Semantic Genetic Programming for Sentiment Analysis

Mario Graff, Eric S. Tellez, Hugo Jair Escalante and Sabino Miranda-Jiménez

Abstract Sentiment analysis is one of the most important tasks in text mining. This field has a high impact for government and private companies to support major decision-making policies. Even though Genetic Programming (GP) has been widely used to solve real world problems, GP is seldom used to tackle this trendy problem. This contribution starts rectifying this research gap by proposing a novel GP system, namely, Root Genetic Programming, and extending our previous genetic operators based on projections on the phenotype space. The results show that these systems are able to tackle this problem being competitive with other state-of-the-art classifiers, and, also, give insight to approach large scale problems represented on high dimensional spaces.

Keywords Semantic crossover · Sentiment analysis · Genetic programming · Text mining

M. Graff (✉) · E.S. Tellez · S. Miranda-Jiménez
CONACYT INFOTEC Centro de Investigación e Innovación en Tecnologías de la Información y Comunicación, Aguascalientes, Mexico
e-mail: mario.graff@infotec.mx

E.S. Tellez
e-mail: eric.tellez@infotec.mx

S. Miranda-Jiménez
e-mail: sabino.miranda@infotec.mx

H.J. Escalante
Computer Science Department, Instituto Nacional de Astrofísica, Óptica y Electrónica, Cholula, Mexico
e-mail: hugojair@inaoep.mx

O. Schütze et al. (eds.), *NEO 2015*, Studies in Computational Intelligence 663, DOI 10.1007/978-3-319-44003-3_2

1 Introduction

In recent years, the production of textual documents in social media has increased exponentially. For instance, during 2014, around 300,000 tweets were generated every minute,[1] and 2.5 million pieces of content in Facebook. This ever-growing amount of available information promotes research and business activities around opinion mining and sentiment analysis areas. In social media, people share comments about many disparate topics. i.e., events, movie reviews, sports, and organization, among others. The main result is that social media has now become a gold mine of human opinions. This is perhaps the reason that social media has received a lot of attention from companies and governments.

Automatic sentiment analysis of texts is one of the most important tasks in text mining. The goal is to determine whether a particular document has a positive, negative or neutral opinion on a given topic. There exist other variations considering intermediate (gradual) levels for sentiments. Determining whether a text document has a positive or negative opinion is becoming an essential tool for both public and private companies [15, 23]. This tool is useful to know "What people think" about anything; so, it represents a major support for decision-making processes [21].

The sentiment analysis task has been traditionally faced as a classification problem, where supervised learning methods have been mostly used (e.g., Support Vector Machines). Although, this solution has achieved competitive results, there is still room for improvement. In search for a most effective solution to the sentiment analysis problem, this chapter proposes the usage of evolutionary algorithms. Specifically, we consider semantic genetic programming as modeling framework.

Genetic Programming (GP) is an evolutionary algorithm that has received a lot of attention due to its success in solving hard real-world problems [25]. GP has been known for obtaining human-competitive results, actually, GP has outperformed the solutions found by humans in a number of problems. For instance, since 2004, there has been a competition called "Hummies" held at the *Genetic and Evolutionary Computation Conference (GECCO)* where GP systems have been awarded 7 gold medals, 2 silver and 1 bronze from 2004 to 2014. Only on 2011 GP did not obtained any award. Nonetheless, a variant of GP Cartesian GP obtained the silver medal. Even with this proved effectiveness, to the best of our knowledge, GP has almost not been used to tackle the problem of sentiment analysis, being [1], the exception. In fact, the use of GP in the field of text processing is rather scarce, being one of these exceptions our own previous work, see [6]. In the previous work GP was used to optimize the weighting schemes of a vector space model for text classification; in addition, the work of [16] proposed a GP for evolving features with the goal of reducing the dimensionality of data; finally, there are some works in automatic text summarization with GP [31, 33].

Sentiment analysis poses a number of challenges where GP might be a feasible option. Some of these problems come from its high-dimensional representation and the considerable training set size. In order to give an idea of the well-known curse

[1] http://aci.info/2014/07/12/the-data-explosion-in-2014-minute-by-minute-infographic/.

of dimensionality, a typical real-world dataset for text-mining is represented using tens to hundred of thousands coordinates and tens of thousands examples.[2] However, most of the components of these vectors are zero. Unfortunately, the most popular GP systems (ECJ [36], GPLAB [32], TinyGP [24], among others) do not use sparse representation making them unfeasible to tackle problems with these characteristics given memory restrictions.

Some papers in the GP literature are dedicated to solving problems with a high-dimensional representation and a considerable training size. [38] uses an ensemble of GP created on a problem with 300,000 exemplars on 41 dimensions. In [12–14] a symbolic regression problem is tackled where there are 1,000,000 points with 20 dimensions. In [11] it is proposed to learn a multiplexor of 135 bits this represents a training size of 2^{135}; nonetheless, the fitness function only uses 1,000,000 training cases. On the other hand, looking at problems having a high-dimensional representation in [17] a novel symbolic regression method is proposed on a problem with 340 dimensions and 600 training cases. In [5] different classifiers were co-evolve on problems having from 649 to 102,660 features on 7,000 exemplars. This review shows that the use of GP on problems with high-dimensional representation and considerable training size are scarce, it might be possible that one of the limitations is the time required to find an acceptable solution on GP; this restriction has been mentioned previously by [7].

However, the so-called semantic GP that uses novel semantic operators seem a feasible alternative to tackle text mining problems. This is due to their fast convergence rate and the efficient implementations; being able to evaluate a new individual in $O(n)$ where n is the size of the training set. Among the different semantic operators (see [35] for a recent survey of semantic methods in GP) the ones that seem to have the highest convergence rate were proposed by [4, 9]. Both techniques were inspired by the geometric semantic crossover proposed by Moraglio et al. [18] with the implementation of Vanneschi et al. [3, 34]. The key idea for these new approaches consists in creating the best offspring that can be produced by a linear combination of the parents.

In this contribution, it is proposed an extension of our previous work based on projection in the phenotype space (PrGP) [9]. In addition, a novel GP system named Root Genetic Programming (RGP) is introduced. In RGP all the genetic interchange is performed at the root node. These two systems are tested on a sentiment analysis benchmark obtaining excellent results. That is, the performance of these systems are competitive with the performance of state-of-the-art classifiers. Furthermore, given the characteristics of the benchmark (i.e., the problem has a high-dimensional representation and a considerable dataset) this is an indication that the novel semantic operators are a feasible approach to tackle these kind of problems.

The rest of this chapter is organized as follows. Section 2 presents the procedure used to transform a short text into a vector representation. Section 3 describes the extension of PrGP and our novel proposal, namely Root Genetic Programming

[2] The interested reader in how a document collection is processed to obtain a vector representation is referenced to the specialized literature [2, 28].

(RGP). The parameters settings and the description of the benchmark used are presented on Sect. 4. Finally, the conclusions and some directions for future work are described on Sect. 6.

2 Text Representation

Natural language processing (NLP) is a broad and complex area of knowledge having many ways to represent an input text [8, 27]. In this contribution, we select the widely used vector representation of a text given the simplicity and the proven utility of the model. Figure 1 depicts the procedure used to transform a text input into a vector. There are three main blocks, the first one normalizes the text, then the normalized text is transformed into a vector using Term Frequency Inverse Document Frequency TFIDF and Latent Semantic Index (LSI), more about these blocks below.

