Information Sciences 562 (2021) 116–135

Contents lists available at ScienceDirect

Information Sciences

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

Sentiment analysis with genetic programming

Airton Bordin Junior *, Nádia Félix F. da Silva, Thierson Couto Rosa, Celso G.C. Junior

Institute of Informatics, Federal University of Goiás, Goiânia 74690-900, Brazil

ARTICLE INFO

Article history:
Received 25 November 2019
Received in revised form 7 January 2021
Accepted 10 January 2021
Available online 1 February 2021

Keywords:
Sentiment analysis
Genetic programming
Lexicon
Classifiers

ABSTRACT

With the advent of online social networks, people became more eager to express and share their opinions and sentiment about all kinds of targets. The overwhelming amount of opinion texts soon attracted the interest of many entities (industry, e-commerce, celebrities, etc.) that were interested in analyzing the sentiment people express about what they produce or communicate. This interest has led to the surge of the sentiment analysis (SA) field. One of the most studied subfields of SA is polarity detection, which is the problem of classifying a text as positive, negative, or neutral. This classification problem is difficult to solve automatically, and many hand-adjusted resources are needed to overcome the difficulties in detecting sentiment from text. These resources include hand-adjusted textual features as well as lexicons. Deciding which resource and which combination of resources are more appropriate to a given scenario is a time-consuming trial-and-error process. Thus, in this work, we propose the use of Genetic Programming (GP) as a tool for automatically choosing, combining, and classifying sentiment from text. We propose a series of functions that allow GP to deal with preprocessing tasks, handcrafted features, and automatic weighting of lexicons for a given training set. Our experiments show that our GP solution is competitive and sometimes better than SVM and superior to naïve Bayes, logistic regression, and stochastic gradient descent, which are methods used in SA competitions.

1. Introduction

Sentiment analysis (SA) is the research field that aims to classify the emotions of a particular text, usually as polarities, i.e., positive, negative, or neutral [1]. Research in this field has gained prominence since the emergence of online social network platforms, such as Twitter[1] and Facebook[2]. These social tools allow their users to share text, usually short opinion text, with their social contacts. Soon after they were developed, online social networks became available via smart phones, which contributed even more to their popularity among all socioeconomic classes. The consequence of the popularity of social networks is that the text posted on social networks has become a very important source of information about what people think and feel about anything, including trademarks, products, celebrities, and political subjects. Thus, there is great interest from industry, e-commerce, celebrities, and political agents, among other entities, in applying SA to these texts. This demand has motivated research in SA [2–4].

* Corresponding author.
E-mail addresses: airtonbjunior@inf.ufg.br (A.B. Junior), nadia@inf.ufg.br (N.F.F. da Silva), thierson@inf.ufg.br (T.C. Rosa), celso@inf.ufg.br (C.G.C. Junior).

[1] https://twitter.com/.
[2] https://www.facebook.com/.

https://doi.org/10.1016/j.ins.2021.01.025
0020-0255/

SA is performed by *sentiment classifiers*, which are text classifiers developed to identify the polarity of pieces of text according to the prevalent sentiment they transmit. Sentiment classifiers are often generated manually by specialists in the specific domain of the classification, thus capturing the experience of the designer in the analyzed context [5–8]. However, the manual approach increases the model generation cost for each scenario, penalizes generalization and does not scale to large collections of messages, such as those derived from online social networks [9].

Given the massive amount of text to be classified and the cost involved in manually generating a classifier, some form of automatic classifier generation is mandatory. Automatic sentiment classification approaches are commonly divided into three classes [7,10,11]: (i) lexicon-based techniques, (ii) machine learning (ML)-based techniques, and (iii) hybrid methods.

A lexicon-based strategy is a nonsupervised approach that consists of going through the text word by word and consulting the polarity (and perhaps the polarity intensity) of each term in a lexicon. A combining function (such as a sum or average) is applied to make the final prediction regarding the overall sentiment of the text. This approach is easy to implement, and its accuracy greatly depends on the quality of the lexicon used [12].

Machine learning strategies apply learning algorithms (SVM, logistic regression, etc.) that can derive a classifier automatically from a set of texts with polarities that are usually manually labeled (the *training set*). Hybrid methods combine the first two approaches.

Unfortunately, neither of the first two approaches is able to perform polarity classification accurately in most cases. For instance, lexicon-based approaches alone are not able to deal with cases in which none of the words in the text occur in the lexicon as well as other situations [13]. On the other hand, machine learning methods alone are not able to capture the polarity of individual words as well as lexicon approaches. Thus, hybrid approaches have attracted research attention as an alternative solution for SA.

Despite the many advances in the area, SA continues to be a difficult text classification problem. Sentiment analysis suffers from the same difficulties found in other text classification tasks (such as topic classification), which include the curse of dimensionality, vocabulary sparsity, and category imbalance, among others. In addition, SA detection has its own idiosyncrasies, which further complicate matters. Texts appearing in polarity detection applications are mostly short opinion texts, such as tweets and product reviews, and contain many typos, slang, emoticons, emojis, and rhetorical figures, such as sarcasm and irony. As a consequence, most preprocessing that is usually applied to the words in other text classification tasks (stopword filtering, feature selection, normalization, etc.) is usually not sufficient for SA. Thus, handcrafted features (part-of-speech tagging, features informing the presence of negation, uppercase words, the polarity of emoticons or emojis, etc.) are often applied to enhance text representation before a model is trained.

In addition, there are many possibilities for combining different types of features (text-directed-derived features such as tf-idf[3] and handcrafted features). Additionally, there are many lexicons available in SA software tools, and some have been proposed in the literature; some of them, as we show, are more useful than others. Deciding which of these resources should be used and how to combine them for a given context is decisive for the success of SA. In Section 5.4 of this paper, we present the examples of Task 9 of SemEval 2014 and Task 4 Subtask A of the SemEval 2016 competitions to show how combinations of the abovementioned resources make the difference for classifier generation. In our experiments in which we apply the same features for different learners, we see that classifier trained using a SVM, a state-of-the-art learning algorithm, exceeds that generated via logistic regression (LR) in four out of six test datasets. However, the SemEval winning solution adopted LR as the learning algorithm. In fact, the competitors were allowed to use any combination of features, handcraft their own features, and apply any lexicon. The combination of resources was decisive for the winning solution as it compensated for the weakness of LR and allowed the final model to outperform SVM-based models in the competition. Nevertheless, combining all these resources (many features and many lexicons) is time-consuming manual work since there are many features and lexicons and the number of all possible combinations of them is exponential.

We show in this paper that genetic programming (GP) [14] automatically solves this resource combination problem as an optimization approach to evolve the best resource combinations. Once one defines functions to deal with each kind of feature/resource, GP finds approximations of the optimal combinations of these functions. Even preprocessing decisions, such as punctuation removal or stopword filtering, can be left for GP to decide whether they should be applied or not. Thus, enormous manual/mental effort savings are garnered to generate a sentiment classifier.

We propose functions to implement a large number of preprocessing tasks usually applied in SA. These functions are devised in a way that their results can be used in the processing of other functions, allowing the creation of a hierarchical structure (tree) of functions (also known as *program* or *individual* in GP jargon). These trees are manipulated by genetic programming via genetic operators (crossover, replication, and mutation) to evolve new trees. In particular, we propose a function that is able to obtain a weighted average of the polarities of the words in a text. The weight (importance) of each lexicon is taken into consideration in this function. These weights are obtained automatically by GP, which allows us not only to compute the polarities of words in the text better but also to identify which lexicons are more important to a given training dataset. In our case, this weighting scheme showed that only a few of the lexicons used are truly relevant and that they seem to complement each other.

