Knowledge-Based Systems 212 (2021) 106597

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Output-based transfer learning in genetic programming for document classification

Wenlong Fu *, Bing Xue, Xiaoying Gao, Mengjie Zhang

School of Engineering and Computer Science, PO Box 600, Wellington 6140, New Zealand

ARTICLE INFO

Article history:
Received 14 May 2020
Received in revised form 18 August 2020
Accepted 3 November 2020
Available online 11 November 2020

Keywords:
Genetic programming
Transfer learning
Feature extraction
Document classification

ABSTRACT

Transfer learning has been studied in document classification for transferring a model trained from a source domain (*SD*) to a relatively similar target domain (*TD*). In feature-based transfer learning techniques, there is an investigation on the features being transferred from *SD* to *TD*. This paper conducts an investigation on an output-based transfer learning system using Genetic Programming (GP) in document classification tasks, which automatically selects features to construct classifiers. The proposed GP system directly generates programs from a set of sparse features and only considers the output change of the evolved programs from *SD* to *TD*. A linear model is then used to combine existing GP programs from *SD* as features to *TD*. Also, new GP programs are mutated from the programs evolved in *SD* to improve the accuracy. Via directly utilizing the evolved GP programs and their mutations, the feature extraction and estimation processes on *TD* are avoided. The results for the experiments demonstrates that the GP programs from *SD* can be effectively used for classifying documents in the relevant *TD*. The results also show that it is easy to train effective classifiers on *TD* when the GP programs are used as features. Furthermore, the proposed linear model, using multiple GP programs from *SD* as its inputs, outperforms single GP programs which are directly obtained from *TD*.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction

Document classification has been addressed by a large number of machine learning algorithms [1–4]. The number of features in a document classification task is often large, and a feature selection method is typically required [5,6]. Some categories in a document classification task, such as category "comp.graphics" and category "comp.windows.x" in [7], are very closely related to each other [3,7,8].

Transfer learning techniques [9–12] have been employed for training classifiers from few categories then pass learned knowledge to other relevantly similar categories [7]. In a set of similar categories, the distribution of the selected features in different categories is often different [7]. Therefore, when a trained model from a source domain is applied to a target domain, the distribution change of the selected features has to be taken into account [13,14].

Genetic programming (GP) has been effectively utilized for feature selection in different applications, e.g., multiple classification [15], cybersecurity [16], and high dimensional data classification [17]. GP evolves programs which automatically include a subset of features. In a question–answer ranking task [18], a smaller set of features is effectively selected by GP than other methods. It reveals that GP has capability of automatically selecting proper features to construct effective text classifiers. This research is motivated by that GP evolves programs to transfer features selected by GP to a single output-based composite feature. Between a source domain and a target domain, it is taken into account the difference in the transferred space only. Thus the features selected from the source domain are not required for the relevant distribution change estimation when these features are employed in the target domain. It is promising to explore how to effectively transfer GP programs to the target domain when these programs come from the source domain. Additionally, output-based transfer learning approaches [19,20] have been proposed to train a classifier on a small dataset. The shared features are transformed from the target domain to the source domain, and the proposed target objective function considers the probability of each predicted category based on the outputs of the classifier in all the training data (including both the source domain and the target domain training data). The results show that the effective of transfer learning can be improved after the output of the trained classifier is considered.

This paper conducts an investigation on an output-based transfer learning system using GP for document classification. In this paper, GP programs are directly evolved from a set of sparse features *without* using feature selection methods on a source

* Corresponding author.
E-mail address: wenlong.fu@vuw.ac.nz (W. Fu).

https://doi.org/10.1016/j.knosys.2020.106597
0950-7051/© 2020 Elsevier B.V. All rights reserved.

domain. The evolved programs from a source domain will be applied to a target domain without considering of the distribution of input features. Instead, after GP programs are evolved in the source domain, they are directly applied to the target domain, and we only consider the change of the outputs of the GP programs. A linear model is proposed to combine a set of these GP programs, and the linear model is optimised based on the training data from the target domain. Furthermore, new programs are directly mutated from these GP programs. The features represented by the mutated programs are used to enrich the data to be used by the linear model on the target domain. The major contributions of this paper are as follows:

- a transfer learning technique is proposed that can effectively and directly transfer GP programs evolved from the source domain to the target domain;
- a method is introduced that can effectively combine GP programs evolved from the source domain and their mutated programs for predicting the test documents in the target domain; and
- the evolved GP program can be explained to some extent in the context of document classification.

After this section, the background of document classification is provided in Section 2. The backgrounds of transfer learning and GP for text classification are given in Section 2 as well. The output-based transfer learning system for document classification is introduced in Section 3. After Section 4 introduces the design of experiments, the results and discussions are presented in Section 5. Section 6 brings conclusions and addresses future research directions.

2. Background

2.1. Document classification

Document classification is the task of discriminating a document as one category or more categories based on the text content in the document. To automatically handle a document classification task, statistical techniques and artificial intelligence approaches have been utilized [5,21–23].

Normally, document classification includes the stages of pre-processing, feature extraction, model training, and predication. Some words, such as “an” and “the”, are removed in the stage of pre-processing. In feature extraction, n-gram techniques [24] are popularly used [5,22,25]. One typical scenario in the n-gram model is the bag-of-words model which takes the frequency of each word as a basic and general feature. Since there are some low frequency words and a document usually includes a large number of different words, the bag-of-words model generally has a large and spare space of its frequency-based features. A subset of features are selected by feature selection techniques before training a model [6,21]. The selected features are employed for building models by machine learning algorithms, e.g., k-Nearest Neighbors(KNN) [26] and Support Vector Machine (SVM) [6,27]. The trained/learned models will be used for document prediction.

In this paper, we only address classification problems using the word frequency as basic features. To consider the similarities among categories, domain adaptation has been proposed [7]. For instance, there are similar sub-categories between “comp.graphics” and “comp.windows.x” in a twenty newsgroup dataset [28]. Both are grouped in “comp” (compute) [7]. If a classifier is learned from categories which are utilized as a source domain, the classifier could be transferred to other similar categories being taken into account target domains [14]. In general, transfer learning techniques transfer knowledge learned from a source domain to target domains [13,14]. There are many ways to share learned knowledge between source domains and target domains [13,29,30]. This paper only addresses the share of feature-based knowledge.

2.2. Transfer learning for document classification

In a transfer learning document classification task, there are two domains, namely the source domain SD and the target domain TD. In SD, the document feature x_{SD} have a marginal probability distribution $P_{x_{SD}}$; In TD, the document features x_{TD} have another marginal probability distribution $P_{x_{TD}}$. After feature selection is conducted in SD, there is a subset of features $x_{sel,SD}$. The subset features $x_{sel,SD}$ of documents x_{SD} in the source domain are used to train a model $M(x_{sel,SD})$. Category c for document x^i_{sd} (i for the current document in SD) with its selected features $x^i_{sel,SD}$ is discriminated by the model $M(x^i_{sel,SD})$.

In order to use the trained model $M(x_{sel,SD})$ in the target domain, we assume that the features x_{sel} selected in the model $M(x_{sel,SD})$ are the same between SD and TD. x_{sel} is used for $x_{sel,SD}$ and $x_{sel,TD}$. For a document x^j_{TD} (j for the current document in TD) in the target domain, x^j_{sel} is used to represent its selected subset features. Generally, marginal probability distributions of x^j_{sel} are different in SD and TD. For x_{sel} (the selected features), SD includes a marginal probability distribution $P_{x_{sel,SD}}$ and TD includes a marginal probability distribution $P_{x_{sel,TD}}$. For a document x^i_{SD} with its estimated probability distribution $p_{x^i_{sel,SD}}$ in SD, its predicated category c^i is obtained by $M(x^i_{sel}, p_{x^i_{sel,SD}})$. After $P_{x_{sel,TD}}$ is employed to transfer the model M trained from SD to TD, for a document x^j_{TD} with its estimated probability distribution $p_{x^j_{sel,TD}}$ in the target domain, its predicated category c^j is obtained by $M(x^j_{sel}, p_{x^j_{sel,TD}})$.

2.3. Related work to GP for text classification

The rule-based GP system and the value-based GP system are typically employed for text classification [31,32].

In the rule-based GP system, the evolved results can be easily understood by humans. Similar to human rule design, GP systems used letters as terminals, and logical operators and string operators as functions [31,33]. An expand function was utilized for n-gram term construction so that an exhaustive search of all solution candidates (combinations of letters) can be avoided. The evolved programs mainly search for n-gram terms existing or not in a document, but ignore the number of occurrences of words. When documents with different categories are complicated and similar, the rule-based GP system might have low test accuracy on these documents. Additionally, prior knowledge on text classification is needed for setting up the rule-based GP system.

In the value-based GP system, existing features for text classification are used, and evolved binary GP programs are used to discriminate documents to predefined categories [32,34]. After a small set of features are selected, all features in the selected set are used as terminals. In a general function set, mathematical operators are usually included. For instance, add, minus, times and division are commonly used in the value-based GP system [35]. A threshold is employed for discriminating a document as a predefined label when the output value of a GP program is larger than the value of the threshold . The value-based GP system [36,37] automatically selected features and constructed programs for predicting labels of documents. Since the term frequency and n-gram terms are used in the value-based GP system, the value-based GP system can outperform the rule-based GP system [32,34,38].

In summary, a value-based GP system is promising for document classifications. However, there are few work on feature extraction on documents using GP, and how to effectively transfer features to a target domain when these features are extracted by GP in a source domain.

