Learning new Term Weighting Schemes with Genetic Programming

Ahmad Mazyad, Fabien Teytaud, Cyril Fonlupt

Learning new Term Weighting Schemes with Genetic Programming

Ahmad Mazyad Fabien Teytaud Cyril Fonlupt

LISIC, ULCO, Université du Littoral Côte d'Opale

Abstract

Text Classification (or Text Categorization) is a popular machine learning task which consists in assigning categories to documents. Feature weight methods are classic tools that are used in text categorization in order to assign a score to each term of a document based on a mathematical formula. In this paper, we are interested in automatically generating these formulas based on genetic programming. We experiment the generated formulas on three well-known benchmarks and state of the art classifiers.

1 Introduction

Text classification is an important task, with numerous applications, such as web search, document classification, information retrieval [5, 16, 15]. Over time, numerous text classification methods have appeared [6], such as k-nearest neighbor [22], Naïve Bayes [13], decision trees [1], neural networks [14], boosting methods [18] and Support Vector Machines [2].

For text classification a crucial point is to represent text documents in a suitable format recognizable by a classifier. In the Vector Space Model (VSM) representation, a document is represented by a vector of terms and each term is associated with a weight. This weight represents how informative/discriminative the correspondent term is. The method which assigns a weight to a term is called Term Weighting Scheme (TWS).
Numerous TWS exist and the most used are presented in Section 2. They are generated according to human a priori and mathematical rules. TWS 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, we are interested in this paper to study whether it is possible to generate TWS automatically using this programming approach. We are 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 TWS and the main state-of-the-art methods. In section 3 we present Genetic Programming. Section 3.2 presents how we apply genetic programming to TWS. Section 4 presents the experiments and the results, and then we conclude in section 5.

2 Term weighting scheme

First, we present Term Frequency-Inverse Document Frequency ($tf.idf$) which is the most popular TWS method, proposed by Jones in [19]. This method is unsupervised (does not take into account the class of a document) and can be formally defined as :

$$w_{t,d} = tf_{t,d} \times \log \frac{N}{N_t} \ , \tag{1}$$

where, $w_{t,d}$ is the weight of term t in document d, $tf_{t,d}$ is the raw count of t in d, and $\log(N/N_t)$ is the inverse document frequency (idf) where N is the total number of documents in the document set and $N_t = |\{d' \in D | t \in d'\}|$ is the number of documents that contains the term t. Beside the raw count ($f_{t,d}$) representation of tf, there exit numerous other variants such as binary weight, $log(f_{t,d}) + 1$, $f_{t,d} / \sum_{t' \in d} f_{t',d}$. idf has also a number of variants such as $\log(N/N_t) + 1$, $\log((N - N_t)/N_t)$, for instance.

The intuition behind this formula is to have a large weight for terms which are frequent in only a few number of documents (having a term present in all documents is not informative).

In [17], Salton *et al.* pointed out three main considerations for a text retrieval system 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. This is the first term in the previous formula.

- *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. This corresponds to the second part of the previous formula. Main known CF factors are presented in Table 1.

- *Normalization factor*: In text retrieval, all documents are considered equally important, however TF factor will favor large documents over shorter ones, as large documents contains more unique terms and/or greater occurrence values. The normalization factor can eliminate this length effect.

Text classification is a supervised learning task, consequently a document class is known in advance. To compute weights some methods used this information. Such methods are called Supervised Term Weighting (STW). In that context and based on the considerations discussed above, researchers proposed various supervised term weighting to use in feature selection methods namely, χ^2 [7, 4], information gain [7, 4], gain ratio [4], odds ratio [7], relevance frequency [11], and recently inverse category frequency [21].

Table 1: Four traditional CF factors. Given a term t and a category cat, N stands for the total number of documents, a is the number of documents that contain t and belong to cat, c is the number of documents that contain t and do not belong to cat, b is the number of documents that do not contain t and belong to cat and d is the number of documents that do not contain t and do not belong to cat.

CF	Defined by
idf	$\log(\frac{N}{N_t})$
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	$\log(2+\frac{a}{\max(1,c)})$

3 Genetic programming

In this section, we first introduce the Genetic Programming model, and then we present how we adapt the GP model to the TWS.

