, the problem is to recognize the best TWS for a specific task. Finding the TWS by cross-validation does not mandatory return the best TWS.

Regarding Reuters-21578, the generated TWSs and the baseline schemes have similar performances. However, on Oshumed and Webkb data sets, the GP-Based TWSs outperform the best baseline schemes. Reuters-21578 is one of the most studied data-set in TC for TWS making the task of finding better TWS harder. Moreover, it is the most unbalanced data set in the study which makes generalization harder.

Table 5: Classification performance on top 10 categories of Reuters-21578 obtained with the generated TWSs and the best standard TWS. Best results are bolded.

	GP			Best TWS	
Label	f1	Min	Max	f1	TWS
earn	98.34±0.09	98.24	98.54	**98.38**	tf.idf
acq	96.93±0.23	96.55	97.54	**97.10**	tf.idf
money-fx	**79.60**±0.50	78.16	80.45	78.63	tf.idf
grain	**94.25**±0.63	93.10	95.22	93.43	tf.rf
crude	**90.01**±0.81	88.27	90.94	88.24	tf.rf
trade	**79.10**±1.21	77.69	80.18	78.03	tf.rf
interest	75.16±0.50	74.45	76.19	**76.19**	tf.idf
ship	**80.52**±1.54	77.84	82.93	78.95	tf.or
wheat	88.11±1.26	86.12	90.96	**90.20**	tf.chi
corn	92.80±0.27	90.83	93.94	**93.91**	tf.chi
Average	**87.48**±0.70	86.13	88.69	87.30	

From Table 8 , we can see that the performance of generated TWSs on the test subset of the training data during the cross-validation (See Section 3.2.3) is very similar to the performance on the test data. In addition, the standard TWSs have different results. This is interesting as it suggests that there is no overfitting and that further learning can improve the performance.

Table 6: Classification performance on Oshumed data set obtained with the generated TWSs and the best baseline of the standard TWSs.

	GP			Best TWS	
L	f1	Min	Max	f1	TWS
C01	**68.19**±1.00	65.91	70.71	64.36	tf.or
C02	**41.28**±1.20	38.45	43.51	36.38	tf.or
C03	76.54±3.28	72.03	81.21	**78.23**	tf.or
C04	80.06±1.48	77.67	81.72	**80.06**	tf.chi
C05	**59.48**±0.20	59.05	60.59	52.85	tf.or
C06	**73.99**±1.29	71.49	75.76	71.44	tf.or
C07	**41.40**±3.35	34.86	47.45	32.6	tf.or
C08	**63.97**±2.51	59.13	67.69	61.34	tf.or
C09	**53.75**±2.63	50.85	58.43	48.00	tf.or
C10	**57.00**±2.33	51.05	59.53	50.2	tf.rf
C11	**67.78**±1.06	65.52	69.23	66.67	tf.or
C12	**76.72**±1.10	73.52	78.25	72.86	tf.or
	GP			Best TWS	
L	f1	Min	Max	f1	TWS
C13	**66.48**±0.47	64.72	67.92	63.70	tf.or
C14	**80.08**±0.39	79.22	80.55	77.11	tf.idf
C15	**65.98**±0.71	64.16	67.20	61.53	tf.chi
C16	**33.54**±0.89	31.14	35.41	28.00	tf.or
C17	**64.85**±0.90	61.87	66.87	59.24	tf.chi
C18	61.21±1.50	57.50	65.12	**61.22**	tf.or
C19	**41.60**±2.04	38.23	45.01	39.84	tf.or
C20	**71.61**±0.28	70.96	72.07	69.62	tf.or
C21	**65.55**±0.32	64.18	67.56	64.37	tf.chi
C22	**10.31**±0.12	8.33	14.37	4.21	tf.or
C23	**46.77**±0.08	45.59	47.20	46.15	tf.idf
Avg	**59.48**±1.26	56.76	61.89	56.08	

From Table 9, we can see that the average performance (macro-f_1) of the generated TWSs outperforms the best baseline on the three corpora which means that the three learned TWS have good generalization performance.