3. The proposed output-based transfer learning approach

The proposed GP-based transfer learning system is presented in this section. The main components of the GP system are described in Section 3.1. Section 3.2 introduces use of the output-based GP programs to the transfer learning system. Section 3.3 provides the overall structure of the system.

3.1. GP components

Representation. The tree-based representation is used in the proposed GP system, which is the most promising representation in GP for classification, especially for binary classification [35]. Each evolved tree can be a (complete) classifier, and feature selection and feature construction are automatically and simultaneously achieved during the evolving stage [35,39]. A tree-based GP system includes a given terminal set and a given function set. Each leaf node in a GP program (tree) comes from the terminal set of the tree-based GP; and each internal node comes from the function set. In order to easily handle different types of functions, such as an arithmetic function and a relational function, a strongly typed GP [35] is used in this paper.

Terminal set. Random constants (uniformly generated in a range) and features form the terminal set. Each random constant is in the range of $(-10, 10)$. The occurrence of a word in a document can be considered as a feature in the terminal set. When a predefined vocabulary is provided, the frequency of one word will be zero if the word is not included in the vocabulary. To construct a GP program, the system will automatically select **the frequencies of some words** from the given vocabulary. When the words of the provided vocabulary are the same as the words in all training documents, the proposed GP system is a specific bag-of-words model using the full set of words whose frequencies are used as features. Generally, a given vocabulary does not include useless features, such as comma.

Function set. It is made up of the arithmetic operators $\{+, -, \times, \div\}$ and rational operators {"if", $<$, and $>$}. The operator $\div$ is a protected division which returns value one when the divisor is zero. GP has commonly employed these four arithmetic operators [35]. The relational operators are utilized to construct non-linear features and obtain more flexible search spaces for evolved GP trees.

Classification. Each GP evolved program (tree) can be used to classify documents. Only a single value comes from the output of a GP evolved program for a document. When a threshold θ for GP is given for binary discrimination and the single value for the document is larger than θ, the document is predicated a positive one "1". Otherwise, a negative one "-1" is labeled to the document. Eq. (1) defines that a GP program discriminates a document as the positive/negative category.

$$c(x_{SD}) = \begin{cases} 1 & \text{if } \ Tree_{SD}(x_{SD}) > \theta \\ -1 & \text{otherwise} \end{cases} \tag{1}$$

where x_{SD} means the document from the source domain SD to be predicted, $Tree_{SD}$ stands for the GP program (tree) evolved from SD, $c(x_{SD})$ represents the class label of x_{SD} predicted by $Tree_{SD}$.

Fig. 1 shows a brief tree example of a GP program. Here, the GP program selects "#[meat]", "#[food]", and "#[car]" as terminals. "#[meat]" stands for the number of occurrences of word "meat" in a document, and the same meaning for "#[food]" and "#[car]". If there are two topics food and traffic, the GP program will predict a document to the topic food or the topic traffic. When θ is set to zero and "#[meat]" and "#[food]" both are smaller than "#[car]", the topic traffic will be labeled to the document.

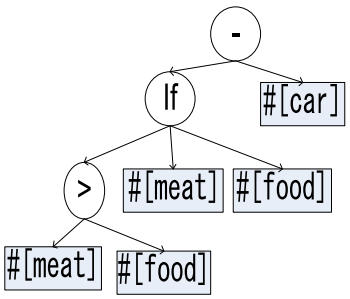

Fig. 1. An Example of GP tree.

Fitness function. The GP evolutionary process is guided by the fitness function, which evaluates how good each candidate tree/solution is. Classification accuracy is used as the fitness function in this work. We mainly focus on binary classification problems and assume the training data is reasonably balanced. The fitness function is shown by Eq. (2):

$$f = \frac{N_{correct}}{N_{total}} \tag{2}$$

where N_{total} is the number of all documents and $N_{correct}$ is the number of correctly predicted documents.

Process of evolving GP programs. The process of evolving GP programs is basically the same as a standard GP algorithm [35]. This process is described below:

(1) An initial population is randomly created by the ramped-half-and-half method. All individuals are measured by the fitness function f (Eq. (2)).
(2) If the termination criterion is met, the best GP program will be returned; otherwise, the population will be evolved by the following operators.
   (i) Reproduction: a small number of the best individuals are directly copied to the next generation.
   (ii) Crossover: two parent individuals are selected by tournament selection, and a sub-tree is randomly selected from each parent individual and two sub-trees are swapped to create two children (new individuals). Crossover is repeated to create a number of child individuals.
   (iii) Mutation: a parent individual is selected and a random sub-tree is replaced by a new randomly generated sub-tree to obtain a new individual. Mutation is repeated to create a number of child individuals
(3) All new individuals are evaluated using the predefined fitness function and repeat the step (2).

The implementation of GP is based on the RMITGP package.[1]

3.2. Output-based transfer learning

Evolved programs are helpful for training classifiers on the target domains [40]. It is assumed that there is a concept shift from the source domain SD to the target domain TD. In this paper, a GP-based transfer learning system is proposed to handle the concept shift from SD to TD. In GP for classification, an evolved tree selects features via leaf nodes, and then uses the selected features to construct a new high-level feature via function nodes. This is essentially the same as mapping a high-dimensional feature

[1] http://titan.csit.rmit.edu.au/e23005/rmitgp/

space to a one-dimensional feature space. For binary document classification, documents of two different classes are expected to be easily separated in the mapped one-dimensional feature space using the threshold θ. Since the GP process does not require any domain knowledge or any assumption of the distributions of features, what features are selected and the distributions of these features are not required in the proposed approach. The key point here is how to handle the concept shift well on the mapped one-dimensional space, which is also the main challenge.

Threshold θ can be any value in the source domain *SD*, since GP will automatically evolve and adjust the GP tree. However, the θ value often needs to be changed when a concept shift occurs from *SD* to *TD*, i.e. the (optimal) θ_{SD} value is often not the optimal θ_{TD} value. Therefore, after obtaining $Tree_{SD}$ from *SD*, an optimal θ_{TD} value needs to be found to adaptively transfer $Tree_{SD}$ to *TD*. This task can be formed as an optimization problem, which is to find the θ_{TD} value, with which $Tree_{SD}$ can minimize the classification error rate in *TD*. This optimisation problem can be formulated as Eq. (4):

$$c(x_{TD}) = \begin{cases} 1 & \text{if } \ Tree_{TD}(x_{TD}) > \theta_{TD} \\ -1 & \text{otherwise} \end{cases} \tag{3}$$

$$\min_{\theta_{TD}} \frac{\sum |c(x_{TD}) - c'(x_{TD})|}{2N_{total,TD}} \tag{4}$$

where $c(x_{TD})$ (Eq. (3)) can be interpreted in the same way as Eq. (1), but to classify a document in *TD*. $c'(x_{TD})$ represents the true class label of document x. $N_{\texttt{total},TD}$ is the number of all documents in *TD*.

Multi-tree output-based transfer learning. A single GP program ($Tree_{SD}$) might not work well on a target domain by cause of the differences (possibly being significant) between *SD* and *TD*. It is possible that a single GP ($Tree_{SD}$) only works well on some of the cases in *TD*, which are similar to the data from *SD* in the transformed/constructed one-dimensional feature space. However, a multi-dimensional constructed feature from multiple GP trees can cover a wider range of data in *TD*. Therefore, multiple GP trees evolved from *SD* are combined for solving the *TD* problem, where the task is to find multiple GP ($Tree_{SD}$) with optimal θ_{TD} values that can cover all or almost all cases from *TD*. If a source domain document and a target domain document have the same category/class label, it is quite likely they share the same set of keywords. If a source domain document x^a_{SD} and a target domain document x^b_{TD} have the same category/class label, the output of feeding x^a_{SD} to a GP tree is likely to be similar to the output of feeding x^b_{TD} to this tree. When the two documents have different categories, the similarity between the outputs of the GP tree is low. Therefore, the proposed algorithm considers that the TF features (i.e. keywords) are shared between the source domain and the target domain when GP trees are constructed during the source domain learning, which can improve the transfer learning performance.

Eq. (5) describes a linear model to combine N evolved GP programs ($Tree_{n,SD}$ where $n = 1, \ldots, N$) using a weight vector $\vec{w}$. Here, w_0 is a constant. In this paper, we only investigate the linear combination for multi-tree output-based transfer learning, for the sake of the simplification and generalization. Also, since GP programs are used as high-level features, there are no feature extraction processes on the target domains. It is assumed that it is easy to use these high-level features to train an effective linear model for document prediction on the target domains.

$$c(x_{TD}) = \begin{cases} 1 & \text{if } \ (w_0 + \sum_{n=1,\ldots,N} w_n * Tree_{n,SD}(x_{TD})) > \theta_{TD} \\ -1 & \text{otherwise} \end{cases} \tag{5}$$

If threshold $\theta_{TD,n}$ is the optimal threshold for $Tree_{n,SD}(x_{TD})$, in Eq. (5), the linear combination of $w_n * \theta_{TD,n}$ will be still a constant. Therefore, we only use single constant w_0 here when we use $\theta_{TD} = 0$. When θ_{TD} is not equal to 0 and is not predefined, we can move w_0 with θ_{TD} together as $\theta_{opt,TD}(=\theta_{TD} - w_0)$. Therefore, the linear model used in this paper is defined in Eq. (6) for binary classification tasks.

$$c(x_{TD}) = \begin{cases} 1 & \text{if } \ (\sum_{n=1,\ldots,N} w_n * Tree_{n,SD}(x_{TD})) > \theta_{opt,TD} \\ -1 & \text{otherwise} \end{cases} \tag{6}$$