[3] Abbreviation for term frequency-inverse document frequency. It is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

In addition to combining many resources, GP shows itself to be a promising sentiment classification tool. We compared GP with the three machine learning algorithms used by the three best-ranked competitors in Task 9 of the SemEval 2014 Competition. These algorithms were logistic regression (LR) [15], used by the winner; support vector machines (SVM) [16], used by the first and second runners-up; and stochastic gradient descent (SGD) [17], used by the fourth-placed competitor. We also compared the results with random forest (RF) [18] and naive Bayes (NB) [19].[4] The GP results are comparable to those of the SVMs, but it surpasses surpassed the SVMs considerably in the most challenging dataset in the competition. However, GP performed better than SGD in two of the datasets and better than LR, RF, and NB in many datasets. We also present a comparison between our GP results and the three best-ranked solutions submitted to Subtask A "Message Polarity Classification" of the SemEval 2016 competition [20] for each test dataset. According to our experiments, the GP results are close to the third-ranked solutions, except in the Sarcasm dataset. Our opponents used sophisticated resources, such as classifier ensembles, deep neural networks, and extra datasets, in addition to the competition benchmark used for training, with more handmade features.

Finally, another advantage of GP as a hybrid approach is that the final classifier derived (i.e., the last individual) is highly interpretable. This individual corresponds to a tree in which the internal nodes are functions and the leaves are input features. Thus, it is possible to identify the set of functions most used in the last generation, which is useful not only to know the best features but also to know how they are combined in most of the best individuals (trees). This is unlike other traditional methods, such as artificial neural networks, which are known to generate noninterpretable models.

In summary, the main contributions of this work are as follows:

a) We show, using our experiments and the results for Task 9 of the SemEval 2014 and Task 4 Subtask A of the SemEval 2016 competitions, the importance of a good choice of resources (features and lexicons) and of a good combination of them to derive a competitive hybrid SA solution.
b) We propose a new hybrid approach using GP that can automatically combine preprocessing tasks and resources to derive an effective sentiment classifier. This is a very important contribution in our opinion since the choice and combination of text preprocessing approaches, features and lexicons are usually made manually. This is a time-consuming and error-prone task when performed manually.
c) We show examples of functions that can be used to compose individuals for GP. This set of functions can be extended to support other preprocessing and handcrafted features, which makes GP an extensible tool.
d) We provide a discussion of the interpretability of the final solution created by GP.
e) We provide a comparative study of GP and other sentiment classifiers that use different machine learning methods. The results show the promising applicability of GP to SA.

The rest of this work is organized as follows. Initially, the related work is presented in Section 2. Section 3 presents an overview of GP. Section 4 shows our proposal to adapt the GP method to the SA problem. The experiments are described in Section 5, and the conclusions are presented in Section 6.

2. Related works

We briefly describe the state of the art of SA with a hybrid strategy in Section 2.1 and the use of evolutionary computing in SA from the literature in Section 2.2.

2.1. Sentiment analysis

Most sentiment analysis works use a single classifier. The most commonly used classification methods are SVM, NB, and LR [21]. Many of the analyzed papers use a hybrid strategy, combining several lexical dictionaries. These approaches have been successful, often superior to those applied in isolation ML or lexicon methods [22].

Go et al. [23] classified tweets into two classes: positive and negative. The authors used classic ML techniques, such as SVM, LR, and NB. They used unigrams and word frequency (*bag-of-words*) as features.

An approach to SA using a linear classifier trained with the stochastic gradient descent (SGD) technique can be seen in [24]. SGD was also used in [25], in which the author used the LIU,[5] MPQA[6] and NRC[7] lexicons to obtain the polarities of each word.

The research published in [26] approaches SA with a hybrid strategy using LR and seven dictionaries. As features, it uses n-grams based on words and characters, parts-of-speech (PoS), and a term disambiguator. The LR is also used in [27] with seven lexicons, namely, LIU, MPQA, NRC, SentiWordNet, Sentiment140[8] and one manually annotated, and in [28] with a

[4] Naive Bayes and Random Forest are commonly used in ML approaches to SA.
[5] https://www.cs.uic.edu/liub/FBS/sentiment-analysis.html#lexicon
[6] https://mpqa.cs.pitt.edu/lexicons/subj_lexicon/.
[7] *Multi-Perspective Question Answering*https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm.
[8] http://saifmohammad.com/Lexicons/Sentiment140-Lexicon-v0.1.zip.

lexicon created from automated documents collected from the web that were processed with pointwise mutual information (PMI) [29].

Zhu et al. [30] proposed a hybrid solution with the use of a SVM and a combination of six lexical dictionaries to classify tweets into three classes. Another hybrid approach is presented in [31], which used the lexical SentiStrength,[9] LIU and a manually constructed negation word set.

Karampatsis et al. [32] chose a 2-stage SA approach (using an SVM). In the first stage, the model classifies messages as objective or subjective. The second stage aims to classify subjective messages as positive or negative. The classifier uses several morphological features, PoS, and eleven dictionaries. SVM is also used in the research of [33], in which the author uses the MPQA lexicon [34] and special characteristics of tweets, such as retweets, hashtags, emoticons, and URLs.

Jain and Kumar [35] used SA to monitor and classify 21,040 tweets about the H1N1 flu in India. SA uses messages without stopwords, stemming, and a set of keywords as attributes. The classification process was performed using SVM, NB, and random forest (RF) approaches. As seen, the feature selection used by the classification methods is essential to obtain good results [21], and some papers are concerned with the main features and their impact on classification techniques [36–39].

2.2. Sentiment analysis using evolutionary computing

Even with the results consolidated in other areas of research, the number of papers that use evolutionary methods in SA is small [40]. Almeida [41] used genetic algorithms (GAs)[10] to identify the best combination of 17 dictionaries for a tweet sentiment classifier. In the same way, Keshavarz et al. [43] used a GA to optimize the word polarities of a lexicon with values set to vary in the range $[-10, 10] \in \mathbf{Z}$. The main objective of the work was to find a combination of polarities that minimize classifier error, using accuracy as the fitness function.

Abbasi et al. [44] used a hybrid method containing a GA for feature selection. At the same time, Basari et al. [45] applied the particle swarm optimization (PSO) method with SVM for the analysis of movie reviews posted on Twitter. Nevertheless, in the movie review context, [46] presented a hybrid solution using NB and GA.

Lopez et al. [4] used seven off-the-shelf sentiment analysis methods in an ensemble classifier for domain adaptation. Using two evolutionary algorithms – Memetic Algorithms and Differential Evolution [47] – the work sought to create a linear combination with the weights of each method defined by the evolutionary algorithm itself. The research evaluated the solution on 13 datasets, achieving the best results with a variation of the differential evolution algorithm called L-Shade (proposed by Tanabe and Fukunaga [48]).

Specifically, in the genetic programming context, we can cite the work of Graff et al. [40], which used a variation of GP called semantic genetic programming (SGP), that considered the phenotypic representation of individuals in the use of genetic operators [49]. This research proposed a new technique called root genetic programming, in which genetic operations are applied only at the tree root. The data provided by SEPLN[11] (Spanish Society for Natural Language Processing) were used as the benchmark, and the main objective was to classify messages into six classes, adopting a one-to-one strategy [50,51]. The representation of the input was performed using a numerical vector of the message created using TF-IDF (Frequency–Inverse Document Frequency).

We note that our research differs from existing works in several aspects: the use of genetic programming defined in [52] and without modifications of the main genetic operators or restrictions on crossover. In addition, the representation of the message is made in its original textual format, rather than its conversion into a numerical vector.

This last characteristic directly reflects the set of functions used for the manipulation of individuals. In our research, we did not find papers that use this type of representation for SA that used GP. It is also possible to highlight the difference in the classification strategy: the present work creates a single ternary classifier (positive, negative, and neutral), whereas the research published in [40] uses 15 binary classifiers for the evaluation of the six available classes. The type of selection and the way of creating individuals are the same in both papers: a tournament with $k = 2$ and ramped half-and-half, respectively.