In the vector representation each word, in the collection, is associated with a coordinate in a high dimensional space. The numeric value of each coordinate is sometimes called the *weight* of the word. The precise weighting scheme is tightly dependent on the underlying task; we use TFIDF [2] as weighting scheme. The formulae $\mathsf{tf} \times \mathsf{idf}$ puts the name to the technique. Here, tf means for term's frequency, and idf means for the inverse document frequency. More precisely, let $D = \{D_1, D_2, \ldots, D_N\}$ be the set of all documents in the corpus, and f^i_w be the frequency of the word w in document D_i. tf^i_w is defined as the normalized frequency of w in D_i

$$\mathsf{tf}^i_w = \frac{f^i_w}{\max_{u \in D_i}\{f^i_u\}}.$$

In some way, tf describes the importance of w, locally in D_i. On the other hand, idf gives a global measure of the importance of w;

$$\mathsf{idf}_w = \log \frac{N}{\left|\{D_i \mid f^i_w > 0\}\right|};$$

in some sense, idf resembles the definition of self-information in Information Theory [29].

The final product, $\mathsf{tf} \times \mathsf{idf}$, tries to find a balance between the local and the global importance of a term. It is common to use variants of tf and idf instead of the original ones, depending in the application domain [27]. Let v_i be the vector of D_i, a weighted

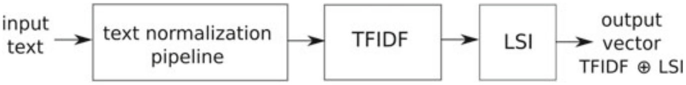


Fig. 1 Generic treatment of input text to obtain input vectors used in our algorithms

matrix TFIDF of the collection D is created concatenating all individual vectors, in some consistent order. Using this representation, a number of machine learning methods can be applied; however, the plain transformation of text to TFIDF poses some problems. On the one hand, all documents will contain common terms having a small semantic charge, e.g., articles and determiners. These terms are known as *stop words*. The bad effects of stop words are controlled by TFIDF, but most of them can be directly removed since they are fixed for a given language. On the other hand, after removing stop words, TFIDF will produce a very high dimensional vector space, $O(N)$ in Twitter, since new terms are commonly introduced (e.g. misspellings, urls, hashtags). This will rapidly yield to the *Curse of Dimension* (CoD); which makes hard to learn from examples since any two random vectors will be orthogonal with high probability. From a more practical point of view, a high dimensionality will also have huge memory requirements, at the point of being impossible to train a typical implementation of a machine learning algorithm (not being designed to use sparse vectors).

One solution is to find which terms have the same meaning and then group them to reduce the dimension. These terms can be synonyms, misspellings, composed terms, hashtags, etc. The task can be effectively performed by a semantic identifier method, like *Latent Semantic Indexing* (LSI). From a general perspective, the context where each term occurs is used to determine its semantic similarity. LSI starts with a weighted matrix, like that created by TFIDF, and then factorizes the matrix through *Singular Values Decomposition* (SVD). SVD finds an orthonormal basis of dimension δ that will be used to describe documents as if they were words. A proper study of LSI is beyond the scope of this document; however, the interested reader is encouraged to review the specialized literature about the technique [2]. It is important to mention that depending on the desired δ, the LSI transformation will lose some of the precision that we can found in the original space; this is a non-desirable behavior. In order to avoid this problem, we kept both the original and the transformed spaces, combined through the *direct sum*, i.e., the document D_i is represented by the concatenation of TFIDF and its LSI transformation. Using this approach we correlate documents that use synonyms (or even common typos) of the same words.

In the following subsections we describe a set of NLP techniques designed to fight the source of variations and errors of the words; after applying these procedures, the documents will be transformed into a cleaner input to compute both TFIDF and LSI models. Even when most of the NLP techniques have a counterpart in many languages, the proper implementation of them are highly dependent on the particular targeted language; in our case study, *Spanish* language. The interested reader looking for solutions in a particular idiom is encouraged to follow the relevant linguistic literature for its objective language, in addition to the general literature in NLP [27].

2.1 Text Normalization

Since Twitter messages (tweets) are full of slang and misspellings, we normalize tweets in order to transform messages into standard language for better text representation. In our case, normalization is language dependent, i.e., Spanish language. The normalization pipeline consists in four steps: error correction, part of speech (POS) tagging, negation, and filtering content words.

Error Correction Step. Language used in Twitter is very informal, with slang, misspellings, new words, creative spelling, URLs, specific abbreviations, hashtags, which are especial words for tagging in Twitter messages, and emoticons, which are short strings and symbols that express different emotions. These problems are treated to prepare and standardize tweets for the POS-tagging stage. All words in each tweet are checked to be a valid Spanish word, or are reduced according to valid rules for Spanish word formation. In general, words or tokens with invalid duplicated vowels or consonants are reduced to valid or standard Spanish words, e.g., (*ruiiidoooo* → *ruido* (noise); *jajajaaa* → *jaja*; *jijijji* → *jaja*). We used an approach based on Spanish dictionary, statistical model for common double letters, and heuristic rules for common interjections. In general, the duplicated vowels or consonants are removed from the target word; the resulting word is looked up in a Spanish dictionary (ca. 550,000 entries) to be validated. For words that are not in the dictionary are reduced at least with valid rules for Spanish word formation. In addition, colloquial words and abbreviations are transformed using a regular expression based on a dictionary of those sort of words. Figure 2 illustrates the procedure. Due to its nature, such dictionary should evolve on large scale living systems to capture the dynamic of the language. Twitter tags such as user names and hashtags (topics) are handled as special tags in our representation to keep the structure of the sentence, and URLs are removed. In the case of emotions, we classify the 500 most popular emoticons into four classes (P = Positive, N = Negative, NEU = Neutral, NONE = none), which are replaced by a polarity word in the text, e.g., positive emoticons are replace by the word positivo (positive), negative emoticons are replaced by the word negativo (negative), neutral emotions are replaced by the word neutro (neutral), and emoticons that do not represent a polarity are discarded. Table 1 shows an excerpt of the dictionary that maps emoticons to their corresponding polarity class.

For instance, in Fig. 4a, in 'Original text', the bold tokens refer to errors that need to be fixed. Those tokens are transformed into a standardized form, as we can see after text 'Error Correction step', Fig. 4a. The resulting text, in this step, is the input for the POS-tagging stage.

tqm → *te quiero mucho* (I love you so much),
compu → *computadora* (computer).

Fig. 2 Expansion of colloquial words and abbreviations

Table 1 An excerpt of the mapping table from Emoticons to its polarity words

:)	:D	:P	→	positivo
:(	:-(	:'(	→	negativo
:-|	U_U	-.-	→	neutro
💻 ✏	▶ 🕘	🍕	→	∅

POS-Tagging Step. At this point, all words are tagged and lemmatized using the POS tagger of Freeling for Spanish language [20]. In Fig. 4b, we can see the parsed text after applying the POS tagger. Each token has the word, left side of the slash symbol, and its lexical information, the right side.

Figure 4b shows the output of our example, for instance, the token `orgulloso/AQ0MS0` (proud) stands for adjective as part of speech (AQ), masculine gender (M), and singular number (S); the token `querer/VMIP1S0` (to want) stands for main verb as part of speech (VM), indicative mood (I), present time (P), singular form of the first person (1S); `positivo_tag/NTE0000` stands for noun tag as part of speech, and so on.

Negation Step. Spanish negation markers might change the polarity of the message. Thus, we attached the negation clue to the nearest word, similar to the approaches used in [30]. We designed 50 regular expression rules for common Spanish negation structures that involve negation markers, namely, *no* (not), *nunca, jamás* (never), and *sin* (without). The rules are processed in order and when one of them matches, the remaining rules are discarded.

A rule consists in two parts: the left side of the arrow represents the text to be matched, and the right side of the arrow is the structure to be replaced. All rules are based on a linguistic motivation taking into account lexical information from the POS-Tagging step.