To enrich features in the target domains, each selected program $Tree_{n,SD}(x_{TD})$ is randomly mutated on the target domain using one-point mutation. A new mutated program $Tree^m_{n,SD}(x_{TD})$ with the relevant weight w^m_n is added into the linear model. Eq. (9) is employed for document label prediction.

$$sum(x_{TD}) = \sum_{n=1,\ldots,N} w_n * Tree_{n,SD}(x_{TD}) \tag{7}$$

$$sum^m(x_{TD}) = \sum_{n=1,\ldots,N} w^m_n * Tree^m_{n,SD}(x_{TD}) \tag{8}$$

$$c(x_{TD}) = \begin{cases} 1 & \text{if } \ sum(x_{TD}) + sum^m(x_{TD}) > \theta_{opt,TD} \\ -1 & \text{otherwise} \end{cases} \tag{9}$$

In the interest of searching for optimal $\vec{w}$, a hybrid Particle Swarm Optimisation (PSO) with Differential Evolution (DE) technique [41] is employed. The fitness function in the hybrid PSO approach uses the accuracy of the GP programs ($Tree_{n,SD}$) on the training data from the target domain. This hybrid method is employed with the help of its ability of promising global search [41]. It utilized PSO to efficiently search for global solutions and DE to keep population diversity [41]. Via combining PSO and DE to find more solution candidates, the hybrid PSO method efficiently and effectively searched for the global solutions on multiple dimensional linear and non-linear function optimization problems.

3.3. The overall method

Fig. 2 presents the overall process of the GP based transfer learning system. Firstly, GP programs are evolved from the source domain, then are combined by a linear model for solving the target domain problem. Then PSO is used to find the optimal value of the weights as shown by Eq. (5). "$GP_{PSO,N}$" is used to represent the transferred learning system using Eq. (6), where 'N" means to use N GP evolved Programs ($Tree_{n,SD}$ where $n = 1, \ldots, N$) from a source domain; and PSO is applied to a target domain. "$GP^m_{PSO,N}$" is used for using Eq. (9). Therefore, "$GP^m_{PSO,N}$" includes the GP programs from the source domain and their mutations. Note that this paper only works on how to effectively transfer existing GP programs from the source domain to the target domain. On the target domain, GP is not used to evolve programs with a fitness function. Newly mutated programs are directly based on the evolved programs from the source domains. The mutated programs are not evaluated and no further operations from GP operators. Furthermore, to improve test accuracy, predictions from linear models are combined to vote for document labels. In this paper, "$VGP_{PSO,N}$" indicates that the predictions are voted from the results of "$GP_{PSO,N}$", and "$VGP^m_{PSO,N}$" for "$GP^m_{PSO,N}$". GP_{SD} means that the GP programs evolved from the source domain, and GP_{TD} for the target domain.

4. Design of the experiment

4.1. Dataset

The twenty newsgroup dataset [28] has been widely employed by researches [26,27,42]. Some categories in the dataset are very

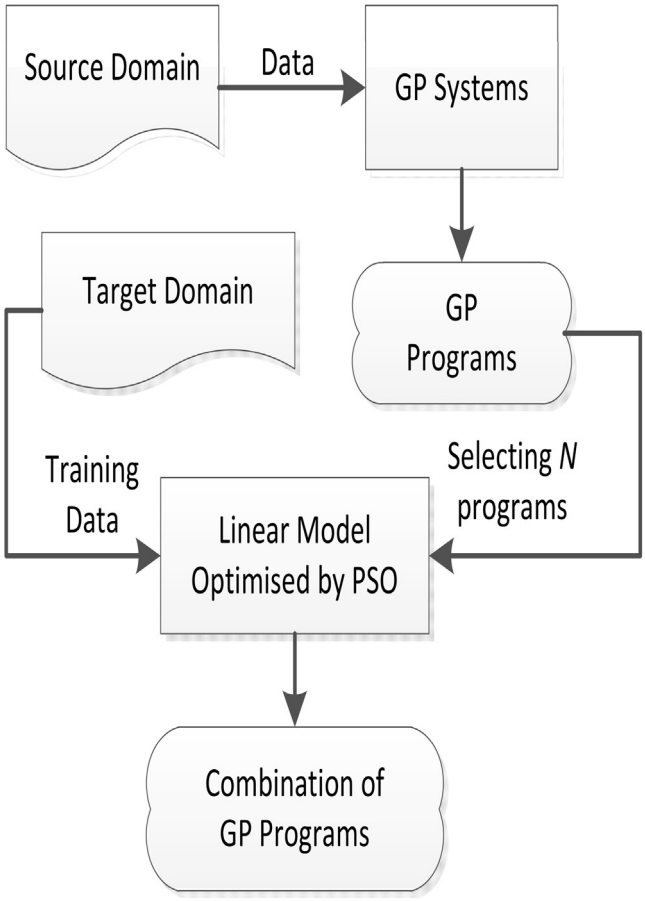

Fig. 2. Overall Processes of the proposed approach ($GP_{PSO,N}$).

closely related to each other, such as category "comp.graphics" and category "comp.windows.x". There are twenty categories, and four groups. Each group is a major category, such as "comp.graphics" and "comp.windows.x" belonging to the category "comp". Following [42], six datasets are generated. Fig. 3 lists the details of the six binary classification tasks. The training data and the test data for the twenty newsgroup dataset are obtained from "20news-bydate". The test dataset is independent of the training dataset. A vocabulary is given in the twenty newsgroup dataset as well. There are 61188 words in the vocabulary. In order to evolve general GP programs for different subcategories, we do not consider the domain differences among the subcategories. This is different from other algorithms considering differences among those subcategories [7,42]. There are more details on training data, test data, and the number of words in [7,42] and the twenty newsgroup dataset website.[2]

4.2. GP settings

Following the common GP parameter settings in [43], we use the probability 0.15 for mutation, the probability 0.80 for crossover, the probability 0.05 for elitism (reproduction), size 200 for population, 200 for the maximum generation, and 8 for the maximum depth (of a program). For N in $GP_{PSO,N}$ and $GP^m_{PSO,N}$, this paper uses 15, 31, 51 and 71 separately. For $VGP_{PSO,N}$ and $VGP^m_{PSO,N}$, there are 15 linear models used to vote for final predictions in each run since 15 programs have been shown good performance for combinations [36].

[2] http://qwone.com/jason/20Newsgroups/

Table 1
Test Accuracy Performances (Mean ± Standard Deviation) on the Target Domain by GP_{SD} and GP_{TD}.

Data	GP_{TD}	GP_{SD}
$data_1$	0.704 ± 0.027 ↑	0.659 ± 0.026
$data_2$	0.735 ± 0.037 ↑	0.683 ± 0.027
$data_3$	0.693 ± 0.035 ↑	0.624 ± 0.039
$data_4$	0.689 ± 0.046 ↑	0.636 ± 0.043
$data_5$	0.715 ± 0.041 ↑	0.634 ± 0.038
$data_6$	0.752 ± 0.033 ↑	0.703 ± 0.031

Table 2
Training Accuracy Performances (Mean ± Standard Deviation) on the Target Domains by $GP_{PSO,N}$.

Data	$GP_{PSO,15}$	$GP_{PSO,31}$	$GP_{PSO,51}$	$GP_{PSO,71}$
$data_1$	0.783 ± 0.021	0.799 ± 0.020	0.810 ± 0.017	$\mathbf{0.814 \pm 0.018}$
$data_2$	0.817 ± 0.015	0.829 ± 0.012	0.833 ± 0.010	$\mathbf{0.834 \pm 0.009}$
$data_3$	0.894 ± 0.018	0.945 ± 0.009	0.967 ± 0.005	$\mathbf{0.976 \pm 0.004}$
$data_4$	0.836 ± 0.043	0.888 ± 0.037	0.921 ± 0.029	$\mathbf{0.940 \pm 0.020}$
$data_5$	0.907 ± 0.016	0.947 ± 0.008	0.966 ± 0.005	$\mathbf{0.975 \pm 0.003}$
$data_6$	0.883 ± 0.026	0.912 ± 0.022	0.926 ± 0.018	$\mathbf{0.934 \pm 0.015}$

There are 100 independent runs for the each experiment in the source domains so that the 100 independent evolved GP programs are randomly selected for optimizing linear models in the target domain. To search for optimal weights w, PSO uses the population size 50 and the maximum generation is 1000. The range of w is from -10 to 10. The other parameters from the PSO hybrid algorithm can be found in [41]. There are 100 independent runs for PSO searching for optimal weights w on the target domains. Since it is time consuming to evolve programs on the target domains, there are only 30 independent runs on GP_{TD}. Please note that it may take around a half hour to evolve a GP program on each experiment. However, it takes far less than 0.1 s to randomly create a mutated GP program from an existing one.

5. Results and discussions

The test results on each dataset are provided in this section. There are discussions of the test results as well.

5.1. Test performances on GP_{SD} and GP_{TD}

Table 1 provides the test performances on the target domains when the evolved GP programs are from GP_{SD} and GP_{TD} separately used for the test target domain documents. Here, t-tests are used for the result comparisons between GP_{SD} and GP_{TD}, using a significance level of 0.05. "↑" means that the results from GP_{TD} are significantly better than the results from GP_{SD}. It is obvious that the GP programs from the target domains are significantly better than the GP programs from the source domains, in terms of the accuracy on the test documents from the target domains. It is interested that the test performances on the GP_{SD} programs are not very low. It is possible that GP automatically selects some words which are shared by similar topics (from a source domain and a target domain separately) and these words are helpful for classifying documents in both domains. This is promising to directly transfer the GP programs evolved from the source domains as features to classify target documents.