To summarize the data used in related works, relevant information from leading studies is organized in tables. In Table 1, the lexical dictionaries used by the main papers are presented. Likewise, the main attributes used in SA are presented in Table 2.

3. Genetic programming

GP is an evolutionary technique that aims to obtain a good population of computer programs (or solutions) to solve a given problem. GP starts with an initial population of programs, usually generated randomly. In each population, individuals satisfying a fitness criterion (i.e., best individuals) are chosen to form the next population. Genetic operations such as crossover, mutation, and replication are applied to the best individual of a population to generate the population of the next gen-

[9] http://sentistrength.wlv.ac.uk/.

[10] We remind the reader that in this work we use Genetic Programming and not Genetic Algorithms. For more information on the difference between the methods see [42].

[11] www.sepln.org/workshops/tass/2015/tass2015.php.

Table 1
Lexicons used in related works. NRC_e means an emoticon NRC lexicon, NRC_h means a hashtag NRC lexicon, SWordnet means the SentiWordNet lexicon and S140 means the Sentiment140 lexicon.

	Lexicon							
	LIU	MPQA	NRC_e	NRC_h	SWordnet	AFINN	S140	Own Lexicon
[30]	✓	✓	✓	✓	✓	✓		✓
[31]	✓				✓			
[53]								✓
[28]			✓		✓		✓	✓
[26]	✓	✓	✓	✓	✓	✓	✓	
[32]	✓	✓	✓	✓	✓	✓	✓	✓
[54]	✓	✓	✓	✓		✓	✓	✓
[55]	✓	✓				✓		✓
[56]	✓		✓	✓			✓	
[57]	✓	✓	✓	✓			✓	
[25]	✓	✓		✓				
[58]	✓	✓	✓	✓			✓	✓
[59]	✓	✓		✓		✓	✓	✓
[31]	✓							
[60]	✓	✓		✓				✓
[61]	✓	✓	✓					✓
[41]	✓		✓		✓		✓	
[62]	✓	✓	✓				✓	✓
[27]	✓	✓		✓	✓		✓	✓
[63]					✓			

Table 2
Features used in related works. PoS means Part-of-speech, Neg means negation, Intens means intensification, Pont means punctuation and Rep means repetition.

	Feature							
	PoS	Neg	Intens	N-gram	Pont	Rep	Upper	Lexicon
[30]	✓	✓	✓	✓	✓		✓	✓
[31]	✓	✓		✓	✓	✓	✓	✓
[53]	✓				✓			✓
[28]	✓	✓		✓	✓	✓	✓	✓
[26]	✓	✓		✓				✓
[32]	✓		✓	✓	✓	✓	✓	✓
[54]	✓	✓		✓	✓	✓	✓	✓
[55]	✓	✓		✓	✓	✓	✓	✓
[56]	✓	✓	✓	✓	✓	✓		✓
[57]	✓	✓		✓				✓
[25]	✓	✓	✓	✓	✓	✓	✓	✓
[58]		✓		✓	✓	✓	✓	✓
[59]		✓		✓				✓
[31]	✓	✓				✓	✓	✓
[60]		✓						✓
[61]	✓	✓		✓	✓	✓	✓	✓
[41]		✓		✓				✓
[62]	✓	✓		✓	✓	✓	✓	✓
[27]	✓	✓				✓	✓	✓
[63]				✓	✓	✓		✓

eration. This process is repeated until a stop condition is satisfied. A criterion is then used to choose the unique program of the last population as the definite solution to the problem [52,14].

For many practical problems, programs in GP are represented as *syntax trees* rather than as lines of code [14]. A syntax tree is a tree in which the root and all the internal nodes correspond to functions or operators. Internal nodes have one or more children that can be other internal nodes (functions) or terminals. The output of a function (internal node) x is used as the input of the function in the parent node of x, except when x is the root of the hierarchy. In this case, the output of the root function corresponds to the output of the complete program. Terminals are entities that represent a value or an object and have no child.

Fig. 1 shows an example of a program used for the problem we focus on in this work, that of labeling a text as positive or negative. The dark gray nodes are terminal nodes, and the while light gray nodes represent functions. Both the functions and terminals in Fig. 1 are explained in the next section.

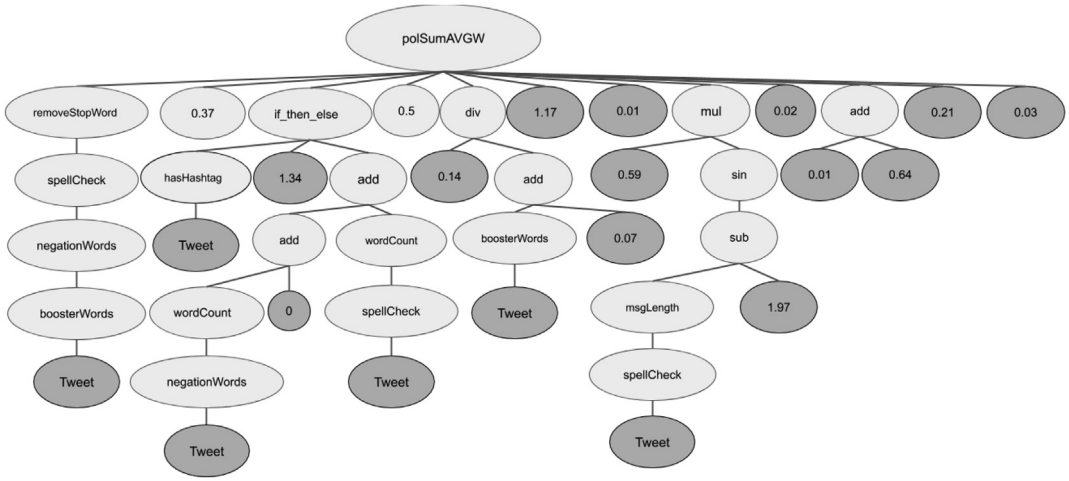

Fig. 1. Example of an individual.

To apply GP to a problem, several decisions need to be made, and they are often termed *the preparatory steps* [14], which are:

- Terminal set definition – This is the definition of a finite set of data objects that the functions composing the program will operate on to derive an output. Usually, there are three types of terminals: a) input-related terminals (numbers or strings composing the input data), b) constants or c) a special function without input parameters that generates data not related to the input.[12]
- Function set definition – The definition of a finite set of functions or operators that are used to compose the program. These functions have as many input parameters as the number of children of the node corresponding to it in a syntax tree.
- Fitness function definition – A function used to evaluate how good a program is at solving the problem. This is an important component because only programs that satisfy the fitness function will survive to "breed" new programs to form a new population.
- Algorithm control parameters – The definition of the control parameters for running GP. The most relevant parameters include population size, probability of performing genetic operations, and maximum height of each tree.
- Termination criterion and the final solution – The termination criterion may include a maximum number of generations to be run as well as a problem-specific success predicate. The final solution (final program) is usually the best solution found in the last population of the program.

Once the preparatory steps have been performed, the rest of the GP process is completely automatic. However, the success of obtaining a good final population of programs is strongly dependent on the preparatory steps. The steps must be defined specifically for the target problem. In the next section, we show our proposal to define the preparatory steps for the problem of sentiment analysis.

4. Genetic programming applied to sentiment analysis

In this section, we present three of the preparatory steps used to adapt GP for the sentiment analysis problem. Specifically, we present the terminal set, the function set, and the fitness function we propose to use with GP. The GP control parameters are discussed in the experimental setup (Section 5.2) since some of them are defined experimentally. Finally, as the last component of the preparatory step, the number of generations is the criterion adopted to interrupt the evolutive processing of GP. The final program used as the definite sentiment classifier is the program with the best fitness value among the programs of the last population.