Finally, compared to the results obtained in [9] on Reuters-21578 and Webkb, we have similar results. Note that, in [9], they used Reuters-10 data set which contains only documents from the top 10 categories of the Reuters-21578 data set, whereas we use Reuters-21578 "ModApte" split which contains documents from 90 categories.

Table 7: Classification performance on Webkb data set obtained with the generated TWSs and the best baseline of the standard TWSs.

	GP			Best TWS	
L	f1	Min	Max	f1	TWS
student	**90.29**±0.5	89.05	90.90	90.11	tf.rf
faculty	**86.62**±0.1	85.69	87.81	86.21	tf.rf
project	**80.82**±0.6	77.48	81.76	80.25	tf.rf
course	**94.47**±0.3	93.86	96.08	93.56	tf.rf
Avg	**88.05**±0.4	86.52	89.14	87.53	

Table 8: Average classification performance for validation phase and test phase.

	Validation	Test
Reuters	**89.15**±0.4	**87.48**±0.7
Oshumed	**59.74**±0.9	**59.48**±1.3
Webkb	**87.74**±0.3	**88.05**±0.4

Table 9: Average classification performance of random TWS learned for a single-label task on its corresponding data set and the best baseline. The selected TWS is randomly chosen between the best generated TWSs for each category.

	GP-Based			Baseline	
Data Set	Prefixed formula	TWS	f_1	f_1	Best
Reuters	$**C*//acN\ log2c\ C$	$C*C*(\frac{a}{c*N}*\log(2+c))$	**86.88**	85.92	tf.rf
Oshumed	$/d/+N_t log2\ C_t a$	$\frac{a}{d*(N_t+log(2+C_t))}$	**60.30**	57.10	tf.chi
Webkb	$log1\ log2\ a$	$\log(1+log(2+a))$	**88.43**	87.53	tf.rf

5 Conclusion

In this paper, we have studied the benefits of using genetic programming for generating term-weighting schemes for text categorization. Unlike previous studies, we generate formulas by combining statistical information at a microscopic level. This kind of generation is new, and we can conclude that :

- Different data sets require different formulas. This means that having a good generic formula is really hard to find.
- Within a corpus, it is even better to use a different formula for each category. The hard task is to find out the best for each one.
- Genetic programming is able to find very good formulas which outperform standard formulas given by experts in the literature.
- Eventually, even if the generated formula is specific to a given category,

results show that the best formula for one category is generic enough to be good (but not best) for other categories.

References

[1] Cazenave, T.: Nested monte-carlo expression discovery. In: ECAI. pp. 1057–1058 (2010)

[2] Cover, T.M., Thomas, J.A.: Elements of information theory. John Wiley and Sons (2012)

[3] Cramer, N.L.: A representation for the adaptive generation of simple sequential programs. In: Proceedings of the First International Conference on Genetic Algorithms. pp. 183–187 (1985)

[4] Cummins, R., O'riordan, C.: Evolving general term-weighting schemes for information retrieval: Tests on larger collections. Artificial Intelligence Review 24(3-4), 277–299 (2005)

[5] Cummins, R., O'Riordan, C.: Evolved term-weighting schemes in information retrieval: an analysis of the solution space. Artificial Intelligence Review 26(1-2), 35–47 (2006)

[6] Cummins, R., O'Riordan, C.: Evolving local and global weighting schemes in information retrieval. Information Retrieval 9(3), 311–330 (2006)

[7] Debole, F., Sebastiani, F.: Supervised term weighting for automated text categorization. In: Text mining and its applications, pp. 81–97. Springer (2004)

[8] Deng, Z.H., Tang, S.W., Yang, D.Q., Li, M.Z.L.Y., Xie, K.Q.: A comparative study on feature weight in text categorization. In: Advanced Web Technologies and Applications, pp. 588–597. Springer (2004)