For example, in the sentence *"El coche no es ni bonito ni espacioso"* ("The car is neither nice nor spacious"), the negation marker *no* is attached to its two adjectives *no_bonito* (not nice) and *no_espacioso* (not spacious); as it is showed in Pattern 1, the negation marker is attached to group 3 (\3) and group 4 (\4) that stand for adjective position because of the coordinating conjunction *ni*. The number of group is identified by parenthesis in the rule from left to right. Negation markers are attached to content words (nouns, verbs, adjectives, and some adverbs), e.g., *'no seguir'* (*do not follow*) is replaced by *'no_seguir'*, *'no es bueno'* (*it is not good*) is replaced by *'es no_bueno'*, *'sin comida'* (*without food*) is replaced by *'no_comida'*. Figure 3 illustrate the negation stem with two examples.

Filtering Step. In the last step of the normalization pipeline, all words are filtered and the lexical information is removed. Content words are the filtered words that we used as features in the following processes. The words are filtered based on heuristic rules that take into account the lexical information shown in Fig. 4b. Figure 4c shows the bag of words resulting of the example parsed above.

— Pattern 1: *el coche no es ni bonito ni espacioso* (the car is neither nice nor spacious)

```
(no/RN)\s+(ser/VS\w+)\s+ni/CC\s+(\w+/AQ\w+)\s+ni/CC\s+(\w+/AQ\w+) → \2 no_\3 y/CC
no_\4
```

— Pattern 2: *no es (de) madera* (X is not made of wood)

```
(no/RN)\s+(ser/VS\w+)\s+(\w+/S\w+\s+)?(\w+/N[^TP]\w+) → \2 \3 no_\4
```

Fig. 3 A set of rules are applied to transform the text giving special emphasis to negation words and its negated concepts

(a)

Original text:

@username èl siempre estará contigo, muy orgulloso de tiiiii y del graaaaannn ser humano que eres :) ... Tqm!!! Buen jueves.

(@username he will always be with you, so proud of you and great human being that you are :) ... ILY!!!! good Thursday.)

After Error Correction step:

user_tag él siempre estará contigo muy orgulloso de ti y del gran ser humano que eres positivo_tag te quiero mucho Buen jueves

(user_tag he will always be with you, so proud of you and great human being that you are positive_tag I love you good Thursday.)

Error correction step

(b)

```
user_tag/NT00000 él/PP3MS000 siempre/RG estar/VAIF3S0 contigo/PP2CS000
muy/RG orgulloso/AQ0MS0 de/SPS00 ti/PP2CS000 y/CC de/SPS00 el/DA0MS0
gran/AQ0CS0 ser/NCMS000 humano/AQ0MS0 que/PR0CN000 ser/VSIP2S0
positivo_tag/NTE0000 te/PP2CS000 querer/VMIP1S0 mucho/DI0MS0 bueno/AQ0MS0
jueves/NCMN000
```

The output of a Spanish sentence parsed with Freeling

(c)

@username siempre orgulloso gran humano positivo querer bueno jueves
(@username always proud great human positive want good thursday)

After the filtering step only content words are preserved

Fig. 4 A step-by-step transformation of a tweet. The negation step is not considered in favor of Fig. 3

Finally, all diacritic and punctuation symbols are also removed.

Q-Gram Expansion. In addition to the exposed NLP transformations, we also appended one more technique to our pipeline. After applying the mentioned NLP machinery, we expanded the resulting text to q-grams. A *character-based q-grams*, or simply q-grams, is an agnostic language transformation that consists in representing a document by all its substring of length q. Please notice that our q-grams differ from the traditional *word-based* q-grams used in traditional NLP that represents a document by the set of all consecutive q words in the text. For example, let $T =$ `abra_cadabra`, its 3-g set are

$$Q_3^T = \{\texttt{abr, bra, ra\_, a\_c, \_ca, aca, cad, ada, dab}\},$$

so, given text of size n we obtain a set with at most $n - q + 1$ elements. Notice that this transformation handle white-spaces as part of the text. Since there will be q-grams connecting words, in some sense, applying q-grams to the entire text can capture part of the syntactic information in the sentence. The main idea behind using character-based q-grams is to tackle misspelled sentences from another perspective, independently of the mentioned ones. The technique is borrowed from the approximate pattern matching literature [19], where it is used for efficient searching and matching text with some degree of error.

A more elaborated example shows why the q-gram transformation is more robust to variations of the text. Let $T = \texttt{I\_like\_vanilla}$ and $T' = \texttt{I\_lik3\_vanila}$, clearly, both texts are different and a plain algorithm will simply associate a low similarity between both texts. However, after extracting its 3-g, the resulting objects are more similar:

$Q_3^T = \{\texttt{I\_l, \_li, lik, ike, ke\_, e\_v, \_va, van, ani,nil, ill, lla}\}$
$Q_3^{T'} = \{\texttt{I\_l, \_li, lik, ik3, k3\_, 3\_v, \_va, van, ani, nil, ila}\}$

Using the Jaccard's coefficient to compare these sets we observe the following similarity values:

$$\frac{|Q_3^T \cap Q_3^{T'}|}{|Q_3^T \cup Q_3^{T'}|} = 0.448.$$

These sets are more similar than the ones resulting from the original text split as words

$$\frac{|\{\texttt{I, like, vanilla}\} \cap \{\texttt{I, lik3, vanila}\}|}{|\{\texttt{I, like, vanilla}\} \cup \{\texttt{I, lik3, vanila}\}|} = 0.2$$

The hope is that a machine learning algorithm knowing how to classify T will do a better job classifying T' using q- grams than a plain representation. This fact is used to create a robust method against misspelled words and other deliberated modifications to the text. We have found that using $q = 5$ is the better value for our problem.

3 Root Genetic Programming

So far, we have described the procedure used to transform a text into a vector. Now, it is time to present the extension performed to PrGP and our novel RGP. Let us start by describing the framework where these systems are developed. PrGP as well as RGP are supervised learning algorithms that learn the instances of a training set $\mathbb{T}$ formed by $n \in \mathbb{N}$ pairs of inputs and outputs, i.e., $\mathbb{T} = \{(x_i, y_i) | i = 1 \ldots n\}$ where

x_i[3] represents the i-th input, and y_i is the associated output. The objective is to find a function f such that $\forall_{(x,y)\in\mathbb{T}}\ f(x) = y$ and that could be evaluated in any element x of the input space. In general, it is not possible to find a function f that learns perfectly $\mathbb{T}$, consequently, one tries to find a function f that minimize an error function, e.g., sum of squared errors $\sum_{(x,y)\in\mathbb{T}}((y - f(x))^2$.

Let us consider a fixed order in $\mathbb{T}$ to define $\mathbf{t} = (y_1, \ldots, y_n) \in \mathbb{R}^n$, namely the target vector, which contains all the outputs in $\mathbb{T}$. Let $s(p, x)$ be a function that evaluates the individual p on input x. Using the order in $\mathbb{T}$, it is possible to define $\mathbf{p} = (s(p, x_1), \ldots, s(p, x_n))$ that contains the evaluation of individual p in all the inputs x of the training set. In this scenario the fitness (using as fitness function the sum of squared error) of individual p can be computed as the square of the euclidean norm $||\mathbf{t} - \mathbf{p}||^2$; the euclidean norm is enough since the induced order is not modified.

PrGP is based on the ideas of the geometric semantic crossover proposed by Moraglio et al. [18] and the implementation developed in [3, 34]. The geometric semantic crossover is defined as follows: Let p_1 and p_2 be the first and second parent the offspring produce by these parent is $o = p_1 r + p_2(1 - r)$, where r is a random function or a constant in the range [0, 1]. The output of individual o at input x is computed as $s(o, x) = s(p_1, x)s(r, x) + s(p_2, x)(1 - s(r, x))$.

Let us assume that r in the geometric semantic crossover is a constant, then the offspring is just a linear combination of the parents. This combination lies in the line segment intersecting the parents. This characteristic influenced the development of PrGP. That is, it is reasonable to investigate whether there is a better linear combination between two parents, and, the effects that this modification has in the converge of the algorithm.