4.1. Terminals and functions

Since the aim is to use GP to classify sentiment in texts, it is essential to define terminals and functions that allow for the manipulation of this type of information. Thus, raw texts are used as terminals of the programs used in our GP settings. In addition, numerical constants are also used as terminals of mathematical functions. Thee constants are introduced as leaves of the syntax trees; they are randomly created to form the initial population of programs. The values of these constants are

[12] For instance, a function that generates a random number.

Table 3
Main GP functions.

ID	Function	Description	Input type	Output type
1	polSum(m)	Sum the polarity of words in m, using all lexicons	String	Float
2	polSumAVG(m)	Arithmetic mean of word polarities in m,using all lexicons	String	Float
3	polSumAVGW(m, [$w_1 \ldots w_n$])	Weighted arithmetic mean of word polarities	String, n Floats	Float
4	hashtagPolSum(m)	Hashtags polarity sum	String	Float
5	emoticonPolSum(m)	Emoticons polarity sum	String	Float
6	hasHashtags(m)	Check for hashtags in m	String	Boolean
7	hasEmoticons(m)	Check for emoticons in m	String	Boolean
8	hasURL(m)	Check if the message msg. has URLs	String	Boolean
9	hasDate(m)	Check for dates in m	String	Boolean
10	if_then_else(exp, c1, c2)	If exp true then c1 else c2	Bool., Float	$type(c_1) \| \| type(c_2)$
11	removeStopwords(m)	Remove the stopwords from the message msg	String	String
12	removeAllPonctuaction(m)	Remove all ponctuaction from the message msg	String	String
13	removeLinks(m)	Remove all links in m	String	String
14	spellCheck(m)	Message Spell Check	String	String
15	neutralRange(IR, SR)	Range values for neutral class	Float, Float	None
16	negationWords(m)	Check for negation words	String	String
17	boosterWords(m)	Check booster words	String	String
18	upperCaseWords(m)	Check for uppercase words	String	String
19	wordCount(m)	Word quantity	String	Int
20	msgLength(m)	Message length	String	Int
21	positiveHashtags(m)	# positive hashtags in m	String	Int
22	negativeHashtags(m)	# negative hashtags in m	String	Int
23	add, sub, mul, div, sen, cos	Math functions	String	Float

generated randomly in the interval $[-2, 2]$. These constants are referred to as δ. In summary, the set $\mathscr{T}$ of terminals is defined as:

$$\mathscr{T} = \{\text{text}, \delta\}, -2 \leqslant \delta \leqslant 2 \tag{1}$$

As stated in Section 1, many resources (handcrafted features, textual features, and lexicons) can be combined in different ways to enhance text sentiment classification. Below, we present the definition of the function sets tailored to this resource combination and to allow for generating programs (syntax trees) that are able to classify the sentiment of input texts. These functions are shown in Table 3 and are grouped into categories according to their similarity regarding the operation they perform. To describe the functions, an input textual message $m = \langle w_1, \ldots, w_{|m|} \rangle$, where $w_j, 1 \leqslant j \leqslant |m|$ are the words composing m, $|m|$ denotes the number of words in m, and the set $\mathscr{L} = \{L_1, L_2, \ldots L_{|\mathscr{L}|}\}$ of lexicons are considered.

4.1.1. Message transformation functions

The message transformation functions receive a text m as an input parameter and generate a new text by applying some type of transformation on m. The functions in this group include:

- *removeStopwords*(m) – generates a new text from m by removing the stop-words[13] contained in text m.
- *removeAllPucntuation*(m) – generates a new text from m by removing all the punctuations in m.
- *removeLinks*(m) – generates a new text from m by removing all the URLs in m.
- *spellCheck*(m) – generates a new text from m by correcting misspelled words in m.
- *negationWords*(m) – generates a new text from m by appending the tag "_neg" to negation words found (ex.: *don't, no, doesn't, aren't, etc.*) in m. For instance, if m = "I'm not happy.", then the generated text is $m\prime$ = "I'm not_neg happy."
- *boosterWords*(m) – generates a new text from m by appending the tag "_boost" to boosting words (ex.: *very, much, deep, etc.*) in m. For instance, if m = "I'm very happy.", then the generated text is $m\prime$ = "I'm very_boost happy.'
- *upperCaseWords*(m) – generates a new text from m by appending the tag "_up" to words with all letters in upper case.

Functions *removeStopwords*() and *spellCheck*() allow GP to choose if and when text preprocessing must be applied to the texts of a dataset. On the other hand, functions *negationWords*(m), *boosterWords*(m) and function *upperCaseWords*(m) work as polarity modifiers of words in a text. The tags generated by these functions are used by the polarity function group to compute the polarity values of a text, as discussed below.

4.1.2. Polarity functions

The following polarity functions over the input message m are defined:

[13] In the experiments described this work we used the stop-word list of the NLTK library (available inhttps://www.nltk.org/) [64].

$$polSum(m) = \sum_{i=1}^{|\mathscr{L}|}\sum_{j=1}^{|m|} pol(w_j, L_i),$$

$$polSumAVG(m) = \frac{1}{|\mathscr{L}|}\sum_{i=1}^{|\mathscr{L}|}\sum_{j=1}^{|m|} pol(w_j, L_i),$$

$$polSumAVGW(m, w_1, \cdots, w_{|\mathscr{L}|}) = \frac{1}{\sum_{i=1}^{|\mathscr{L}|} w_i} \times \sum_{i=1}^{|\mathscr{L}|}\sum_{j=1}^{|m|} pol(w_j, L_i) \times w_i,$$

where function $pol(w, L)$ returns the polarity value of a word w in the lexicon L. Function $polSum(m)$ accumulates the polarities of all the words of m found in all lexicons. Function $polSumAVG(m)$ first computes the polarity values for a message m by summing the polarities found for the words of m in a given lexicon L_i (internal summation). A distinct polarity for m is computed using each lexicon in $\mathscr{L}$. Then, the function computes the average of the $|\mathscr{L}|$ message polarity values obtained. Function $polSumAVGW(msg, w_1, \cdots, w_{|\mathscr{L}|})$ computes the weighted average of the polarities of the words of m. This function receives the weights of the lexicons as input parameters. As the root of the program in Fig. 1 shows, the value of each parameter can be a constant (terminal node) or the result of another function (internal node). As a consequence, the weights of each lexicon can be automatically computed by GP during the evolution process. This function is very important because it allows GP to combine different lexicons by adjusting the importance of each of them (through dynamic weight computation) to a given dataset.

Whenever any of the functions above find a word with the tag "_neg" appended to it, the function inverts the polarity of the next word in the text. If any of the functions find the boosting tag "_boost" appended to a word w, then it duplicates the absolute value of the polarity of the next word in the text. Similarly, if a word is written with all letters in upper case, then the polarity functions duplicate the absolute value of the polarity of this word found in a given lexicon.

Two other functions that address the polarities of hashtags and emoticons are included in this category of functions:

$$hashtagPolSum(m) = \sum_{i=1}^{|\mathscr{L}|}\sum_{j=1}^{|m|} ishashtag(w_j) \times pol(w_j, L_i),$$

$$emoticonPolSum(m) = \sum_{j=1}^{|m|} isemoticon(w_j) \times pol(w_j, L_{emoticon}),$$

where the auxiliary functions $ishashtag(w)$ and $isemoticon(w)$ return one if w is a hashtag or an emoticon, respectively, and return zero otherwise. In the specific case of function $emoticonPolSum(m)$, only a specific emoticon lexicon $L_{emoticon}$[14] is used.