5.2. Training accuracies on $GP_{PSO,N}$ and GP^m_{PSO}

Table 2 provides the average training accuracies on the six datasets from $GP_{PSO,N}$. As we can see, when N becomes larger, the training accuracy is higher. It is interesting that the improvement percentages become smaller along with the increase of N. For

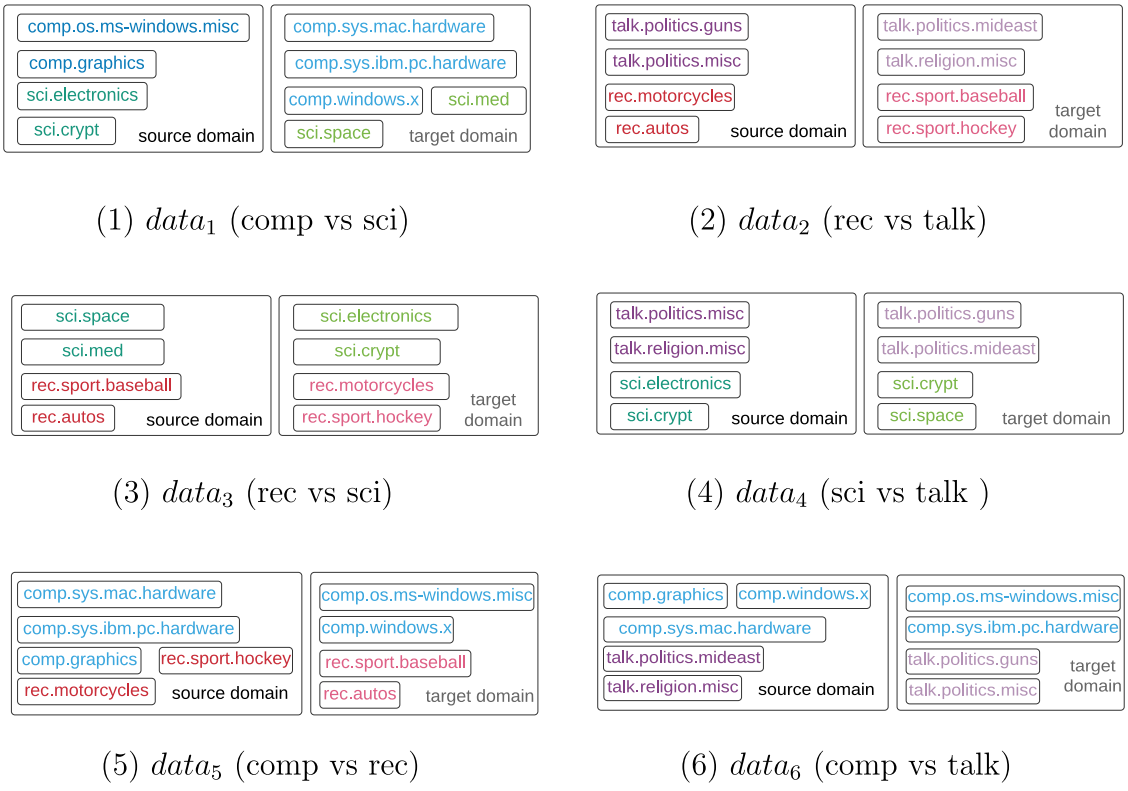

Fig. 3. Six binary classification tasks from the twenty newsgroup.

Table 3
Training Accuracy Performances (Mean ± Standard Deviation) on the Target Domains by $GP^m_{PSO,N}$.

Data	$GP^m_{PSO,15}$	$GP^m_{PSO,31}$	$GP^m_{PSO,51}$	$GP^m_{PSO,71}$
$data_1$	0.853 ± 0.015 ↑	0.889 ± 0.012 ↑	0.911 ± 0.010 ↑	$\mathbf{0.918 \pm 0.008}$ ↑
$data_2$	0.890 ± 0.016 ↑	0.925 ± 0.010 ↑	0.946 ± 0.008 ↑	$\mathbf{0.952 \pm 0.007}$ ↑
$data_3$	0.900 ± 0.017 ↑	0.948 ± 0.009 ↑	0.969 ± 0.005 ↑	$\mathbf{0.978 \pm 0.004}$ ↑
$data_4$	0.862 ± 0.027 ↑	0.910 ± 0.019 ↑	0.938 ± 0.012 ↑	$\mathbf{0.953 \pm 0.009}$ ↑
$data_5$	0.913 ± 0.015 ↑	0.953 ± 0.007 ↑	0.971 ± 0.005 ↑	$\mathbf{0.978 \pm 0.003}$ ↑
$data_6$	0.911 ± 0.018 ↑	0.941 ± 0.011 ↑	0.958 ± 0.008 ↑	$\mathbf{0.963 \pm 0.006}$ ↑

"↑" initiates the results from $GP^m_{PSO,N}$ are significantly better than the results from $GP_{PSO,N}$. t-tests are used for the result comparisons between $GP^m_{PSO,N}$ and $GP_{PSO,N}$, using a significance level of 0.05.

Table 4
Test Accuracy Performances (Mean ± Standard Deviation) on the Target Domain from by $GP_{PSO,N}$.

Data	$GP_{PSO,15}$	$GP_{PSO,31}$	$GP_{PSO,51}$	$GP_{PSO,71}$
$data_1$	0.715 ± 0.024 ↑	0.726 ± 0.022 ↑	0.739 ± 0.020 ↑	$\mathbf{0.742 \pm 0.022}$ ↑
$data_2$	0.755 ± 0.018 ↑	0.767 ± 0.013 ↑	0.773 ± 0.011 ↑	$\mathbf{0.774 \pm 0.009}$ ↑
$data_3$	0.833 ± 0.025 ↑	0.885 ± 0.014 ↑	0.911 ± 0.011 ↑	$\mathbf{0.922 \pm 0.008}$ ↑
$data_4$	0.776 ± 0.040 ↑	0.821 ± 0.036 ↑	0.853 ± 0.029 ↑	$\mathbf{0.873 \pm 0.022}$ ↑
$data_5$	0.850 ± 0.026 ↑	0.889 ± 0.016 ↑	0.911 ± 0.011 ↑	$\mathbf{0.921 \pm 0.008}$ ↑
$data_6$	0.833 ± 0.030 ↑	0.861 ± 0.028 ↑	0.875 ± 0.024 ↑	$\mathbf{0.884 \pm 0.020}$ ↑

"↑" initiates the results from $GP_{PSO,N}$ are significantly better than the results from GP_{TD} utilizing t-tests with a significance level of 0.05.

instance, for $data_4$, there are 4.4% improvement from $N = 15$ to $N = 31$, 3.7% from $N = 31$ to $N = 51$, and 2.0% from $N = 51$ to $N = 71$. Note that the six datasets have the same behavior. It is possible that a small number of the GP programs evolved from the source domains are not rich enough to combine as a linear model for correctly classifying all documents in the target domains. When the number of the GP evolved programs is increasing, the information required for correctly classifying documents in the linear model is more but the relevant performance improvement speed slows down.

To enrich the features used in the proposed linear model, Eq. (9) includes GP programs mutated from the GP evolved programs from the source domains. Table 3 provides the training accuracies of the results from $GP^m_{PSO,N}$ on the target domains. To compare the results between $GP_{PSO,N}$ and $GP^m_{PSO,N}$, **t**-tests are employed with significant level of 0.05. "↑" indicates that the results from $GP^m_{PSO,N}$ with the same settings are significantly better than the results from $GP_{PSO,N}$. It is remarkable that all results from $GP^m_{PSO,N}$ have significantly higher training accuracies than the results from $GP_{PSO,N}$. Therefore, the mutated GP programs effectively enrich the features used in the linear model.

Note that evolving a new GP program on the target domain takes more than half hour, but the time for searching for optimal weights on existing programs by PSO is around one minute. Also, the time for mutating new GP programs can be ignored since there are no evaluations on the new GP programs and no further evolving operations. Although the number of combined GP programs in $GP^m_{PSO,N}$ is the double number of GP programs in $GP_{PSO,N}$, the mutated GP programs come from the GP programs in $GP_{PSO,N}$. There are no GP evolving processes on the target domains.

5.3. Test performances on $GP_{PSO,N}$ and GP^m_{PSO}

Table 4 represents the test accuracies of the results from $GP_{PSO,N}$. All results are compared with the test results from GP_{TD} employing t-tests with a significance level of 0.05. It is obvious that all results from $GP_{PSO,N}$ are significantly better than the results from GP_{TD}. Therefore, via combining the GP evolved programs from the source domains, the proposed linear model can effectively predict the labels of the test documents on the target domains.

Table 5
Test Accuracy Performances (Mean ± Standard Deviation) on the Target Domain by $GP^m_{PSO,N}$.