In order to describe PrGP, let us rewrite the geometric semantic crossover with the constraint that r is a constant. The geometric crossover is computed as $o = \alpha p_1 + \beta p_2$ where $\alpha = r$ and $\beta = 1 - \alpha$. Using this notation it is evident that the geometric semantic operators constraints the values of α and β. PrGP removes these restrictions but are not geometric as defined by Moraglio et al. [18].

PrGP crossover is defined as follows: let p_1 and p_2 be the first and second parent, then the offspring o is computed as $o = \alpha p_1 + \beta p_2$ where α and β are calculated solving the following equation $A \cdot (\alpha, \beta)' = \mathbf{t}'$ where $A = (\mathbf{p}_1', \mathbf{p}_2')$, $\mathbf{p}_i = (s(p_i, x_1), \ldots, s(p_i, x_n))$ is the evaluation of parent i in all the inputs, and $\mathbf{t}$ is the target vector. By construction, the offspring o is the projection of $\mathbf{t}$ on the plane produced by the linear combination of $\mathbf{p}_1$ and $\mathbf{p}_2$ in the case of crossover. Given that o is created to minimize $||\mathbf{t} - (s(o, x_1), \ldots, s(o, x_n))\,||$; so, it corresponds to the orthogonal projection of $\mathbf{t}$ in that plane. Figure 5 depicts this process where A and B play the role of $\mathbf{p}_1$ and $\mathbf{p}_2$, it is observed target vector $\mathbf{t}$ which is outside the plane and the offspring is the orthogonal projection of $\mathbf{t}$ on the plane generated. Consequently, if the fitness function is the euclidean distance then the offspring has at least the fitness of the best parent.

At this point, we are in the position to introduce the extension performed to PrGP. PrGP has its origins in the geometric crossover removing the constraint that the

[3] x_i could be a input-vector or a scalar.

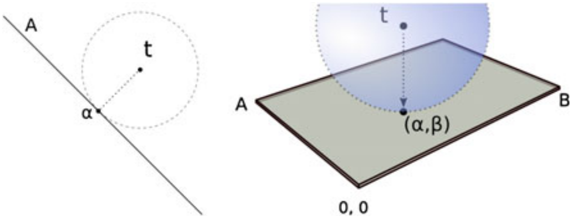


Fig. 5 The optimization process of coefficients. The orthogonal projection is finding the nearest point of target **t** into the hyperplane generated by an individual and its coefficients. On the left, it will find α, the point being orthogonal to **t**. On the right, the procedure on the plane (here, it finds two constants)

offspring must lie in the linear segment connecting the parents, and, instead, a linear combination of the parents is used to create the offspring. From this point, it is evident that PrGP has the constraint that the offspring has only two parents; however, being a linear combination it is straightforward to generalize this beyond two arguments.

PrGP creates an offspring as follows:

- Let $\{p_1, \ldots, p_k\}$ be the arguments of function $\sum^k$.
- The offspring is defined as $\sum_i^k \alpha_i p_i$ where the set $\{\alpha_i\}$ must be computed to minimize the fitness function, in this case the euclidean distance.

The extension performed to PrGP is that the offspring is produced by k parents, that is, the offspring is the orthogonal projection of target **t** on the hyper-plane generated by k parents.

Creating an offspring as the linear combination has the feature that the offspring's fitness is at least as good as the fitness of the best parent. This is clearly a greedy strategy that is suitable to learn the patterns in the training set; however, it is not clear whether this characteristic is reflected on the ability to generalize of the evolve model. This research gap is where our novel proposal, namely Root Genetic Programming (RGP), comes in. That is, RGP explores different combinations at root level, instead of using only the sum. However, the guarantee of producing an offspring that performs better or equal than the best involved parent cannot be preserved in general; this is a secondary effect of allowing more sophisticated combinations.

RGP creates an offspring as follows:

- Let $\{p_1, \ldots, p_k\}$ be the arguments of a function f randomly selected from the function set.
- In the case, f is the sum, i.e., $\sum^k$, the offspring is $\sum_i^k \alpha_i p_i$ which is basically what PrGP does.
- On the other hand, the offspring is defined as $\alpha f(p_1, \ldots, p_k)$ where α is computed to minimize $||\mathbf{t} - \alpha f(p_1, \ldots, p_k)||$.

Figure 6 exemplifies an individual evolved by RGP doing the following. Let $p = [x_1, x_2, x_3, (\exp x_2)]$ be the initial population and p_i refers to the i-th individual in population p. The first offspring was created by selecting function if from the

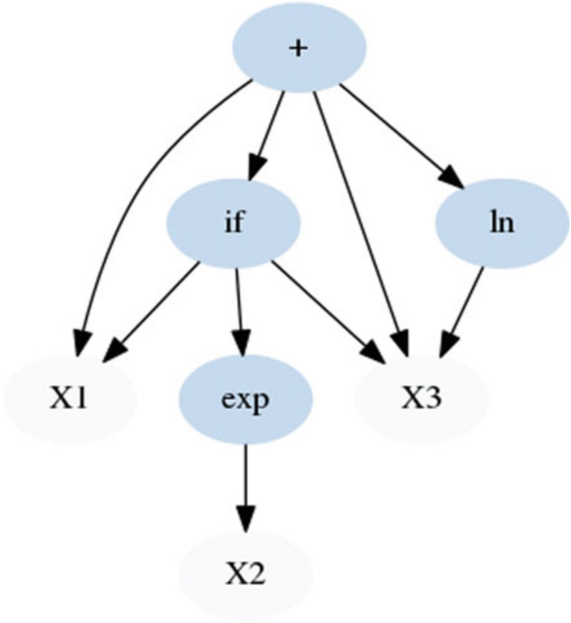


Fig. 6 An individual generated by RGP

function set and as arguments p_1, p_4, and p_3 (i.e., x_1, $(\exp x_2)$, and x_3) resulting in $o =$ (if x_1 $(\exp x_2)$ x_3). Offspring o was set to p_2 thus the population is $p =$ $[x_1$, (if x_1 $(\exp x_2)$ $x_3)$, x_3, $(\exp x_2)]$. The second offspring was created by selecting ln and as argument p_3 resulting in $(\ln x_3)$ this offspring was set in p_4 leaving the population as: $p = [x_1$, (if x_1 $(\exp x_2)$ $x_3)$, x_3, $(\ln x_3)]$. At this point we are in the position to show the procedure followed to create the the individual depicted in the figure. $\sum^k$ was randomly selected from the function set where $k = 4$ and the arguments were p_1, p_2, p_3 and p_4 this produces as offspring the individual depicted on Fig. 6.

4 Problem and Parameters Settings

At this point, we are in the position to analyze the performance of the systems proposed on a sentiment analysis benchmark. We decided to use as testbed task 1 of TASS 2015 (Taller de Análisis de Sentimientos en la SEPLN) workshop [26]. This task consists on performing an automatic sentiment classification to determine the global polarity—six polarity levels positive (P), strong positive (P+), neutral (NEU), negative (N), strong negative (N+) and no-sentiment tag (NONE)—of each tweet in the provided dataset. The data consists of 7,218 tweets distributed as follows P: 1232, P+: 1652, N: 1335, N+: 847, NEU: 670 and NONE: 1482. Spanish native speakers from different universities around the world performed the labeling task.

In order to test the generality of the models evolved, a 10-fold cross-validation was performed. Each fold was treated as follows: the first part of each fold, i.e., the data used to train the algorithm was split in two: 80 % is used to actually train the algorithm, and the rest 20 % was used as validation set. That is, each fold contains a training set, a validation set and a test set. The validation set was used to make early stopping, i.e., to keep the model that best performed on the validation set. In addition to this, the training set is always balanced that is, each label contains the same number of exemplars.