4.1.3. Checking functions

A group of Boolean functions is defined to check the existence of some components in the input text m. These functions are $hasHashtags(m)$, $hasEmoticons(m)$, $hasURL(m)$, and $hasDate(m)$; these functions check if there is at least one of the following components in the input text m: hashtag, emoticon, URL, and date, respectively. The functions return true if the corresponding component is found; otherwise, the functions return false.

4.1.4. Counting functions

This group includes the functions that perform some kind of counting in input text m:

- $wordCount(m)$ – returns the number of words in m.
- $msgLength(m)$ – returns the number of characters in m.
- $negativeHashtags(m)$ returns the number of negative hashtags in m.
- $negativeEmoticons(m)$ – returns the number negative emoticons[15] in m.

4.1.5. Math functions

Since many of the above functions return numeric values, math functions are also included in the function set of GP for sentiment analysis. These functions may be especially useful for computing the weights of lexicons, which are used as parameters of function $polSumAVGW()$. The math functions used are the following: $add(x, y)$, $sub(x, y)$, $mul(x, y)$, $div(x, y)$, $sin(x)$ and $cos(x)$. The first four functions correspond to the four basic arithmetic operators: $+, -, \times$ and $\div$, respectively. The last two functions correspond to the *sine* and *cosine* trigonometric functions.

[14] In this work, we use the NRC [65] emoticon lexicon as $L_{emoticon}$.

[15] As in function $emoticonPolSum()$, the polarity of the emoticon is verified using $L_{emoticon}$.

4.1.6. Conditional function

The conditional function *if_then_else*(exp, n_1, n_2) defines a choice between two branches in the syntax tree corresponding to a program. This function has three parameters: a Boolean expression *exp* and two subtree roots, n_1 and n_2. If *exp* is evaluated as true, then node n_1 is returned as the output of the function; otherwise, node n_2 is returned as the output of the function.

For instance, refer to the syntax tree shown in Fig. 1. The third child of the root node is a node corresponding to the function *if_then_else*(), which means that the value of this node will be 1.34 if *hasHashtags*(*tweet*) is true; otherwise, the value of this node will be equal to the result of its second rightmost child, node *add*. Therefore, the value of node *if_then_else*() in this example is a real number because it corresponds to the weight of the second lexicon used by function *polSumAVGW* positioned at the root of the tree.

4.1.7. Neutral range function

A program always outputs a value $r, r \in \mathbb{R}$. It is necessary to convert this value to a label (*neutral*, *negative*, or *positive*) to classify the input text. A common approach adopted by many solutions [43,66] is to label the input as neutral if $r = 0$, as negative if $r < 0$, or as positive if $r > 0$. However, we found, through experimentation, that for some datasets, the decision based on a zero value only is too restrictive. The final model generated by GP may be more accurate if a range of values can be used to decide the polarity of the input text.

Function *neutralRange*(IT, ST) is proposed to introduce more flexibility for the classification of a text. This function computes the values of two thresholds: *IT*, the inferior threshold, and *ST*, the superior threshold of the *neutral range* $[IT, ST]$. Values falling in this range indicate that the input text is neutral. *IT* and *ST* are two global variables, both initialized to zero before the generation of a program.

4.2. Labeling an input text

Let p be a program of GP. Let m be an input text. Let $p(m) \in \mathbb{R}$ be the result of program p after processing m. Function *label*(p, m, IT, ST)maps the result $p(m)$ of a GP program into a label (*negative*, *neutral* or *positive*) for m as follows:

$$label(p, m, IT, ST) = \begin{cases} positive, & p(m) > ST \\ neutral, & IT \leqslant p(m) \leqslant ST \\ negative, & p(m) < IT \end{cases}$$

Note that this labeling function is not part of the function set used by GP. This function set is not used as internal nodes in the syntax trees; it is an external function used only to convert the output of a program into a label.

4.3. The fitness function

The fitness evaluation function is a criterion used to select the best fit individuals for breeding the next generation. Immediately after a population is generated or after the initial random population is created, a fitness criterion is applied to all the individuals (programs in the case of GP) to determine the best individual of that population. The best individuals are those to which the genetic operations (crossover, mutation, etc.) will be applied [52].

Since the programs created by GP for SA are classifiers, we opted to use a measure of classification effectiveness as the fitness function. We used the *arithmetic mean of the positive and negative F1-scores* as the fitness function. We adopted this measure because it is the same measure used in the SemEval2014 and SemEval2016 competitions [67,20], and we wanted to compare our results with the best results of those competitions.[16]

Positive F1 ($F1_{pos}$) is defined as $F1_{pos} = \frac{2(p_{pos}+r_{pos})}{p_{pos} \times r_{pos}}$, where p_{pos} is the *positive precision* and r_{pos} is the *positive recall*. p_{pos} is defined as the number of texts that the program correctly predicted as positive divided by the total number the program predicted to be positive. Positive recall, r_{pos}, is the number of messages correctly predicted to be positive, divided by the total number of positive texts in the gold standard. Negative F1 is computed in a similar way after first computing *negative precision* (p_{neg}) and *negative recall* (r_{neg}).

5. Experimental methodology

In this section, we present experiments comparing GP adapted to SA with classical techniques of machine learning: support vector machines (SVM) [16,68] (using a linear kernel), naive Bayes (NB) [19], logistic regression (LR) [15], random forest (RF) [18] and stochastic gradient descent (SGD) [17]. Models using these techniques were generated with the support of the Scikit-learn library.[17]

[16] The metric for evaluating the participants' systems in SemEval2014 and SemEval 2016 was the average F-measure (averaged F-positive and F-negative, and ignoring F-neutral); note that this does not make the task binary [67,20].

[17] http://scikit-learn.org/.

We also compare the GP results to the best solutions in SemEval 2014 and SemEval 2016. Below, we first present the dataset used in the experiments, the experimental setup, and the evaluation of the experiments.

5.1. Dataset

The experiments were performed on representative datasets provided for Task 9 at SemEval 2014[18] — *International Workshop on Semantic Evaluation* [67] and Subtask A "Message Polarity Classification" of the SemEval 2016[19] Task 4 competition [20]. The training set has 9.684 messages — classified as positive, negative, and neutral — and it is organized as shown in Table 4. The test set contains 8.987 messages, divided into five subsets, as shown in Table 5. Task 9 at SemEval is a competition, and the organization used the average F1-score of positive and negative messages to rank the submitted approaches.

5.2. Experimental setup

5.2.1. Control parameters

We conducted a grid search to choose the main parameters of the machine learning algorithms we used to compare with our GP solution. We applied a grid search with 5-fold cross-validation in the training set only. The best parameters found are shown below.

- SVM: linear kernel, $c = 0.005$;
- RF: $n\_estimators = 400, max\_depth = 35$;
- SGD: $alpha = 0.001, penalty = default$;
- LR: $n\_estimators = 100$;

For GP, the general parameters used are presented in Table 6. The parameters were chosen through empirical tests using the training set only. Some authors argue that the best configuration for most problems has a small number of generations (often 51) and a substantially larger population [52,14]. However, for the problem of SA, tests show that using a number of generations of approximately 150 could result in better models.

The probabilities of crossing over and mutation were defined as 90% and 5%, respectively. These values are suggested in the literature [52,14] and were also confirmed in our tests.

The choice of the tournament as a selection method aims to prevent the best individuals from monopolizing the process, especially after a few generations [14]. Due to its function, the technique prevents this problem from occurring, helping to maintain the genetic diversity of the population. The value of parameter k for the tournament – number of individuals selected for each round – was defined as two after empirical tests and after consulting the literature.

The creation of the GP population is performed using the half-and-half method because it is the most commonly used technique in the literature [52,14] and generates heterogeneous individuals, supporting the diversity of the initial population.