Data	$GP^m_{PSO,15}$	$GP^m_{PSO,31}$	$GP^m_{PSO,51}$	$GP^m_{PSO,71}$
$data_1$	0.771 ± 0.016 ↑	0.800 ± 0.014 ↑	0.821 ± 0.013 ↑	$\mathbf{0.826 \pm 0.012}$ ↑
$data_2$	0.813 ± 0.022 ↑	0.842 ± 0.017 ↑	0.865 ± 0.014 ↑	$\mathbf{0.871 \pm 0.012}$ ↑
$data_3$	0.836 ± 0.023	0.886 ± 0.014	0.911 ± 0.010	$\mathbf{0.922 \pm 0.008}$
$data_4$	0.798 ± 0.027 ↑	0.837 ± 0.023 ↑	0.869 ± 0.016 ↑	$\mathbf{0.881 \pm 0.014}$ ↑
$data_5$	0.855 ± 0.025	0.896 ± 0.015 ↑	0.920 ± 0.011 ↑	$\mathbf{0.928 \pm 0.009}$ ↑
$data_6$	0.857 ± 0.023 ↑	0.884 ± 0.018 ↑	0.906 ± 0.014 ↑	$\mathbf{0.909 \pm 0.012}$ ↑

"↑" initiates the results from $GP^m_{PSO,N}$ are significantly better than the results from $GP_{PSO,N}$ utilizing t-tests with a significance level of 0.05.

When the results between GP_{SD} and $GP_{PSO,71}$ are compared, the improvement percentages from GP_{SD} to $GP_{PSO,71}$ are 12.6%, 13.3%, 47.8%, 37.3%, 45.3%, and 25.7% one the six datasets, respectively. The improvement percentages on the last four datasets ($data_3$ to $data_6$) are larger than 25%. Therefore, combining the GP programs from the source domains can be very effectively used to improve test accuracy. Comparing the results between GP_{TD} and $GP_{PSO,71}$, the improvement percentages from GP_{TD} to $GP_{PSO,71}$ are 5.4%, 5.3%, 33.2%, 26.7%, 28.8%, and 17.6% one the six datasets, respectively. For datasets $data_1$ and $data_2$, the improvements in $GP_{PSO,N}$ are not large, which requires further work on how to effectively transfer the evolved GP programs from the source domains to the target domains.

Table 5 provides the means and standard deviations of the test accuracies on the target domains from $GP^m_{PSO,N}$. "↑" initiates the results from $GP^m_{PSO,N}$ are significantly better than the results from $GP_{PSO,N}$ utilizing t-tests with a significance level of 0.05, in terms of the test accuracy on each setting. Most of results from $GP^m_{PSO,N}$ are significantly better than the results from $GP_{PSO,N}$. There are three interesting observations. First, there are no significant differences between $GP_{PSO,N}$ and $GP^m_{PSO,N}$ on $data_3$. It is possible that the GP evolved programs from the source domains on $data_3$ include the most useful information in the target domains. Therefore, there are no further performance improvements when the mutated GP programs are used. Second, on $data_5$, there are no significant differences between $GP_{PSO,N}$ and $GP^m_{PSO,N}$ when N is 15 only. When N is larger than 15, the results from $GP^m_{PSO,N}$ are significantly better than the results from $GP_{PSO,N}$ on $data_3$. It is possible that a small set of the mutated GP programs might not be enough to enrich the features in the proposed linear model. Note that each mutated program is randomly driven from the relevant GP evolved program from a source domain. A small set of mutated programs might not be always obtained new useful information on the target domain. There is future investigation on how to add conditions on mutation for performance further improvement. Third, for $data_1$ and $data_2$, there are much more obvious test accuracy improvements, comparing $GP^m_{PSO,N}$ with GP_{TD}. From the comparisons between $GP^m_{PSO,71}$ and GP_{TD}, the improvement percentages on both datasets are 17.3% and 18.5%. Note that the improvement percentages from $GP_{PSO,71}$ against GP_{TD} are 5.4% and 5.3% only. A potential reason for the prominent improvement is that the mutated programs add more helpful information from the datasets on the target domains. Therefore, the mutated GP programs can be effectively used for improving test accuracies when the GP evolved programs are not sufficient for further performance improvement in the proposed linear model.

5.4. Test performances on $VGP_{PSO,N}$ and VGP^m_{PSO}

Table 6 gives the means and standard deviations of the test accuracies on the target domains from $VGP_{PSO,N}$. "↑" initiates the results from $GP_{PSO,N}$ are significantly better than the results from $GP_{PSO,N}$ utilizing t-tests with a significance level of 0.05. From the table, $VGP_{PSO,N}$ significantly improves test accuracies on the six datasets with reference to $GP_{PSO,N}$. However, the improvement percentages on the $data_1$ and $data_2$ are small when $N = 51$ and $N = 71$. It also shows that the GP programs evolved from the source domains may not be sufficient to cover all useful information for classifying documents on the target domains. Note that their standard deviations for $N = 51$ and $N = 71$ are very small. When N is large, there is good stability of correctly classifying documents in $VGP_{PSO,N}$.

Table 6
Test Accuracy Performances (Mean ± Standard Deviation) on the Target Domains by $VGP_{PSO,N}$.

Data	$VGP_{PSO,15}$	$VGP_{PSO,31}$	$VGP_{PSO,51}$	$VGP_{PSO,71}$
$data_1$	0.732 ± 0.014 ↑	0.739 ± 0.010 ↑	$\mathbf{0.753 \pm 0.007}$ ↑	$\mathbf{0.753 \pm 0.006}$ ↑
$data_2$	0.771 ± 0.007 ↑	0.776 ± 0.006 ↑	$\mathbf{0.777 \pm 0.004}$ ↑	$\mathbf{0.777 \pm 0.004}$ ↑
$data_3$	0.920 ± 0.007 ↑	0.934 ± 0.004 ↑	0.942 ± 0.003 ↑	$\mathbf{0.943 \pm 0.002}$ ↑
$data_4$	0.848 ± 0.015 ↑	0.881 ± 0.010 ↑	0.900 ± 0.005 ↑	$\mathbf{0.911 \pm 0.004}$ ↑
$data_5$	0.922 ± 0.006 ↑	0.929 ± 0.005 ↑	0.938 ± 0.003 ↑	$\mathbf{0.939 \pm 0.003}$ ↑
$data_6$	0.872 ± 0.009 ↑	0.889 ± 0.009 ↑	0.897 ± 0.007 ↑	$\mathbf{0.899 \pm 0.006}$ ↑

"↑" initiates the results from $VGP_{PSO,N}$ are significantly better than the results from $GP_{PSO,N}$. T-tests are used for the result comparisons between $VGP_{PSO,N}$ and $GP_{PSO,N}$, using a significance level of 0.05.

Table 7
Test Accuracy (Mean ± Standard Deviation) on the Target Domains by $VGP^m_{PSO,N}$.

Data	$VGP^m_{PSO,15}$	$VGP^m_{PSO,31}$	$VGP^m_{PSO,51}$	$VGP^m_{PSO,71}$
$data_1$	0.822 ± 0.005 ↑	0.834 ± 0.005 ↑	$\mathbf{0.850 \pm 0.005}$ ↑	0.846 ± 0.004 ↑
$data_2$	0.869 ± 0.007 ↑	0.878 ± 0.006 ↑	$\mathbf{0.898 \pm 0.005}$ ↑	0.893 ± 0.005 ↑
$data_3$	0.923 ± 0.007 ↑	0.937 ± 0.004 ↑	$\mathbf{0.946 \pm 0.002}$ ↑	0.945 ± 0.003 ↑
$data_4$	0.874 ± 0.009 ↑	0.890 ± 0.006 ↑	0.910 ± 0.004 ↑	$\mathbf{0.912 \pm 0.003}$ ↑
$data_5$	0.931 ± 0.006 ↑	0.937 ± 0.003 ↑	$\mathbf{0.947 \pm 0.003}$ ↑	0.946 ± 0.003 ↑
$data_6$	0.905 ± 0.007 ↑	0.913 ± 0.005 ↑	$\mathbf{0.930 \pm 0.004}$ ↑	0.925 ± 0.004 ↑

"↑" initiates the results from $VGP^m_{PSO,N}$ are significantly better than the results from $GP^m_{PSO,N}$ utilizing t-tests with a significance level of 0.05.

Table 7 provides the test results from $VGP^m_{PSO,N}$ on the target domains. Also, $VGP^m_{PSO,N}$ significantly improves the test accuracy performances on the six datasets. The improvement percentages from GP_{TD} to $VGP^m_{PSO,51}$ are 20.6%, 22.2%, 36.5%, 32.1%, 32.4%, and 23.7% for the six datasets. $VGP^m_{PSO,N}$ remarkably improves the test accuracies on the six datasets when the GP evolved programs from the source domains are transferred to the target domains. Based on the t-tests between $VGP_{PSO,N}$ and $VGP^m_{PSO,N}$ on each setting, all results from $VGP^m_{PSO,N}$ are significantly better than $VGP_{PSO,N}$. Therefore, it also shows that the mutated GP programs are helpful for classifying documents on the target domains.

Additionally, the results from $VGP^m_{PSO,71}$ have slightly lower test accuracies on the six datasets than $VGP^m_{PSO,51}$, except for $data_4$. It is possible that $GP^m_{PSO,71}$ improves the test accuracies than $GP_{PSO,51}$ and the proposed linear models may have the same predictions on most of documents. The diversity of the multiple predictions from the linear models in $VG^m_{PSO,71}$ is lower than results from $VGP^m_{PSO,51}$. In this paper, $N = 51$ is suggested in $VGP^m_{PSO,N}$.

5.5. Comparing with existing algorithms

We also compare $VGP^m_{PSO,51}$ with other existing transfer learning algorithms [7]. Here we choose SVM using TF (SVM_{TF}), SVM using *Doc2Vec* (SVM_{Vec}) [44], and Transfer Component Analysis (TCA) [7] using TF after Principle Component Analysis (PCA) on TF. SVM_{TF} and TCA are chosen here to compare with $VGP^m_{PSO,51}$ since they outperform several other methods as shown in [44]. In SVM_{TF}, all the TF features in the source domain and the target domain are used to train classifiers for predicting labels of the test documents from the target domain. In TCA, PCA is applied to the TF features, and the 30 top principle components are used

Table 8
Test Accuracy on the Target Domain($GP^m_{PSO,51}$, SVM_{TF}, TCA, and SVM_{Vec}).