There are different paths to tackle classification problems with GP. The simplest of this is to treat the classification problems as a symbolic regression problem, this is straightforward in the case of two classes, e.g., on class is identified with -1 and the other with 1, in this scenario the output, of evolved tree, is just the sign albeit the class. However, in the sentiment analysis problem presented here there are six classes. Under this circumstance, the strategy followed is the one vs one classifier, i.e., for all the pairs of different labels a different classifier is trained. Given that there are 6 different labels then it is needed to create 15 different classifiers. Each pair of labels is treated as an independent symbolic regression problem where the output was -1 and 1 to represent each class.

In addition to using one vs one strategy, it was decided to predict the class using an ensemble of 5 different instances. That is, given a pair of labels, a system is trained using 80 % of the data randomly selected, with a particular seed, from the training part of the fold, and validated using the remaining 20 %. Then another instance is created by choosing another seed, and, consequently, selecting different exemplars. This process is repeated 5 times and the final prediction on the test set is the sign of the average of all the systems.

We decided to test different parameters settings for our proposal RGP, as well as for PrGP. The first change was the replacement of add function, i.e. $+$ in the function set by the sum, i.e., $\sum^{k}$. These modifications allows to have more than 2 arguments, hereafter, it will make explicit the number of arguments by the value of k. Secondly, inspired by the PrGP's feature that guarantees offspring as fit as the best of the parents, it was decided to force RGP to keep an offspring only when this offspring is better than its parents this constraint is referred here after as *greedy*. *Standard* is used to refer a systems that keep an offspring regardless of its fitness. Finally, it is traditionally to initialize the population using the ramped half-and-half method, besides this procedure, this contribution test another approach which consists in creating the initial population with only leafs, i.e., the inputs of the problem. The former method is referred as *Ramped half-and-half* and the later is referred as *Inputs*.

Table 2 presents the parameters used by all GP systems. All systems use a steady state strategy with tournament selection. The size of the population is considerable greater than traditional parameters, and, the number of generations is lower; however, the number of individuals evaluated (100,000) are comparable with the parameters used traditionally. In fact, our first approach was to use a traditional configuration such as a population of 1000 individuals evolved during 100 generations; however, on preliminary runs, this standard parameter configuration was outperformed by the used of a larger population. Furthermore, it was observed that the convergence rate

Table 2 Genetic Programming's parameters

Parameter	Value
Selection	Tournament size 2
Population size	100,000
Number of generations	10
Function Set ($\mathcal{F}$)	$\{\sum^k, \times, /, \lvert\cdot\rvert, \text{exp}, \sqrt{\cdot}\ \sin, \cos, \text{sigmoid}, \ln, (\cdot)^2, \text{if}\}$

decreased considerable around generation 10th, so we decided to cut the evolution at generation 10th (more about the number of generations below). The fitness function is the mean square error, this is coherent with the procedure used to optimize the constants in the tree. Furthermore, the early stopping individual is selected using the balance error rate (BER) this is to consider the imbalance in the validation set.

5 Results

At this point, we are in the position to start comparing the performance of the different systems. One of the unusual parameters, used in this contribution, is the number of generations which is considerable low. To address whether this parameter affects the performance, Table 3 presents the average of the fraction of total individuals evaluated to find the early-stopping-individual on the different systems, namely PrGP and RGP, with the different parameters. The number of arguments k of $\sum^k$ is specified in the first column.

It is observed, from Table 3, that the different configurations of PrGP and RGP with the greedy strategy obtained the early-stopping-individual before evaluating 65 % of

Table 3 Average with 95 % confidence interval of the fraction of total individuals evaluated to reach the best individual in the validation set, i.e., the early stopping solution

$\sum^k$	Ramped half-and-half			Inputs		
	PrGP	RGP		PrGP	RGP	
		Greedy	Standard		Greedy	Standard
2	0.529 ± 0.016	0.610 ± 0.016	0.873 ± 0.010	0.528 ± 0.015	0.623 ± 0.016	0.876 ± 0.010
3	0.381 ± 0.014	0.523 ± 0.016	0.838 ± 0.010	0.408 ± 0.015	0.564 ± 0.017	0.845 ± 0.010
4	0.339 ± 0.015	0.450 ± 0.014	0.762 ± 0.011	0.342 ± 0.014	0.498 ± 0.016	0.774 ± 0.011
5	0.302 ± 0.013	0.405 ± 0.014	0.698 ± 0.013	0.314 ± 0.014	0.442 ± 0.016	0.704 ± 0.013
6	0.278 ± 0.013	0.365 ± 0.013	0.658 ± 0.014	0.293 ± 0.013	0.413 ± 0.015	0.651 ± 0.014
7	0.254 ± 0.012	0.356 ± 0.013	0.616 ± 0.014	0.250 ± 0.011	0.380 ± 0.014	0.616 ± 0.015
8	0.244 ± 0.011	0.337 ± 0.013	0.600 ± 0.015	0.228 ± 0.010	0.338 ± 0.012	0.599 ± 0.015
9	0.215 ± 0.009	0.322 ± 0.013	0.581 ± 0.016	0.209 ± 0.008	0.319 ± 0.011	0.571 ± 0.015
10	0.203 ± 0.009	0.291 ± 0.011	0.567 ± 0.016	0.199 ± 0.008	0.300 ± 0.011	0.558 ± 0.016

the total of individuals evaluated. This clearly indicates that the number of generations is not affecting the performance of these systems. On the other hand, RGP with the standard strategy and using $k \leq 6$ obtained the early-stopping-individual after evaluating 65 % of the total individuals; and, for the case $k = 1$, the early-stopping-individual is obtained after 80 % of the individuals had been evaluated. That is, in the later case, the early-stopping-individual is obtained around generation 8 (number of generation is 10). Consequently, one may wonder whether by increasing the number of generations the performance for these systems would improved as well.

Table 4 presents the average performance using the balance error rate (BER) of the early-stopping-individual on the different systems and configurations. The best performance in each configuration is highlighted in boldface to facilitate the comparison. It is observed that the best performance is always obtained by RGP using inputs as initial population; and for $k \geq 3$ the best performance is obtained by RGP with the standard evolution strategy. The system having the best performance was compared against all other systems; in the case, the difference in performance was statistically significant (with a confidence of 95 %) the superscript $*$ was used on the losing system. This comparison was performance using the Wilcoxon signed-rank test [37] and the p-values were adjusted using the Holm–Bonferroni method [10] in order to consider the multiple comparisons performed.

It is observed that most of the times, the systems having the best performance is statistically better than the others, the only exception is the systems RGP with inputs as initial population and $k = 3$. Let us focus on RGP with standard evolution and inputs as initial population, i.e., seventh column. In this column, the best performance was $k = 10$. It is natural to test whether this particular configuration is statistically better than the others. Performing an equivalent statistical analysis as the previously described, it is obtained that the best system is statistically better than the systems

Table 4 Average performance (BER) of the best individual found in the validation set (early stopping)

$\sum^k$	Ramped half-and-half			Inputs		
	PrGP	RGP		PrGP	RGP	
		Greedy	Standard		Greedy	Standard
2	33.541*	33.271*	35.703*	33.450*	**33.049**	34.695*
3	33.081*	32.928*	33.816*	33.207*	32.728	**32.676**
4	33.119*	32.735*	33.347*	32.994*	32.684*	**32.165**
5	33.054*	32.775*	33.256*	32.899*	32.679*	**32.061**
6	32.856*	32.704*	33.269*	32.782*	32.401*	**31.956**
7	32.981*	32.674*	33.231*	32.849*	32.385*	**32.056**
8	32.873*	32.729*	33.206*	32.726*	32.350*	**31.991**
9	32.846*	32.828*	33.250*	32.746*	32.332*	**32.033**
10	32.882*	32.684*	33.139*	32.717*	32.401*	**31.916**

Table 5 Performance in terms of macro-F1 on the test set of different GP systems

$\sum^k$	Ramped half-and-half			Inputs		
	PrGP	RGP		PrGP	RGP	
		Greedy	Standard		Greedy	Standard
2	0.356	**0.362**	0.340	0.353	0.353	0.345
3	0.356	**0.369**	0.356	0.348	0.359	0.363
4	0.359	0.373	0.372	0.349	0.365	**0.379**
5	0.363	0.368	0.373	0.353	0.364	**0.377**
6	0.363	0.370	0.368	0.354	0.367	**0.377**
7	0.371	**0.375**	0.370	0.354	0.371	**0.375**
8	0.369	0.377	0.367	0.357	0.367	**0.383**
9	0.369	**0.383**	0.370	0.354	0.373	0.380
10	0.373	**0.380**	0.374	0.361	0.369	0.376

with $k \leq 4$, consequently, for the remaining values of k the null hypothesis cannot be rejected.