3.1 Presentation

Evolutionary computing is based on the 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 approaches.

Genetic Programming (GP) belongs to the family of evolutionary algorithms. It was first proposed by Cramer in [3] and then popularized by Koza [10]. Unlike genetic algorithms where the aim is to find out 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 is generated. As in all evolutionary computation algorithms, this set of

programs will evolve and change dynamically during the course of 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, decision trees, programs that control a robot to fulfill a certain task or programs that are able to predict defibrillation success in patients ans 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...

If the stopping criteria is reached, then 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 [10, 9] for more information.

3.2 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, that is, it tells how much a term is related to a certain category. These statistical information are combined by means of mathematical operators and functions.

We are interested in automatically evolving a CF factor (an individual) using GP (the learned CF factor combined to the TF factor form 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). A presentation of the implementation and the parameters used in this study will be presented in Section 4.1.

Tables 2 and 3 show the statistical information used as terminal set, and the operators for generating formulas (the function set) which represent CF factors. As it can be seen, the function set is made of very simple arithmetical functions

Table 2: Statistical information (Terminals) used to evolve a TWS.

Label	Description
N	Total number of documents
C	Number of categories
C_t	Number of categories that contain the term t
N_{cat}	Number of documents in the positive category cat
$\overline{N_{cat}}$	Number of documents that do not belong to cat
N_t	Number of documents that contain t
$\overline{Nt}$	Number of documents that do not contain t
a	Number of documents that contain t and belong to the positive category cat
b	Number of documents that do not contain t and belong to cat
c	Number of documents that contain t and do not belong to cat
d	Number of documents that do not contain t and do not belong to cat

Table 3: Arithmetic operators and functions used to evolve a TWS.

Operator	Description
$+$	Arithmetic addition operator
$-$	Arithmetic subtraction Operator
$*$	Arithmetic multiplication operator
$/$	Arithmetic division operator
$\log 1p(x)$	Natural (base e) logarithm of 1 plus ($\ln(1+x)$)
$\log 2p(x)$	Natural (base e) logarithm of 2 plus ($\ln(2+x)$)

while the terminal set includes to the best of our knowledge all the statistical information used to build a TWS.

4 Experiments and results

In order to validate our approach, we compare the four most well-known traditional weighting methods with GP evolved formulas on the ten most frequent categories of the popular dataset Reuters-21578 using linear Support Vector Machine (SVM) [8].

In all our experiments, we perform a cross-validation process.

4.1 Experimental setup

Reuters-21578 is a multi-labeled dataset and one of the most frequently used test collections for text classification research.

We use the traditional "ModApte" split which contains 90 categories. The dataset is transformed into multiple single-label binary classification tasks using the binary relevance transformation strategy.

The point is that the ten binary tasks corresponding to the ten most frequent categories are used. No preprocessing steps or feature selection are performed on the resulting dataset.

For our experiments, the Liblinear classification library [8] is used with a L2-regularized L2-loss Support Vector classification (dual) solver. The other parameters are set to their default values.

Concerning the GP, we use a population of 100 individuals (candidate solutions, i.e. formulas), and 100 generations (corresponding to the stopping criterion).

The crossover probability is set to 0.85 and the mutation probability is defined as one over the number of terminals and operators.

A validation phase is needed as one cannot learn directly over the test set. Thus, the population of formulas are evaluated by means of stratified 3-fold cross validation over the training dataset.

To assess the performance of STW, we report in the result tables the precision p, the recall r and the standard $F1$ measure.

The precision is the true positive tp over the true positive plus the false positive fp. The recall is defined as the true positive over the true positive plus the false negative fn.

The $F1$ measure is defined as

$$F1(tp, fn, fp) = \frac{2 * tp}{2 * tp + fn + fp} \ .$$

In order to obtain one general score for the multiple tasks, the F1 measure is usually averaged using micro-/macro-averaging methods. Macro-averaging gives equal weight to each category, thus, it gives an insight on the efficiency/effectiveness of a classifier on small categories. In contrast, micro-averaging gives weight depending on the category size (the larger the category, the larger the weight), thus results are dominated by large categories.