Data	$VGP^m_{PSO,51}$	SVM_{TF}	TCA	SVM_{Vec}
$data_1$	0.850 ± 0.005	$0.683 \pm 0.123 \downarrow$	$0.690 \pm 0.239 \downarrow$	$\mathbf{0.911 \pm 0.006} \uparrow$
$data_2$	$\mathbf{0.898 \pm 0.005}$	$0.723 \pm 0.232 \downarrow$	$0.773 \pm 0.159 \downarrow$	$0.890 \pm 0.006 \downarrow$
$data_3$	$\mathbf{0.946 \pm 0.002}$	$0.759 \pm 0.155 \downarrow$	$0.844 \pm 0.023 \downarrow$	$0.887 \pm 0.007 \downarrow$
$data_4$	$\mathbf{0.910 \pm 0.004}$	$0.767 \pm 0.105 \downarrow$	$0.798 \pm 0.042 \downarrow$	$0.855 \pm 0.006 \downarrow$
$data_5$	$\mathbf{0.947 \pm 0.003}$	$0.816 \pm 0.136 \downarrow$	$0.918 \pm 0.024 \downarrow$	$0.933 \pm 0.005 \downarrow$
$data_6$	0.930 ± 0.004	$0.905 \pm 0.070 \downarrow$	$0.922 \pm 0.024 \downarrow$	$\mathbf{0.975 \pm 0.003} \uparrow$

for obtaining a 30 dimensional latent space. The classifiers in TCA are trained from the latent space. More details of SVM_{TF} and TCA can be seen from [44]. More importantly, In SVM_{Vec}, the popular deep learning technique *Doc2Vec* [45–47] is employed to extract features [44], and these extracted features are directly used to train SVM classifiers on the target domain. The dimension of a vector in *Doc2Vec* for a document is 30 and the other parameters are using the default settings in gensim.models.doc2vec.[3] We performed 30 runs of experiments using SVM_{Vec} on each dataset. In each run, a *Doc2Vec* model is trained from the training data and the (30-dimensional) outputs of the trained *Doc2Vec* model are used to train a SVM classifier.

Table 8 provides the test results on the target domain obtained by $VGP^m_{PSO,51}$, SVM_{TF}, TCA and SVM_{Vec}. The statistical significance test, i.e. the ANOVA test with the Holm method for p-value adjustment [48], is used for multiple comparisons between the results of different algorithms. The overall significance level is 0.05. The bold numeric means that the corresponding method is the statistically significantly best one. "↓" indicates that the corresponding result is statistically significantly worse than that of $VGP^m_{PSO,51}$, and "↑" for being significantly better than that of $VGP^m_{PSO,51}$.

From the table, there are four interesting observations. First, $VGP^m_{PSO,51}$ significantly outperforms SVM_{TF} and TCA all on the six datasets. Second, $VGP^m_{PSO,51}$ and SVM_{Vec} have more stable test performance than SVM_{TF} and TCA, shown by the smaller standard deviations of the test accuracy on each dataset. All the standard deviation values from $VGP^m_{PSO,51}$ are smaller than 0.006. However, SVM_{TF} and TCA are not very stable, especially on $data_1$ and $data_2$. Third, $VGP^m_{PSO,51}$ has a stable test performance on all the six datasets, but SVM_{TF} and TCA do not work well on $data_1$. Based on the standard deviations from SVM_{TF} and TCA, it is possible that there is negative transfer occurring. Since $VGP^m_{PSO,51}$ transferred multiple GP programs from source domains to the target domains, the combination of multiple GP programs may have the ability to prevent negative transfer. Last, $VGP^m_{PSO,51}$ outperforms SVM_{Vec} on four datasets (from $data_2$ to $data_5$). In summary, from the comparisons with SVM_{TF}, TCA and SVM_{Vec}, $VGP^m_{PSO,51}$ has the robustness performance on the six datasets. Also, it is shown that GP has the ability to effectively evolve general programs from source domains and the evolved programs can be effectively transferred to the target domains.

Additionally, it is noted that SVM_{Vec} has heavy computational cost on the test document prediction since there is a large number of parameters in the *Doc2Vec* model and it is time-consuming to predict a test document [30]. $VGP^m_{PSO,51}$, SVM_{TF}, and TCA have lower computational cost on prediction. For $VGP^m_{PSO,51}$, only the selected TF features are required for prediction. However, SVM_{TF} and TCA need all the TF features for prediction. On a platform with a single CPU (2.6 GHz), the average time of classifying a document is far less than 0.01 s when $VGP^m_{PSO,51}$, SVM_{TF}, or TCA is used. However, it takes around 0.02 s for one document prediction when SVM_{Vec} is used. It is noted that it takes around half an hour to evolve a single GP program on the source domain. To search for the optimal weights in $GP^m_{PSO,N}$, it takes less than one minute. For SVM_{TF}, TCA and SVM_{Vec}, it takes less than 10 min to get a classifier. Although the proposed approach has the longest training time, the testing time for a document is far less than 0.01 s.

[3] https://radimrehurek.com/gensim/models/doc2vec.html

Table 9
Multiple Comparisons for $GP_{PSO,N}$, $GP^m_{PSO,N}$, $VGP_{PSO,N}$ and $VGP^m_{PSO,N}$.

Data	N	$GP_{PSO,N}$			$GP^m_{PSO,N}$			$VGP_{PSO,N}$			$VGP^m_{PSO,N}$		
		31	51	71	31	51	71	31	51	71	31	51	71
$data_1$	15	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	31		↓	↓		↓	↓		↓	↓		↓	↓
	51			−			↓			−			↑
$data_2$	15	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	31		↓	↓		↓	↓		↓	−		↓	↓
	51			−			↓			−			↑
$data_3$	15	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	31		↓	↓		↓	↓		↓	↓		↓	↓
	51			↓			↓			−			−
$data_4$	15	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	31		↓	↓		↓	↓		↓	↓		↓	↓
	51			↓			↓			↓			↓
$data_5$	15	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	31		↓	↓		↓	↓		↓	↓		↓	↓
	51			↓			↓			−			−
$data_6$	15	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	31		↓	↓		↓	↓		↓	↓		↓	↓
	51			↓			−			↓			↑

5.6. The number of GP programs

Table 9 gives multiple comparisons from $GP_{PSO,N}$ and $GP^m_{PSO,N}$ among different N values based on ANOVA tests with the Holm method for p-value adjustment [48]. The overall significance level is 0.05. "↓" means that the item from the first column is significantly worse than the item from the relevant row, and "↑" for being significantly better and "−" for no significant differences. From the multiple comparisons, when N is increased from 13 to 51, the test performance becomes significantly better. Comparing $N = 51$ and $N = 71$, most of the results from $N = 71$ are significantly better than the results from $N = 51$. Therefore, when more the GP evolved programs are selected into the proposed linear model, the test performances on the target domains becomes better. However, when the number of GP evolved programs is large, it is possible that the test performance may not be further improved.

Table 9 also gives the multiple comparisons from $VGP_{PSO,N}$ and $VGP^m_{PSO,N}$ among different N values. Similar to the results from $GP_{PSO,N}$ and $GP^m_{PSO,N}$, when N is increased from 15 to 51, the test performances becomes significantly better, except for the comparison between $VGP_{PSO,31}$ and $VGP_{PSO,71}$. Different from the results from $GP_{PSO,N}$ and $GP^m_{PSO,N}$, the results from $GP^m_{PSO,51}$ are significantly better than the results from $GP^m_{PSO,71}$ on datasets $data_1$, $data_2$ and $data_6$.

To reveal the comparisons among $GP_{PSO,N}$, $GP^m_{PSO,N}$, $VGP_{PSO,N}$, and $VGP^m_{PSO,N}$, Fig. 4 lists all the results (based on their means and standard deviations) from $GP_{PSO,N}$, $GP^m_{PSO,N}$, $VGP_{PSO,N}$, and $VGP^m_{PSO,N}$. First of all, as can be seen, all of the top results come from $VGP^m_{PSO,N}$. $VGP^m_{PSO,51}$ have the best test performances on the six datasets. Second, all have performance improvements when N is from 15 to 71, except for $VGP^m_{PSO,N}$. Third, the improvement speed of the test accuracies from $GP^m_{PSO,N}$ and $VGP_{PSO,N}$ is slower than the improvement speed from $GP^m_{PSO,N}$ and $GP_{PSO,N}$, except for $data_4$. It shows voting based on a small number of GP programs has larger test accuracy improvements than voting based on a large