At this point, it has been analyzed the performance of the systems on the validation set. In order to test the generality of the models, Table 5 presents the performance of the different systems on the test set (i.e., 10-fold cross-validation) using as performance measure macro-F1. macro-F1 is defined as the average F1 score over all the different classes, and F1 is the harmonic mean of precision and recall, i.e., $F1 = 2\frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}}$.

It can be seen from Table 5, that RGP outperformed PrGP. RGP using the ramped half-and-half method and the greedy strategy obtained the best performance of $k = 2, 3, 7, 9, 10$ and RGP using inputs and standard evolution obtained the best performance in $k = 4, \ldots, 8$, these two systems obtained equivalent performance in $k = 7$.

The highest macro-F1 value is 0.383 which was obtained by RGP using the ramped half-and-half method, greedy evolution and $k = 9$ and RGP with inputs, standard evolution and $k = 8$. It is interesting to note that with the information presented so far it is not evident which parameters is obtaining the best performance, it seems that k must be above 4 and taking the performance of the test set then one must use a greedy strategy when the ramped half-and-half method is used, and a standard evolution when only the inputs are the initial population.

At this point, we have compared the performance of PrGP and RGP on different configurations and parameters settings, it is time to test whether RGP is competitive with other state-of-the-art classifiers. Table 6 presents the performance (using score F1 for the different classes and macro-F1) of different classifiers trained with the scikit-learn library [22]. From the table, it is observed that RGP is the second best. RGP was outperformed by SVM in the macro-F1 and in almost all classes except

Table 6 Performance in terms of score F1 of different classifiers. The performance of RGP is reported as a single instance classifier; Table 7 contains the RGP's performance as part of a group

Classifier	P	P+	N	N+	NEU	NONE	Macro-F1
RGP	0.265	0.534	0.314	0.396	0.199	0.548	0.327
SVM	0.285	0.603	0.425	0.408	0.126	0.558	0.401
KNN ($K = 30$)	0.235	0.507	0.356	0.307	0.074	0.394	0.312
Naive Bayes (Gaussian)	0.25	0.457	0.369	0.282	0.079	0.344	0.297
Extreme Random Trees	0.24	0.484	0.301	0.227	0.046	0.447	0.291
AdaBoost	0.127	0.477	0.355	0.301	0.014	0.442	0.286

NEU where RGP obtained the highest F1 score. The third place was obtained by K-nearest neighbor classifier using $K = 30$.[4]

RGP obtained the second best performance among all the classifiers presented on Table 6. Unfortunately, RGP's running time is the highest of all the classifiers presented. This issue is noticeable when RPG is used to create an ensemble. For instance, the performance of SVM was computed in order of minutes whilst the RGP's performance was recorded in the order of hours. It is important to note that RGP has a lower complexity than traditional GP systems, nonetheless there is still room for improvement in order to make it competitive with the running time of traditional classifiers.

So far, we have compared PrGP and RGP using different parameter settings such as: the procedure to create the initial population, the greedy or standard evolution, and k. In addition, the RGP system having one of the best performances was compared against state-of-the-art classifiers. Although RGP obtained the second best performance among the classifiers analyzed, it is still missing to investigate the consequence that one particular parameter affecting the running time of the system has on the performance. This is the size of the ensemble. Let us remember that it was used one-vs-one strategy to convert the multi-class problem into a binary one, then for each pair of classes, the GP system under study was instantiated with different seeds and selecting randomly a training set and validation set from corresponding part of the fold. This process was repeated 5 times and the final prediction is the average of these instances. It is natural to ask whether this arbitrary number of times, i.e., size of the ensemble, was correctly selected.

In order to give insight into the behavior of the ensemble when its size is varied Table 7 presents the performance, using the score F1, of RGP with inputs, standard evolution and $k = 10$; when the size of ensemble is varied from 1 to 30. It is observed

[4] The K-nearest neighbor classifier was tested with varying K from 10 to 100 and $K = 30$ gave the highest result.

Table 7 Performance in terms of score F1 of varying the size of the ensemble using RGP with inputs as initial population and standard evolution with $k = 10$

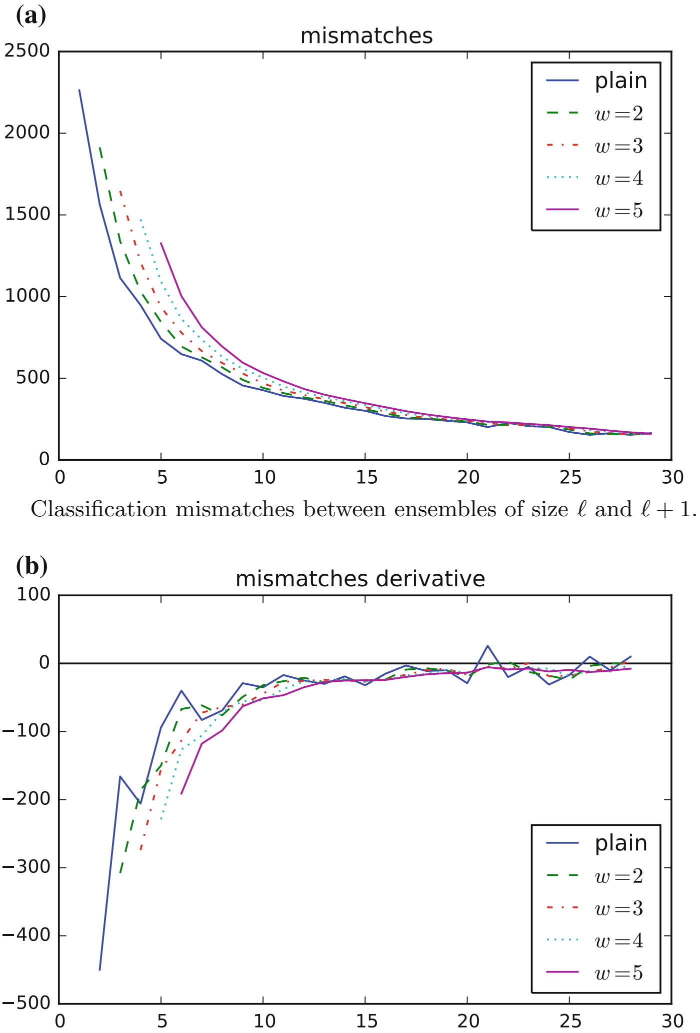
Size (ℓ)	P	P+	N	N+	NEU	NONE	Macro-F1
1	0.251	0.464	0.272	0.343	0.173	0.460	0.327
2	0.257	0.505	0.298	0.360	0.187	0.507	0.352
3	0.267	0.510	0.307	0.379	0.175	0.516	0.359
4	0.262	0.523	0.312	0.388	0.190	0.532	0.368
5	0.265	0.534	0.314	0.396	0.199	0.548	0.376
6	0.280	0.547	0.320	0.393	0.205	0.552	0.383
8	0.276	0.548	0.330	0.403	0.203	0.558	0.386
10	0.283	0.549	0.342	0.408	0.199	0.555	0.389
12	0.279	0.547	0.345	0.409	0.192	0.554	0.388
14	0.272	0.554	0.344	0.409	0.202	0.558	0.390
16	0.271	0.559	0.351	0.413	0.196	0.560	0.391
18	0.272	0.561	0.352	0.412	0.199	0.563	0.393
20	0.276	0.565	0.354	0.409	0.200	0.565	0.395
22	0.265	0.564	0.356	0.410	0.208	0.565	0.395
24	0.264	0.568	0.354	0.411	0.208	0.565	0.395
26	0.267	0.566	0.354	0.414	0.208	0.563	0.395
28	0.274	0.566	0.353	0.413	0.210	0.562	0.396
30	0.274	0.569	0.355	0.417	0.214	0.562	0.399