Two stopping criteria were defined: the maximum number of generations obtained and fitness function with 100% of the averaged F1. Both of these criteria cause the evolution process to stop, and the best individual is returned.

5.2.2. Lexicons

Table 7 shows the set $\mathscr{L}$ of lexicons used by the polarity functions of the GP we propose. The choice of these lexicons was based on the related works discussed in Section 2.

As shown in Table 7, not all the lexicons have the same outputs; however, in all of them, it is possible to identify the polarity of the words. In this work, the polarity of a word, not its intensity, is taken into account. Thus, the sentiment value v of a word in any dictionary is converted to -1 if $v < 0$, to 1 if $v > 0$ and is maintained as zero if $v = 0$. With this conversion, we standardize the values of word polarities in the different lexicons.

Despite the importance of lexicons for sentiment classification, the simple inclusion of more lexicons in the proposed solution does not necessarily improve the final performance. Depending on how the model assigns polarity to messages, the results may even worsen [69].

In this work we let GP decide by itself the weight to be assigned to each of the eleven lexicons by means of function *polSumAVGW*(). In the experiments, if this function is used in the programs, then the correspondence between each function's weight parameters and the lexicons is shown in the last column of Table 7.

5.2.3. Final program selection

We executed our proposed GP solution thirty times because each run produces a different model, given the randomness inherent in GP. We then chose the best individual among the individuals returned by the thirty executions of the algorithm as our GP-generated classifier and used this classifier to compare with the other methods.

[18] http://alt.qcri.org/semeval2014/task9/.
[19] http://alt.qcri.org/semeval2016/task4/.

Table 4
Training dataset.

Polarity	Messages	% of total
Positive	3640	38%
Negative	1458	15%
Neutral	4586	47%
TOTAL	9684	100%

Table 5
Test dataset.

Dataset	Messages	% Positive	% Negative	% Neutral
Twitter2013	3813	41%	16%	**43%**
Twitter2014	1853	**53%**	11%	36%
Sarcasm	86	38%	**47%**	15%
SMS	2093	24%	19%	**57%**
LiveJournal	1142	**37%**	27%	36%
TOTAL	8987	39%	17%	**44%**

Table 6
Genetic programming parameters.

Parameter		Value
Crossover prob.		90%
Mutation prob.		5%
Population		250
Generations		150
Fitness		F1-score
Initial population		
	Type	Ramped half and half
	Tree height	Between 1 and 3
Selection method		
	Type	Tournament
	Size (k)	2
Stop condition		# generations
Tree max height		15

Table 7
Lexicons.

	Words				
Lexicon	Positives	Negatives	Total	Output	Attribute
LIU [70]	2006	4801	6807	positive/negative	w_1
SentiWordNet [71]	15439	16908	32347	$[-1,+1] \in \mathbb{R}$	w_2
AFINN [72]	877	1599	2476	$[-5,+5] \in \mathbb{Z}$	w_3
Vader [73]	3300	4143	7443	$[-4,+4] \in \mathbb{Z}$	w_4
Slang [74]	15298	48827	64125	$[-2,+2] \in \mathbb{Z}$	w_5
Effect [75]	3298	2427	5725	positive/negative	w_6
SemEval2015 [76]	600	330	930	$]-1,+1[\in \mathbb{R}$	w_7
NRC [65]	2312	3324	5636	$[-1,+1] \in \mathbb{Z}$	w_8
General Inquirer [77]	1914	2292	4206	$[-1,+1] \in \mathbb{Z}$	w_9
Sentiment140 [78]	38312	24336	62648	$[-5,+5] \in \mathbb{R}$	w_{10}
MPQA SL [34]	2718	4912	7630	$[-1,+1] \in \mathbb{Z}$	w_{11}
TOTAL	67752	90591	158343		

5.2.4. Features

The features used in the classic machine learning algorithms chosen for comparison with the genetic programming technique are described in Table 8.

In genetic programming, due to its mode of operation, the same features were used but in the form of functions, as shown in Table 3.

Table 8
Features.

Feature	Description
Negation	Appends a negation suffix to words that appear within a negation scope using a negative word. The words were detected using a negation word dictionary.
Intensification	Appends an intensification suffix to words that appear within an intensification scope using a booster word. The words were detected using a booster word dictionary.
Emoticon	The emoticons were detected using a emoticon dictionary NRC_e.
Hashtag	Hashtags were detected by the character "#" at the beginning of the string.
Date	Dates were detected using regular expression.
URL	URLs were detected using regular expressions.
Spelling	Included in the system to normalize misspellings.
Stopwords	Stopwords are words that are filtered out before or after processing of text. Stopwords refer to the most common words in a language, and in our case we used a default stopword set from NLTK [64].
Word count	Number of word of the string.
Emoticon polarity sum	Number of positive emoticons minus number of negative emoticons.
Hashtag polarity sum	Number of positive hashtags minus number of negative hashtags.
Message length	Number of characters in the string.
Polarity sum	Number of positive words minus number of negative words.
Avg polarity	(Number of positive words)/ number of dictionaries minus (number of negative words.)/ number of dictionaries

5.3. Evaluation

When comparing each method with the GP-generated classifier, we apply the paired T-test to determine whether the differences between the results are statistically significant, with a confidence level set at 95%.

The results obtained using the GP are presented in Table 9. The highest values of each metric are in bold, and the worst results are underlined to make reading easier. Hereafter, we use F1-P/N as the arithmetic mean of the positive and negative F1-scores refer, respectively (as defined in Section 4.3). Separate rankings for each subset were produced, as was done in SemEval2014 [67].

The LiveJournal subset achieved the best results in 4 out of 5 metrics (except recall). This set contains messages with less incidence of slang and informal words, reinforcing the difficulty in dealing with texts that present colloquial language, which is a characteristic of social networks. These messages presented the most stable results when considering all of the metrics.

The lowest results for all the metrics were obtained from the subset containing sarcastic entries (Sarcasm), which attests to the difficulty in handling messages with this type of language expression; this is considered one of the open challenges of sentiment analysis [79,13]. It is important to note that in the study published in [80], the author reports that the ability to detect human sarcasm reached an accuracy of only 62.59%, very close to the value resulting from the best classifiers of the benchmark. Finally, it should be noted that there is an even higher level of difficulty in including the neutral class in the detection and classification of sarcastic messages.

Table 10 presents the results in terms of F1-P/N for each method. The asterisk symbol indicates that the difference between the means of the ML technique and the GP is statistically significant.

The GP results are consistently better than those of RF in all the subsets. GP is superior to NB in four of the five subsets and is only inferior in the Sarcasm subset, for which NB is the best. GP is also better than LR in three out of five subsets and is equal to it in the other subsets. SGD is superior to GP only in the SMS subset and is worse or equal to it in the other subsets. Finally, GP and SVM present very similar results.

When analyzing the results for all the test messages, GP is better than RF, NB and LR and is equal to SVM and SGD.

The results presented in Table 10 show that our proposed GP is superior to many methods in many subsets and is equivalent to the considered good machine learning methods, such as SVM and SGD. However, GP can be easily extended by introducing other functions in its function set. Many other features, such as part-of-speech-based features and n-grams, can be used with GP. Thus, we consider GP to be a promising tool for SA.

Table 9
GP Results.

	Average			
Dataset	Accuracy	Precision	Recall	F1-P/N
Twitter2013	65.59	64.53 ±0.13	67.66 ±0.13	65.47 ±0.07
Twitter2014	63.25	58.28 ±0.14	64.12 ±0.13	62.31 ±0.10
Sarcasm	<u>48.84</u>	<u>49.43</u> ±0.09	<u>49.8</u> ±0.20	<u>48.04</u> ±0.10
SMS	65.6	64.26 ±0.17	**70.85** ±0.10	62.6 ±0.01
LiveJournal	**69.44**	**70.07** ±0.07	70.31 ±0.10	**71.24** ±0.01
ALL	65.44	64.71 ±0.12	67.8 ±0.11	65.5 ±0.06

Table 10
Comparison of the main results of the ML techniques and GP (F1-P/N).