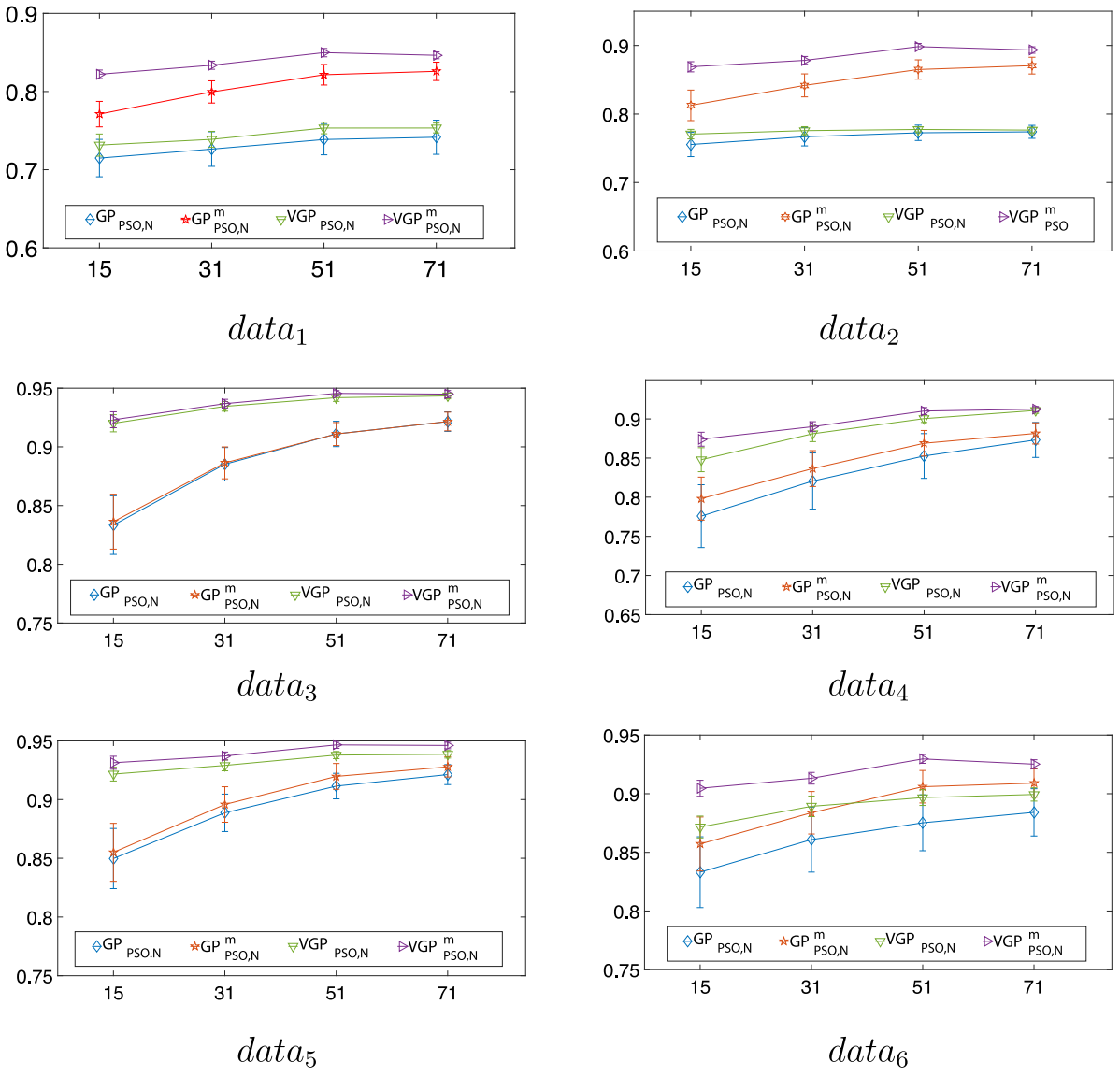

Fig. 4. Test Accuracies From $GP_{PSO,N}$, $GP^{m}_{PSO,N}$, $VGP_{PSO,N}$, and $VGP^{m}_{PSO,N}$.

number of GP programs. It is possible that the diversity level of the predictions from the GP programs becomes low when the number of the GP programs grows.

In summary, a small number of GP programs evolved from the source domains might not be sufficient to cover all test cases from the target domains. To adaptively apply the evolved GP programs from the source domains to the target domains, a proper number of GP programs is suggested here. Based on the comparisons, $N = 51$ is suggested for the six datasets. The proposed linear model can effectively classify the target test documents when the GP programs and their mutated programs are used as features.

5.7. An example evolved GP program

$$Prog_1 = \begin{cases} \#[\text{term}](3 - \#[\text{fifo}]) & \text{if} \frac{\#[\text{what}]}{\#[\text{came}]} > \frac{\#[\text{nor}]-41.9}{\#[\text{being}]} \\ \frac{24.1\#[\text{term}]\#[\text{tomcat}]}{\#[\text{came}](\frac{\#[\text{came}]}{\#[\text{nor}]} - \frac{\#[\text{nor}]}{\#[\text{war}]})} & \text{otherwise} \end{cases} \tag{10}$$

$$Prog_2 = \frac{Prog_1}{\frac{\#[\text{being}]-41.9}{\#[\text{johnson}]} - \#[\text{dpt}]} - \#[\text{employment}] \tag{11}$$

$$Prog_3 = \begin{cases} 1868.8 & \text{if} \#[\text{war}] < 90.7 \\ \#[\text{right}] & \text{otherwise} \end{cases} \tag{12}$$

$$Prog_4 = \#[\text{gun}]\left[\frac{\#[\text{nor}]}{\#[\text{being}]} + \#[\text{asp}] - \#[\text{dozen}]\right] \tag{13}$$

$$Prog_5 = 1.02 - \frac{3.6}{\#[\text{rocket}]} - 73.26\#[\text{term}] - \#[\text{ipser}] \tag{14}$$

$$Prog_6 = \frac{Prog_2}{Prog_5 - Prog_4 \frac{90.7}{Prog_3}} \tag{15}$$

Eq. (15) provides an example evolved GP program from $data_6$ ("comp vs talk"). Eqs. (10)- (14) are combined to Eq. (15). There are several interesting observations. First, very specific words exist in this program, such as "johnson" and "ipser". In the training documents, "johnson" only comes from an Email address and "NASA Johnson Space" in the group "comp"; and "ipser" only comes from a name "Ipser" in the subcategory "talk.politics.misc". Second, general words exist in Eq. (10), such as "term", "what", "came", "nor", and "being". If we only look these words, they can be almost used for anywhere. However, the number of each

word in each group is different. Searching "being" in the two training groups, we can get 149 in group "comp" and 405 in group "talk". It is possible that the combination of different occurrences of words in documents can be helpful to classify documents. Third, a general word exist only for a single group, such as "employment". From the training datasets, "employment" only exists in the group "talk". Fourth, keywords are found for group "talk", such as "war" and "gun". In group "talk", "war" and "gun" have high frequencies existing in documents. Words "fifo" and "rocket" are specific in group "comp". Lastly, some words from the provided dictionary do not exist in the training datasets, such as "tomcat". Note that GP selects words from a dictionary, rather than the limited documents.

From the five interested observations, it is found that the GP program evolved from the source domains can be helpful for classifying documents in the target domains. However, helpful information existing in the source domains might not work well in the target domains. Specific words ("johnson" and "ipser") might not be able to be filtered in GP evolved programs. It is necessary to combine multiple GP evolved programs for reducing the influence from these specific words.

5.8. Discussions

From the analysis of the evolved GP program (Eq. (15)), it shows that the program includes the shared words from the source domain and the target domain. A set of the shared words may be used to describe similar content. A potential reason is that the GP programs evolved from the source domains have a proper ability to classify the documents on the target domains. Since the evolved GP programs include a set of selected words (existing in both domains) and when the GP programs are used as features, it is easy to train a classifier for correctly classifying test documents, which is a reason to use the linear model rather than a complex classifier.

When programs are evolved from source domains and only a few training documents (with or without labels) are provided, it is suggested to avoid influence of specific words from the source domains only. One simple way is to only consider the words in both the training documents and the target domains. We will develop new methods to effectively evolve programs from source domains with a few examples from target domains. Furthermore, we will investigate tasks of varying difficulty, such as the various similarity between source domains and target domains. When each word is considered as a vector [44,45,49,50], it is worth investigating how to use GP to effectively extract common features when a source domain and a target domain are considered in the evolving stage.

6. Conclusions

This paper investigated output-based transfer learning on GP programs for document classification. After GP programs were evolved in the source domains, a linear model was proposed to combine these GP programs for classifying documents on the target domains. From the experiments, the evolved GP classifiers from source domains have been shown to be helpful to classify documents from target domains. The combinations of randomly selected GP programs from a source domain in the proposed linear model was used to effectively classify the test documents on the target domains. The proposed linear model remarkably outperformed the GP programs directly trained from a target domain, in terms of the test accuracy on the target domain. Also, mutated GP programs were used to significantly improve the test accuracies on the target domains. When the number of the selected GP programs is increasing, the test performance is improved. However, after the number reaches to a certain value, the performance improvement is not less obvious. When multiple results from linear models are voted for a final prediction, the test performances may slightly go down when the number of the selected GP programs is too larger. An example evolved GP program was analyzed, and it is suggested that general words are helpful for classifying documents in different domains when each word frequency is used as a basic feature. In future work, we will investigate how to effectively and automatically evolve GP programs using general words in multiple domains. Also, we will investigate how to effectively combine GP with deep learning techniques for performance improvement on document classification tasks.

CRediT authorship contribution statement

Wenlong Fu: Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing - original draft, Writing - review & editing. **Bing Xue:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing - original draft, Writing - review & editing. **Xiaoying Gao:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing - original draft, Writing - review & editing. **Mengjie Zhang:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing - original draft, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

All authors approved the version of the manuscript to be published.

Funding

No funding was received for this work.

Intellectual property

We confirm that we have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing of publication, with respect to intellectual property. In so doing we confirm that we have followed the regulations of our institutions concerning intellectual property.