that the most drastic change is from 1 to 2, i.e., from a single instance (not really an ensemble) to the minimum ensemble. In addition, it is observed that for all the classes the corresponding score F1 increases as the size of the ensemble increases. It is observed from the table that the performance according to macro-F1 reaches at stable point around 20, i.e., it reaches 0.395, then it increases on 28 and 30.

Comparing the performance of RGP when the size is 30 against the performance of SVM, it is observed that RGP outperformed SVM in N+, NEU and NONE and, consequently, SVM obtained the highest F1 on P, P+, N and macro-F1. Nonetheless, the difference in performance between these two systems is small which might be an indicator that RGP could outperform SVM in all this benchmark by optimizing the rest of RGP's parameters.

Determining the Size of the Ensemble Automatically. The optimality of the classifier setup is related to the ability to determine the minimum size that maximizes the classification rate. The classification rate in our approach can be indirectly measured using an additional set of non-labeled examples. As other real-world problems, the TASS's benchmark is full of unlabeled examples; this extra set does not poses a complex requirement in practice.

The idea consists in finding on-line the proper ensemble's size measuring the agreement as we add members to the ensemble. The assumption is that an ensemble of size ℓ can be improved adding an extra member; therefore, the number of mismatches among ℓ and $\ell + 1$ is relatively low, mismatches will be found when ℓ is distant relatively small than for larger values. The assumption should work whenever each member of the ensemble performs better than a random classifier; this behav-



Classification mismatches between ensembles of size ℓ and $\ell + 1$.

The differences of mismatches among consecutive runs, an approximation to the derivative in the region of interest.

Fig. 7 Measuring the agreement between ensembles of size ℓ and $\ell + 1$ through the number of mismatches among predictions. The number of mismatches decreases as the ensemble's size increases, i.e., the derivative approaches to zero. The smoothing parameter w removes *statistical noise* from the *curves*

ior resembles *Boosting* [39]; nonetheless, our approach differs significantly in the algorithms, the requirements, and the strategy to take advantage of the assumption.

Figure 7 shows the behavior of mismatches for those ensembles listed in Table 7. In the left side, the number of mismatches among ensembles of sizes ℓ and $\ell + 1$ are shown; on the right side, the derivative of the number of mismatches is approximated. Notice how the real mismatches (plain legend) have some undesirable noise, then it is necessary to smooth the curve. For this purpose we added the parameter w, which defines the size of a sliding window averaging the number of consecutive mismatches. As w increases, the curve becomes smoother and it is more comfortable to decide the proper size of the ensemble, that is, while the derivative remains negative the classifier had not converged (from the consensus perspective).

Even when this procedure determines a proper size of the ensemble, the precise value of w is dependent on the application domain. As a rule of thumb, a small value $w \leq 5$ should be used; however, large values are preferred when high quality classifiers are needed. The tradeoff consists on interchanging accuracy and construction time because the slow convergence is an intrinsic problem on large w. The slow convergence is exposed as small strict-positive values in the second derivative of the consensus function; therefore, it is possible to tackle this problem stopping the optimization whenever a small threshold is reached. The figures illustrate the slow improvement in the consensus for ensembles larger than 20, this size corresponds to the highest performance of the actual Macro-F1 depicted in Table 7.

6 Conclusions

In this contribution, we have presented an extension of our previous semantic operators PrGP and a novel GP system, namely Root Genetic Programming (RGP). The idea of these two systems is that the offspring is a combination of the complete parents. In the case of PrGP is the linear combination of the parents; the extension is that the offspring can have more than two parent, i.e., parameter k in our notation. Our novel RGP explores more than just a linear combination, e.g., an offspring could be the result of joining three parents with function if.

PrGP and RGP were tested on a sentiment analysis benchmark. The results show that these systems are feasible to tackle this problem. In fact, RGP is very competitive outperforming several state-of-the-art classifiers. Comparing RGP against SVM, it was observed that SVM outperformed RGP; however, the difference in performance is small, and there are a number of scenarios where one would prefer to loose a little bit in performance to elicit knowledge from the model obtained. For example, in the case one is interested on feature selection or feature engineering, it is easier to perform these task using Genetic Programming than a SVM, in fact, GP is performing feature selection automatically.

PrGP and RGP was tested using different parameter settings, some of them are not very usual in the community. Let us pay attention at the procedure used to initialize the population, there, it was tested the ramped half-and-half against a simple approach of selecting only inputs. The results show that selecting only inputs is competitive, in fact, in the validation set, using this technique is better that the traditional ramped half-and-half. This result has an important consequence in the complexity of the algorithm, that is, it is simpler to create the initial population using only the inputs than implementing the ramped half-and-half method.

The size of the ensemble also plays a major role in the performance of RGP. It was observed that the RGP's performance increases when the ensemble size increases, it reaches at stable point around 20. This behavior should be investigated further, for example, it is still unknown whether the use of more sophisticated methods to create the ensemble would improve the performance or reduce the computational cost to create the final model.

Finally, the performance of RGP (using an ensemble of size 20) was similar than SVM's performance. As consequence, it is interesting to investigate whether by optimizing the rest of RGP's parameters it could be possible to improve RPG's performance, and, therefore, to outperform SVM. For example, the number of generations was set to 10; however, given that it is being used an early-stopping strategy then it is possible to stop the evolution using the early stopping rounds parameter. That is, the evolution is stopped when more than early-stopping-rounds individuals have been evaluated and the early-stopping individual has been kept constant. Another parameter that plays an important role is the function set. In this contribution, it has not been explored whether the performance can be improved by using different function sets. Related to this, PrGP was extended by including more than two arguments to the sum. Clearly, this modification can also bee applied to the product, that is, the product can have more than two argument. We will explore the performance of RGP under these different settings in future work.