[9] Escalante, H.J., García-Limón, M.A., Morales-Reyes, A., Graff, M., Montes-y Gómez, M., Morales, E.F., Martínez-Carranza, J.: Term-weighting learning via genetic programming for text classification. Knowledge-Based Systems 83, 176–189 (2015)

[10] Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: A library for large linear classification. Journal of machine learning research 9(Aug), 1871–1874 (2008)

[11] Fan, W., Fox, E.A., Pathak, P., Wu, H.: The effects of fitness functions on genetic programming-based ranking discovery for web search. Journal of the Association for Information Science and Technology 55(7), 628–636 (2004)

[12] Guru, D., Suhil, M.: A novel term class relevance measure for text categorization. Procedia Computer Science 45, 1322 (2015)

[13] Ibrahim, O.A.S., Landa-Silva, D.: Term frequency with average term occurrences for textual information retrieval. Soft Computing 20(8), 30453061 (2016)

[14] Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: European conference on machine learning. pp. 137–142. Springer (1998)

[15] Kadhim, A.I.: Statistical computation and term weighting for feature extraction on twitter. In: 2018 International Conference on Advance of Sustainable Engineering and its Application (ICASEA). p. 109114 (Mar 2018)

[16] Karakus, M.: Function identification for the intrinsic strength and elastic properties of granitic rocks via genetic programming (gp). Computers & geosciences 37(9), 1318–1323 (2011)

[17] Koza, J.R.: Concept formation and decision tree induction using the genetic programming paradigm. In: International Conference on Parallel Problem Solving from Nature. pp. 124–128. Springer (1990)

[18] Koza, J.R.: Genetic Programming II, Automatic Discovery of Reusable Subprograms. MIT Press, Cambridge, MA (1992)

[19] Koza, J.R.: Genetic programming: on the programming of computers by means of natural selection, vol. 1. MIT press (1992)

[20] Lan, M., Tan, C.L., Su, J., Lu, Y.: Supervised and traditional term weighting methods for automatic text categorization. Pattern Analysis and Machine Intelligence, IEEE Transactions on 31(4), 721–735 (2009)

[21] Lewis, M.A., Fagg, A.H., Solidum, A.: Genetic programming approach to the construction of a neural network for control of a walking robot. In: IEEE International Conference on Robotics and Automation. p. 26182623 vol.3 (1992)

[22] Mazyad, A., Teytaud, F., Fonlupt, C.: A comparative study on term weighting schemes for text classification. In: International Workshop on Machine Learning, Optimization, and Big Data. pp. 100–108. Springer (2017)

[23] Mazyad, A., Teytaud, F., Fonlupt, C.: Information gain based term weighting method for multi-label text classification task. In: IntelliSys 2018 (2018)

[24] Mladeni'c, D., Grobelnik, M.: Feature selection for classification based on text hierarchy. In: Text and the Web, Conference on Automated Learning and Discovery CONALD-98. Citeseer (1998)

[25] Oren, N.: Reexamining tf. idf based information retrieval with genetic programming. In: Proceedings of the 2002 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology. pp. 224–234. South African Institute for Computer Scientists and Information Technologists (2002)

[26] Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information processing & management 24(5), 513–523 (1988)

[27] Searson, D.P., Leahy, D.E., Willis, M.J.: Gptips: an open source genetic programming toolbox for multigene symbolic regression. In: Proceedings of the International multiconference of engineers and computer scientists. vol. 1, pp. 77–80. Citeseer (2010)

[28] Tretyakov, K.: Machine learning techniques in spam filtering. In: Data Mining Problem-oriented Seminar, MTAT. vol. 3, pp. 60–79 (2004)

[29] Trotman, A.: Learning to rank. Information Retrieval 8(3), 359–381 (2005)

[30] Wang, D., Zhang, H.: Inverse category frequency based supervised term weighting scheme for text categorization. preprint arXiv:1012.2609v4 (2013)

[31] Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. In: ICML. vol. 97, pp. 412–420 (1997)

[32] Zissman, M.A.: Comparison of four approaches to automatic language identification of telephone speech. IEEE Transactions on speech and audio processing 4(1), 31 (1996)