The micro-$F1$ ($\mu - F1$) is defined as

$$\mu - F1 = F1(\sum_{l=1}^{C} tp_l, \sum_{l=1}^{C} fp_l, \sum_{l=1}^{C} fn_l) \ .$$

The macro-$F1$ ($m - F1$) is defined as

$$m - F1 = \frac{1}{C} \sum_{l=1}^{C} F1(tp_l, fp_l, fn_l) \ .$$

It is important to understand that the macro measure m gives equal weight to each category. The micro measure μ gives equal weight to each per-document classification decision. The main problem of the F1 measure is that it ignores true negatives and its magnitude is mostly determined by the number of true positives, then large classes dominate small classes in terms of micro measure [12, 20].

4.2 Results

Table 4 shows the results of the four traditional TWS on the ten categories. The CF column is the best of the four traditional TWS used in this paper (the best between idf, rf, or, χ^2). We report the precision, the recall and the $F1$-score for the ten categories experimented.

In comparison, in Table 6 we report the same measures but for formulas generated with GP. For each category, the corresponding formulas generated by GP are presented in Table 5.

First, we can notice that for each category the generated formulas are different. The generated formulas (f_1 to f_{10}) perform better on seven out of ten tasks, and the traditional TWS performed better on three tasks (interest, wheat and corn).

In Table 7 we compare $\mu - F1$ and $m - F1$ for the best validated TWS between idf, rf, or, χ^2 for each category against the generated formulas f_1 to f_{10} based on GP. In terms of micro score (i.e. μ) the performance is slightly better for the generated formula, and for the macro score (i.e. m) we are able to improve the performance by 1%. Even if the results seem close, it is impressive that we are able to have such performances with automatically generated TWS.

In Table 8 we compare the performances of each TWS with f_{11} in terms of micro-/macro-averaged $F1$ scores over all categories. f_{11} is a formula which has been generated by GP by considering all ten categories. The goal is to experiment whether GP is able to generate a formula which is good on all ten datasets. In term of micro-averaged score, the generated f_{11} formula is better than all the other formulas. For the macro-averaged score, results are similar.

Finally, in Table 9 it is interesting to see that five from ten generated formulas (f_4, f_5, f_6, f_8 and f_{10}), which have been generated on one specific category, generalize well on the nine other categories, and are slightly better than the four traditional TWS, (see Table 8).

Table 4: Best traditional scheme for each category in top 10 Reuters-21578 categories evaluated by 3-fold cross validation over the training set.

	CF	Precision	Recall	F1
earn	idf	98.98%	97.79%	98.38%
acq	idf	98.71%	95.55%	97.10%
money-fx	idf	80.23%	77.09%	78.63%
grain	rf	96.43%	90.60%	93.43%
crude	rf	89.19%	87.30%	88.24%
trade	rf	82.08%	74.36%	78.03%
interest	idf	88.00%	67.18%	76.19%
ship	or	95.24%	67.42%	78.95%
wheat	chi	84.15%	97.18%	90.20%
corn	chi	91.53%	96.43%	93.91%

Table 5: Formulas generated by GP. f_{11} is a formula generated on all the ten categories.

	Denoted by	Defined by
earn	f_1	$b * (N_t - \overline{N_c})$
acq	f_2	$C_t - (b/(\log 1p((N_t * d * a * \log 2p(\log 2p(\overline{N_t}))) + b)/d))$
money-fx	f_3	$(N + c) - (\log 1p(N_t) * d)$
grain	f_4	$N_t/(c - C_t)$
crude	f_5	$d * (\overline{N_t} + N + C + C_t - (d * d))$
trade	f_6	$N * (\overline{N_t} - N_t + d)$
interest	f_7	$d - (((\overline{N_c} + C) + ((c * ((\overline{N_c} * N_c) + a))/\overline{N_t}))/\overline{N_t})$
ship	f_8	$d * N_t * (\overline{N_t} + N_c + C_t + b)/c$
wheat	f_9	$\log 2p(\overline{N_c} + ((\overline{N_c} * \overline{N_t})/(\log 2p((b + C)/C_t) * b))$
corn	f_{10}	$(c * (N_c + \overline{N_c}) + \overline{N_t} * a * a)/c$
All	f_{11}	$\log 1p(d/c) + (N_t/(\log 1p(\log 2p(\overline{N_c})) * c))$

Table 6: Formulas generated by GP and evaluated by 3-fold cross validation for each category of the ten biggest categories in Reuters-21578.