	F1-P/N					
Dataset	RF	SVM	NB	LR	SGD	GP
Twitter2013	50.47*	65.39	52.84*	61.61*	59.7*	**65.47**
Twitter2014	46.9*	**63.94**	51.24*	63.8	60.4	62.31
Sarcasm	41.43*	41.06*	**54.25***	49	39.11*	48.04
SMS	42.63*	61.35	43.1*	50.5*	**64.77***	62.6
LiveJournal	53.1*	69.02	54.61*	61.12*	67.41	**71.24**
ALL	50.58*	65.07	51.2*	60.97*	63.33	**65.5**

We also conducted another experiment to better understand what functions were most used by the best individuals generated by GP. To this end, we collected the best individuals returned by the 30 executions of GP and counted the number of times each function was used in each individual. The ranking of the most used functions is shown in Table 11.

We note that function *polSumAVGW*, which computes the average of the polarities found in lexicons, takes the weights of lexicons into consideration as the most important function. In fact, as we discuss in Section 5.5, there is a small set of lexicons that are very important to polarity detection. Other preprocessing functions are also used, such as those removing stopwords and links, followed by functions related to sentiment detection.

5.4. Comparison with SemEval best solutions

In this section, we demonstrate the importance of SA in choosing good preprocessing and handcrafted feature lexicons and in combining all these resources to build a sentiment classifier. We present the results from Task 9 of the SemEval 2014 Competition [67] and Task 4 with Subtask A "Message Polarity Classification" of the SemEval 2016 competition [20] to this end. In these tasks, competitors were given the training set discussed in Section 5.1, but they did not have access to the test set. Competitors were allowed to use any feature they wanted, and they could even use other textual corpora to derive their model. Competitors could also apply any lexicon or even build their own.

Table 12 shows the features and lexicons used by the four best solutions for Task 9 of SemEval 2014. All the solutions used a large number of features. Some of these features were very sophisticated. For instance, the second-placed competitor used deep learning to derive sentiment-specific word embeddings. The other two solutions used part-of-speech (PoS) tagging to derive new features. All of the solutions used more than one type of n-gram, with some of them using both word and character n-grams. Additionally, the first solution used seven distinct dictionaries, and the third one used two customized lexicons, which shows the importance of using more than one lexicon.

Our experiment described in Section 5.3 cannot be directly compared to Task 9 of the SemEval 2014 or Task 4 Subtask A of SemEval 2016 competitions. The former had the intention to make a fair comparison among learning methods, whereas the latter is a competition in which the contenders could use a "no holds barred" style to win. However, the three experiments allow for some observations to be derived regarding training a classifier for the SA task. The first is that SA is too sensitive to resource choice. Note that our experiment described in Section 5.3 shows that SVM is superior to LR when submitted to the same set of features; however, the best solution in the competition used LR as the learning method. Thus, by performing a better selection and combination of resources, the winner was able to surpass the competitors that used superior learning algorithms, such as SVM.

Another observation is that the number of resources that can be used in SA is enormous, and the competitors participating in Task 9 of the SemEval 2014 and Task 4 of the SemEval 2016 must have made a great effort to choose and combine the resources they used. Since none of them commented on how they chose and combined all the resources they used in the articles describing their solutions, we conclude that this was done by all of them using a trial-and-error approach. We

Table 11
Most used functions in the individuals obtained after 30 GP executions.

Rank	Function
1	polSumAVGW
2	removeAllPonctuation
3	removeLinks
4	negativeEmoticons
5	hasEmoticons
6	negativeHashtags
7	boostUpperCase
8	removeStopWords
9	negationWords

Table 12
Resources used by the best solutions in SemEval 2014 competition.

1st Ranked Solution [26]	
Learning Method:	Logistic Regression
Lexicons:	AFFIN, LIU, General Inquirer, MPQA, NRC_h, S140, SWordnet
Features:	Spelling correction, PoS tagging, word sense disambiguation, negation detection, word n-grams continuous(1–4) and non continuous(3–4), character n-grams(3–4), word clusters
2nd Ranked Solution [58]	
Learning Method:	SVM
Lexicons:	LIU, MPQA, NRC_e, NRC_h, S140
Features:	Sentiment-specific word embedding features, all caps, emoticons, #enlongated words, #individual negations, #contiguous punctuation, cluster words, word n-grams(1–4), character n-grams(3–5)
3rdRanked Solution [30]	
Learning Method:	SVM
Lexicons:	Two customized lexicons derived from Yelp and Amazon corpora
Features:	Many handcraft features using PoS and context word unigrams and bigrams

consider that GP can be used to alleviate this effort. We proposed in this paper many functions that allow GP to deal with features and different lexicons as a unified solution. The set or our functions can be extended to address most of the features used by the three best solutions for Task 9. Thus, GP can be trained to choose (via the evolutionary process) the best combination of features and lexicons to be used. Consequently, GP can be used to alleviate the great manual/mental effort that is still necessary to derive good features and lexicons for the SA task.

Comparing the approaches submitted for Task 9 of SemEval 2014 with GP trained with the reduced (and much simpler) set of features in Section 5.2.4 is not fair. However, we wanted to know how GP perfumed even when using limited resources. Table 13 shows the results for the three best-ranked solutions submitted to Task 9 of the SemEval 2014 competition for each test dataset. We copied the GP results for these datasets to the second column of the table to facilitate comparison. The third column in the table is the result of the trivial classification of assigning the label of the majority class found in the training set to each test instance. This column was added to show that all the solutions are better than the trivial solution.

Table 13 shows that, except in the Sarcasm dataset, the GP results are only slightly inferior to the third-ranked solution, despite using considerably fewer and simpler features. We observed that this similarity is achieved mainly due to the appropriate weight values that GP assigns to each lexicon. We arrived at this hypothesis because function *polSumAVGW*(), which deals with the weights of lexicons, appears as the root function in all the individuals of the last generation and in the best individual. We discuss these lexicon-related aspects further in the next section.

In the same way, we present in Table 14 a comparison between our GP results and the three best-ranked solutions submitted for Subtask A "Message Polarity Classification' of the SemEval 2016 Task 4 competition [20] for each test dataset. We copied the GP results in these datasets in the second column of the table to facilitate comparison. According to this table, it is possible to verify that the GP results are only slightly inferior to the third-ranked solutions, except in the Sarcasm dataset. This finding is surprising because these solutions apply sophisticated resources, such as classifier ensembles or deep neural networks (such as CNNs and FFNNs), and include extra datasets in addition to the competition official benchmark being used for training or more handmade features. On the other hand, our GP approach uses only the information contained in the training set and a few simple features. Thus, these comparisons with the best competing solutions for both SemEval 2014 Task 9 and SemEval 2016 Subtask A show that GP is a promising approach for sentiment classification.

5.5. Combining lexicons and class range

As discussed earlier, one benefit of applying genetic programming as a machine learning technique is the possibility of reading and interpreting the generated models. In this way, it is possible to analyze the results to understand the solution process [89]. Due to how it generates and evaluates the models, GP somehow performs feature selection, determining the most relevant functions to the problem in which it is being applied. Based on the abovementioned findings, an analysis of the results obtained with the experiments can show the functions and characteristics considered most significant for the benchmark evaluation by GP.

The verification of the lexicons included in the solution and the importance given to them by GP (through the weights) can reveal the most appropriate lexicons to the problem (domain) in question, that is, those that gave the greatest benefit to the evaluation process for the benchmark.