References

1. Arora, S., Mayfield, E., Penstein-Ros, C., Nyberg, E.: Sentiment classification using automatically extracted subgraph features. In: Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text, CAAGET '10, pp. 131–139, Stroudsburg, PA, USA (2010). Association for Computational Linguistics. 00030
2. Baeza-Yates, P.A., Ribeiro-Neto, B.A.: Modern Information Retrieval, 2 edn. Addison-Wesley (2011)
3. Castelli, M., Silva, S., Vanneschi, L.: A C++ framework for geometric semantic genetic programming. Genet. Program. Evol. Mach. **16**(1), 73–81 (2014). 00004
4. Castelli, M., Trujillo, L., Vanneschi, L., Silva, S., Z-Flores, E., Legrand, P.: Geometric semantic genetic programming with local search. In: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference, GECCO '15, pp. 999–1006. ACM, New York, NY, USA (2015). 00000
5. Doucette, J., Lichodzijewski, P., Heywood, M.: Evolving coevolutionary classifiers under large attribute spaces. In: Riolo, R., O'Reilly, U.-M., McConaghy, T. (eds.) Genetic Programming Theory and Practice VII, Genetic and Evolutionary Computation, pp. 37–54. Springer, US (2010). 00008. doi:10.1007/978-1-4419-1626-6_3

6. Escalante, H.J., Garcia-Limon, M.A., Morales-Reyes, A., Graff, M., Montes-y Gomez, M., Morales, E.F., Martinez-Carranza, J.: Term-weighting learning via genetic programming for text classification. Knowl.-Based Syst. (2015). 00000
7. Espejo, P.G., Ventura, S., Herrera, F.: A survey on the application of genetic programming to classification. IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. **40**(2):121–144 (2010)
8. Giannakopoulos, G., Mavridi, P., Paliouras, G., Papadakis, G., Tserpes, K.: Representation models for text classification: a comparative analysis over three web document types. In: Proceedings of the 2Nd International Conference on Web Intelligence, Mining and Semantics, WIMS '12, pp. 13:1–13:12. ACM, New York, NY, USA (2012)
9. Graff, Mario, Tellez, E.S., Villasenor, E., Miranda-Jiménez, S.: Semantic genetic programming operators based on projections in the phenotype space. Res. Comput. Sci. **94**, 73–85 (2015)
10. Holm, S.: A simple sequentially rejective multiple test procedure. Scand. J. Stat. **6**(2):65–70 (1979). 10011
11. Iqbal, M., Browne, W.N., Zhang, M.: Reusing building blocks of extracted knowledge to solve complex, large-scale boolean problems. IEEE Trans. Evol. Comput. **18**(4):465–480 (2014). 00019
12. Korns, M.F.: Large-scale, time-constrained symbolic regression. In: Riolo, R., Soule, T., Worzel, B. (eds.) Genetic Programming Theory and Practice IV, Genetic and Evolutionary Computation, pp. 299–314. Springer, US (2007). 00019 doi:10.1007/978-0-387-49650-4_18
13. Korns, M.F.: Large-scale, time-constrained symbolic regression-classification. In: Riolo, R., Soule, T., Worzel, B. (eds.) Genetic Programming Theory and Practice V, Genetic and Evolutionary Computation Series, pp. 53–68. Springer, US, (2008). 00020 doi:10.1007/978-0-387-76308-8_4
14. Korns, M.F., Nunez, L.: Profiling symbolic regression-classification. In: Genetic Programming Theory and Practice VI, Genetic and Evolutionary Computation, pp. 1–14. Springer, US (2009). 00011 doi:10.1007/978-0-387-87623-8_14
15. Liu, B.: Sentiment Analysis: Mining Opinions, Sentiments, and Emotions, 381 p. Cambridge University Press (2015). ISBN: 1-107-01789-0
16. Mayfield, E., Penstein-Rosé, C.: Using feature construction to avoid large feature spaces in text classification. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, GECCO '10, pp. 1299–1306. ACM, New York, NY, USA (2010). 00013
17. McConaghy, T.: Latent variable symbolic regression for high-dimensional inputs. In: Riolo, R., O'Reilly, U.-M., McConaghy, T. (eds.) Genetic Programming Theory and Practice VII, Genetic and Evolutionary Computation, pp. 103–118. Springer, US (2010). 00007. doi:10.1007/978-1-4419-1626-6_7
18. Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric semantic genetic programming. In: Coello Coello, C.A., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) Parallel Problem Solving from Nature - PPSN XII, number 7491 in Lecture Notes in Computer Science, pp. 21–31. Springer, Berlin, Heidelberg (2012)
19. Navarro, G., Raffinot, M.: Flexible Pattern Matching in Strings – Practical on-line search algorithms for texts and biological sequences, 280 p. Cambridge University Press (2002). ISBN 0-521-81307-7
20. Padr, L., Stanilovsky, E.: Freeling 3.0: Towards wider multilinguality. In: Proceedings of the Language Resources and Evaluation Conference (LREC 2012). ELRA, Istanbul, Turkey (2012)
21. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Found. Trends Inf. Retr. **2**(1–2), 1–135 (2008)
22. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
23. Peng, T., Zuo, W., He, F.: Svm based adaptive learning method for text classification from positive and unlabeled documents. Knowl. Inf. Syst. **16**(3), 281–301 (2008)
24. Poli, R.: TinyGP. See Genetic and Evolutionary Computation Conference (GECCO-2004) competition (2004). http://cswww.essex.ac.uk/staff/sml/gecco/TinyGP.html

25. Poli, R., Langdon, W.B., McPhee, N.F.: A Field Guide to Genetic Programming. Lulu Enterprises UK Ltd (2008)
26. Romn, J.V., Morera, J.G., Garca Cumbreras, M.A., Martnez Cmara, E., Teresa Martn Valdivia, M., Alfonso Urea Lpez, L.: Overview of tass 2015. CEUR Workshop Proc. **1397**:13–21 (2015)
27. Sammut, C., Webb, G.I. (eds.): Statistical natural language processing. Encyclopedia of Machine Learning, pp. 916–916. Springer, US (2010)
28. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. **34**(1), 1–47 (2008)
29. Shannon, C.E.: A mathematical theory of communication. ACM SIGMOBILE Mob. Comput. Commun. Rev. **5**(1), 3–55 (2001)
30. Sidorov, G., Miranda-Jiménez, S., Viveros-Jiménez, F., Gelbukh, A., Castro-Sánchez, N., Velásquez, F., Díaz-Rangel, I., Suárez-Guerra, S., Treviño, A., Gordon, J.: Empirical study of machine learning based approach for opinion mining in tweets. In: Proceedings of the 11th Mexican International Conference on Advances in Artificial Intelligence - Volume Part I, MICAI'12, pp. 1–14. Springer, Berlin, Heidelberg (2013)
31. Silla, C.N. Jr., Pappa, G.L., Freitas, A.A., Kaestner, A.A.: Automatic text summarization with genetic algorithm-based attribute selection. In: Lemaître, C., Reyes, C.A., González, J.A. (eds.) Proceedings 9th Ibero-American Conference on AI Advances in Artificial Intelligence - IBERAMIA 2004. Lecture Notes in Computer Science, vol. 3315, pp. 305–314. Springer, Puebla, Mexico, 22–26 November 2004
32. Silva, S.: Gplab: A genetic programming toolbox for matlab. http://gplab.sourceforge.net
33. Uy, N.Q., Anh, P.T., Doan, T.C., Hoai, N.X.: A study on the use of genetic programming for automatic text summarization. In: Dang-Van, H., Sanders, J. (eds.) The Fourth International Conference on Knowledge and Systems Engineering, KSE 2012, pp. 93–98, Danang, Vietnam, 17–19 August 2012
34. Vanneschi, L., Castelli, M., Manzoni, L., Silva, S.: A new implementation of geometric semantic GP and its application to problems in pharmacokinetics. In: Krawiec, K., Moraglio, A., Hu, T., Ima Etaner-Uyar, A., Hu, B. (eds.) Genetic Programming, number 7831 in Lecture Notes in Computer Science, pp. 205–216. Springer, Berlin, Heidelberg (2013)
35. Vanneschi, L., Castelli, M., Silva, S.: A survey of semantic methods in genetic programming. Genet. Program. Evol. Mach. **15**(2), 195–214 (2014). June
36. White, D.R.: Software review: the ecj toolkit. Genet. Program. Evol. Mach. **13**(1):65–67 (2012)
37. Wilcoxon, F.: Individual comparisons by ranking methods. Biom. Bull. **1**(6), 80 (1945)
38. Zhang, Y., Bhattacharyya, S.: Genetic programming in classifying large-scale data: an ensemble method. Inf. Sci. **163**(1–3):85–101 (2004). 00061
39. Zhou, Z.-H.: Ensemble Methods: Foundations and Algorithms. CRC Press (2012)