	CF	Precision	Recall	F1
earn	f_1	98.70%	98.16%	98.43%
acq	f_2	98.85%	95.83%	97.32%
money-fx	f_3	81.82%	80.45%	81.13%
grain	f_4	94.04%	95.30%	94.67%
crude	f_5	90.22%	87.83%	89.01%
trade	f_6	83.49%	77.78%	80.53%
interest	f_7	87.63%	64.89%	74.56%
ship	f_8	92.00%	77.53%	84.15%
wheat	f_9	86.84%	92.96%	89.80%
corn	f_{10}	86.15%	100.00%	92.56%

Table 7: $\mu - F1$ and $m - F1$ for the best validated TWS between idf, rf, or, χ^2 for each category against the generated formulas f_1 to f_{10} based on GP. Results are really close, especially for the μ score. It is interesting to see that GP is able to compete with *a priori* usual known TWS.

	Best rules	GP
μ-F1	93.19%	93.53%
m-F1	87.31%	88.42%

Table 8: $\mu - F1$ and $m - F1$ for all TWS over all categories against the generated formula f_{11} validated over all categories.

	$\mu - F1$	$m - F1$
idf	92.52%	84.90%
rf	92.56%	85.68%
or	92.54%	85.93%
χ^2	91.59%	87.18%
f_{11}	93.13%	86.55%

Table 9: $\mu - F1$ and $m - F1$ over all categories for the generated formula f_1 to f_{10}. We can note that the generated formula reach good performance even on the other categories, which is important in term of finding a global rule.

	$\mu - F1$	$m - F1$
f_1	91.51%	80.77%
f_2	91.06%	79.40%
f_3	91.77%	83.32%
f_4	92.13%	86.30%
f_5	92.82%	85.59%
f_6	92.68%	85.24%
f_7	92.44%	84.65%
f_8	92.39%	86.96%
f_9	89.47%	75.78%
f_{10}	91.10%	86.82%

5 Conclusion

The aim of this paper is to test the ability of GP in automatically generating efficient TWS.

Experiments are conducted and the generated TWS are compared with four traditional TWS. In the experiments, the generated formulas are able to compete with the traditional schemes and are even slightly better in some cases.

We also find that the generated formulas have a good generalization ability.

Our future work will focus on generating a TWS that can generalize even better over different datasets. That can be done by learning directly over the test data of different datasets using the GP.

References

[1] Apte, C., Damerau, F., Weiss, S., et al.: Text mining with decision rules and decision trees. Citeseer (1998)

[2] Cortes, C., Vapnik, V.: Support-vector networks. Machine learning 20(3), 273–297 (1995)

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

[5] Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American society for information science 41(6), 391 (1990)

[6] 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: Asia-Pacific Web Conference. pp. 588–597. Springer (2004)

[7] 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)

[8] 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)

[9] 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)

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

[11] 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)

[12] Manning, C.D., Raghavan, P., Schütze, H., et al.: Introduction to information retrieval, vol. 1. Cambridge university press Cambridge (2008)

[13] McCallum, A., Nigam, K., et al.: A comparison of event models for naive bayes text classification. In: AAAI-98 workshop on learning for text categorization. vol. 752, pp. 41–48. Citeseer (1998)

[14] Ng, H.T., Goh, W.B., Low, K.L.: Feature selection, perceptron learning, and a usability case study for text categorization. In: ACM SIGIR Forum. vol. 31, pp. 67–73. ACM (1997)

[15] Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using em. Machine learning 39(2-3), 103–134 (2000)

[16] Pang, B., Lee, L., et al.: Opinion mining and sentiment analysis. Foundations and Trends® in Information Retrieval 2(1–2), 1–135 (2008)

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

[18] Schapire, R.E., Singer, Y.: Boostexter: A boosting-based system for text categorization. Machine learning 39, 135–168 (2000)

[19] Sparck Jones, K.: A statistical interpretation of term specificity and its application in retrieval. Journal of documentation 28(1), 11–21 (1972)

[20] Van Asch, V.: Macro-and micro-averaged evaluation measures [[basic draft]] (2013)

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

[22] Yang, Y.: Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 13–22. Springer-Verlag New York, Inc. (1994)