References

[1] P. Li, K. Mao, Y. Xu, Q. Li, J. Zhang, Bag-of-concepts representation for document classification based on automatic knowledge acquisition from probabilistic knowledge base, Knowl.-Based Syst. 193 (2020) 105436.
[2] F. Gargiulo, S. Silvestri, M. Ciampi, G.D. Pietro, Deep neural network for hierarchical extreme multi-label text classification, Appl. Soft Comput. 79 (2019) 125–138.
[3] W. Hadi, Q.A. Al-Radaideh, S. Alhawari, Integrating associative rule-based classification with Naïve Bayes for text classification, Appl. Soft Comput. 69 (2018) 344–356.
[4] R.H. Pinheiro, G.D. Cavalcanti, I.R. Tsang, Combining binary classifiers in different dichotomy spaces for text categorization, Appl. Soft Comput. 76 (2019) 564–574.
[5] M.M. Mirończuk, J. Protasiewicz, A recent overview of the state-of-the-art elements of text classification, Expert Syst. Appl. 106 (2018) 36–54.

[6] G. Feng, B. An, F. Yang, H. Wang, L. Zhang, Relevance popularity: A term event model based feature selection scheme for text classification, PLoS One 12 (4) (2017) e0174341.
[7] S.J. Pan, I.W. Tsang, J.T. Kwok, Q. Yang, Domain adaptation via transfer component analysis, IEEE Trans. Neural Netw. 22 (2) (2011) 199–210.
[8] Y. Zhai, Y.S. Ong, I.W. Tsang, Making trillion correlations feasible in feature grouping and selection, IEEE Trans. Pattern Anal. Mach. Intell. 38 (12) (2016) 2472–2486.
[9] F.H. Khan, U. Qamar, S. Bashir, Enhanced cross-domain sentiment classification utilizing a multi-source transfer learning approach, Soft Comput. 23 (14) (2019) 5431–5442.
[10] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, G. Zhang, Transfer learning using computational intelligence: A survey, Knowl.-Based Syst. 80 (2015) 14–23.
[11] R.K. Sanodiya, J. Mathew, A framework for semi-supervised metric transfer learning on manifolds, Knowl.-Based Syst. 176 (2019) 1–14.
[12] S.M. Salaken, A. Khosravi, T. Nguyen, S. Nahavandi, Extreme learning machine based transfer learning algorithms: A survey, Neurocomputing 267 (2017) 516–524.
[13] S. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1345–1359.
[14] K. Weiss, T.M. Khoshgoftaar, D. Wang, A survey of transfer learning, J. Big Data 3 (1) (2016) 9.
[15] K. Nag, N.R. Pal, Feature extraction and selection for parsimonious classifiers with multiobjective genetic programming, IEEE Trans. Evol. Comput. (2019) 1.
[16] H. Al-Sahaf, I. Welch, A genetic programming approach to feature selection and construction for ransomware, phishing and spam detection, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2019, pp. 332–333.
[17] B. Tran, B. Xue, M. Zhang, Genetic programming for multiple-feature construction on high-dimensional classification, Pattern Recognit. 93 (2019) 404–417.
[18] U. Bhowan, D.J. McCloskey, Genetic programming for feature selection and question-answer ranking in IBM Watson, in: Proceedings of the 8th European Conference on Genetic Programming, EuroGP 2015, 2015, pp. 153–166.
[19] G. Wang, G. Zhang, K. Choi, K. Lam, J. Lu, An output-based knowledge transfer approach and its application in bladder cancer prediction, in: 2017 International Joint Conference on Neural Networks, IJCNN, 2017, pp. 356–363.
[20] G. Wang, G. Zhang, K.-S. Choi, K.-M. Lam, J. Lu, Output based transfer learning with least squares support vector machine and its application in bladder cancer prognosis, Neurocomputing 387 (2020) 279–292.
[21] G. Kou, P. Yang, Y. Peng, F. Xiao, Y. Chen, F.E. Alsaadi, Evaluation of feature selection methods for text classification with small datasets using multiple criteria decision-making methods, Appl. Soft Comput. 86 (2020) 105836.
[22] B. Altınel, M.C. Ganiz, Semantic text classification: A survey of past and recent advances, Inf. Process. Manage. 54 (6) (2018) 1129–1153.
[23] W. Zhang, Y. Li, S. Wang, Learning document representation via topic-enhanced LSTM model, Knowl.-Based Syst. 174 (2019) 194–204.
[24] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, L. Chanona-Hernández, Syntactic N-grams as machine learning features for natural language processing, Expert Syst. Appl. 41 (3) (2014) 853–860.
[25] A. Dey, M. Jenamani, J.J. Thakkar, Senti-N-Gram: An n-gram lexicon for sentiment analysis, Expert Syst. Appl. 103 (2018) 92–105.
[26] T. Dogan, A.K. Uysal, Improved inverse gravity moment term weighting for text classification, Expert Syst. Appl. 130 (2019) 45–59.
[27] B. Škrlj, M. Martinc, J. Kralj, N. Lavra, S. Pollak, Tax2vec: Constructing interpretable features from taxonomies for short text classification, Comput. Speech Lang. (2020) 101104.
[28] K. Lang, NewsWeeder: Learning to filter netnews, in: Proceedings of the 12th International Machine Learning Conference, ML95, 1995.
[29] W.M. Kouw, L.J. van der Maaten, J.H. Krijthe, M. Loog, Feature-level domain adaptation, J. Mach. Learn. Res. 17 (171) (2016) 1–32.
[30] N. Reimers, I. Gurevych, Sentence-BERT: Sentence embeddings using siamese BERT-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2019.
[31] L. Hirsch, M. Saeedi, R. Hirsch, Evolving text classification rules with genetic programming, Appl. Artif. Intell. 19 (7) (2005) 659–676.
[32] H.J. Escalante, M.A. García-Limón, A. Morales-Reyes, M. Graff, M. Montes-y-Gómez, E.F. Morales, J. Martínez-Carranza, Term-weighting learning via genetic programming for text classification, Knowl.-Based Syst. 83 (2015) 176–189.
[33] L. Hirsch, M. Saeedi, R. Hirsch, Evolving rules for document classification, in: M. Keijzer, A. Tettamanzi, P. Collet, J. van Hemert, M. Tomassini (Eds.), Proceedings of the 8th European Conference on Genetic Programming, EuroGP 2005, 2005, pp. 85–95.
[34] B. Zhang, W. Fan, Y. Chen, E.A. Fox, M.A. Gonçalves, M. Cristo, P. Calado, A genetic programming approach for combining structural and citation-based evidence for text classification in web digital libraries, in: E. Herrera-Viedma, G. Pasi, F. Crestani (Eds.), Soft Computing in Web Information Retrieval: Models and Applications, 2006, pp. 65–83.
[35] A. Agapitos, R. Loughran, M. Nicolau, S. Lucas, M. O'Neill, A. Brabazon, A survey of statistical machine learning elements in genetic programming, IEEE Trans. Evol. Comput. 23 (6) (2019) 1029–1048.
[36] W. Fu, B. Xue, M. Zhang, X. Gao, Transductive transfer learning in genetic programming for document classification, in: Simulated Evolution and Learning, 2017, pp. 556–568.
[37] W. Fu, B. Xue, M. Zhang, X. Gao, Genetic programming based transfer learning for document classification with self-taught and ensemble learning, in: Proceeding of the 2019 IEEE Congress on Evolutionary Computation, CEC, 2019, pp. 1–10.
[38] I. Khodadi, M.S. Abadeh, Genetic programming-based feature learning for question answering, Inf. Process. Manage. 52 (2) (2016) 340–357.
[39] J. Ma, X. Gao, A filter-based feature construction and feature selection approach for classification using genetic programming, Knowl.-Based Syst. 196 (2020) 105806.
[40] M. Iqbal, B. Xue, H. Al-Sahaf, M. Zhang, Cross-domain reuse of extracted knowledge in genetic programming for image classification, IEEE Trans. Evol. Comput. 21 (4) (2017) 569–587.
[41] W. Fu, M. Johnston, M. Zhang, Hybrid particle swarm optimisation algorithms based on differential evolution and local search, in: Proceedings of the 23rd Australasian Joint Conference on Advances in Artificial Intelligence, Vol. 6464, 2011, pp. 313–322.
[42] W. Dai, G.-R. Xue, Q. Yang, Y. Yu, Co-clustering based classification for out-of-domain documents, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '07, 2007, pp. 210–219.
[43] W. Fu, M. Johnston, M. Zhang, Distribution-based invariant feature construction using genetic programming for edge detection, Soft Comput. 19 (8) (2015) 2371–2389.
[44] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: Proceedings of International Conference on Learning Representations, ICLR '13, 2013.
[45] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: The 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, 2019, pp. 4171–4186.
[46] Z. Yang, Z. Dai, Y. Yang, J.G. Carbonell, R. Salakhutdinov, Q.V. Le, XLNet: Generalized autoregressive pretraining for language understanding, 2019, CoRR, abs/1906.08237, URL arXiv:1906.08237.
[47] J. Kim, S. Jang, E. Park, S. Choi, Text classification using capsules, Neurocomputing 376 (2020) 214–221.
[48] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. 6 (2) (1979) 65–70.
[49] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of tricks for efficient text classification, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, Association for Computational Linguistics, 2017, pp. 427–431.
[50] M.A. Mouriño-García, R. Pérez-Rodríguez, L. Anido-Rifón, M. Vilares-Ferro, Wikipedia-based hybrid document representation for textual news classification, Soft Comput. 22 (18) (2018) 6047–6065.