Fig. 2 presents the average values of the weights of each lexicon. Note that these values represent the mean weights assigned to each lexicon in the 30 GP models. The lexicons Effect (w_6) and NRC (w_8) received the lowest values in all the models, with

Table 13
Comparison of the GP results with the studies submitted for SemEval 2014 (F1-P/N score).

Dataset	GP result	Majority Baseline	Top 3 SemEval
Twitter2013	65.47	29.2	1º 72.12 2º 70.75 3º 70.40
Twitter2014	62.31	34.6	1º 70.96 2º 70.14 3º 69.95
Sarcasm	48.04	27.7	1º 58.16 2º 57.26 3º 56.50
SMS	62.6	19	1º 70.28 2º 67.68 3º 67.51
LiveJournal	71.24	27.2	1º 74.84 2º 74.46 3º 73.99

Table 14
Comparison of the GP results with the solutions submitted for SemEval 2016 (F1-P/N score) — CNN means Convolutional Neural Networks, FFNN means Feed-Forward Neural Network, LR means Logistic Regression, and TM means Topic Modeling.

Dataset	GP result	Top 3 SemEval	Features	Learning Algorithm
Twitter2013	65.47	1º 72.30 2º 71.40 3º 70.60	Specific embeddings[1] Handcrafted features[2] Specific embeddings[3]	FFNN [81] LR [82] CNN [83]
Twitter2014	62.31	1º 74.40 2º 72.70 3º 72.30	Specific embeddings3 Specific embeddings1 Handcrafted features2	CNN [83] FFNN [81] LR [82]
Sarcasm	48.04	1º 56.61 2º 55.42 3º 54.00	Word2Vec [84]/ Glove [85] Specific embeddings1 Word2Vec [84]	CNN Ensemble [86] FFNN [81] CNN + TM [87]
SMS	62.60	1º 64.10 2º 63.70 3º 63.40	Handcrafted features[4] Word2Vec [84]/ Glove [85] Specific embeddings3	Ensemble Classifier [88] CNN Ensemble [86] CNN [83]
LiveJournal	71.24	1º 74.11 2º 72.60 3º 71.90	Specific embeddings3 Handcrafted features2 Handcrafted features4	CNN [83] LR [82] Ensemble Classifier [88]

[1] 52 million tweets used to train the word embeddings Structured Skip-Gram (SSG).
[2] 1,2,3-grams, negative features, sentiment lexicon, semantic features, Twitter dictionary.
[3] Embeddings trained on different units: lexical, part- of-speech, and sentiment embeddings.
[4] Punctuations, number of happy and sad emoticons, word and character n-grams, part-of-speech, and lexicons.

weights below 0.001. It is important to note that the low relevance of a lexicon in a particular domain does not imply that it assigns incorrect polarities to words but only that the lexicon was not expressive for that particular domain and benchmark. The lexicon AFINN (w_3) obtained the highest weight among the lexicons used, and GP understood that AFINN provides more effective help in the maximization of the correct predictions, followed by the lexicons VADER (w_4) and LIU (w_1).

Fig. 3 displays a graph that demonstrates the most used values for the inferior threshold (*IT*) and the superior threshold (*ST*) of the neutral class. The lowest value defined for the lower limit was slightly higher than -1. Likewise, the largest value for this limit is 4. Regardless, it can be noted that more than 75% of the values assigned to the lower limit are higher than zero, that is, positive. It is also possible to observe that the lowest value defined for the upper limit is zero, that is, all the inputs are positive and 75% of them were between 0 and 4, with the highest value being equal to 7. The mean value of the inferior threshold (*IT*) is 1.0226 and that of the superior threshold (*ST*) is 2.0080. These values demonstrate the adequacy of GP due to the limits of the neutral class, with a preference for positive values. This adaptation can be performed to balance the results from the evaluations made through the lexicons.

It is also interesting to analyze the best program resulting from the evolution process of GP. The best program is referred to as p_{best} below. As seen in Eq. 2, in p_{best}, the inferior threshold of the neutral class was defined by GP as absolute zero, i.e., the default value was maintained. The superior threshold, however, was changed to 0.841. This result reinforces the adequacy of the models to the class thresholds. Thus, the $r \in \mathbb{R}$ result of evaluating a message m for the program p_{best} follows the rule given below

$$r = \begin{cases} positive, & p_{best}(m) > 0.841 \\ neutral, & 0 \leqslant p_{best}(m) \leqslant 0.841 \\ negative, & p_{best}(m) < 0 \end{cases} \quad (2)$$

Analysis of the outputs from the p_{best} classifier can demonstrate the general behavior of the solution and how the best program assigns the classes to the inputs. As shown in Fig. 4, more than 75% of the p_{best} outputs are greater than zero, which does not mean that the resulting classes are all positive, since, as shown in Fig. 3, the threshold values for classification were changed. This behavior directly reflects the imbalance of the negative class in the training databases, which contains only 15% of messages of this type. Moreover, it is possible to observe that the adjustments made by GP tend to prioritize positive outputs and a redefinition of the upper limit of the neutral class. Thus, it is possible to observe that GP was able to adapt to the proposed problem, determining the lexicon weighting and the adjustments in the limits of each class.

GP, however, demands the longest training time compared to the other techniques used mainly because of the need to evaluate each individual for the fitness calculation. There are some ways to try to minimize this impact, such as parallelization [90]. Additionally, some variants may lower the cost of the GP process, such as root genetic programming (RGP), published in [40].

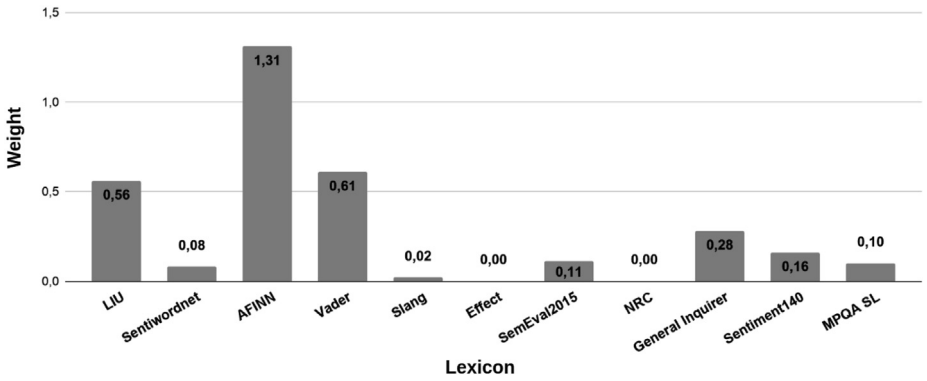

Fig. 2. Average weights per dictionary.

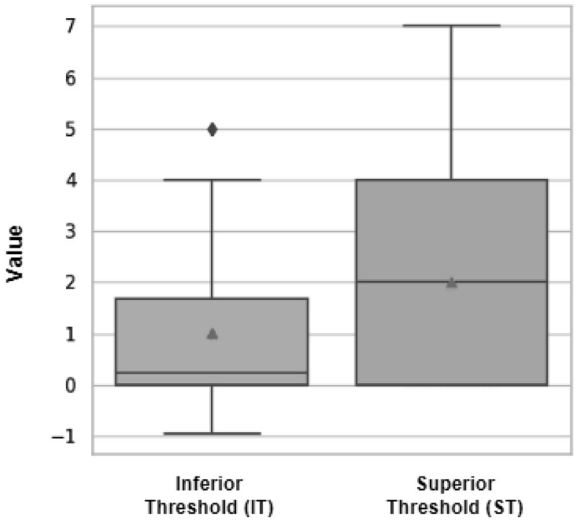

Fig. 3. Limit values of the classes.

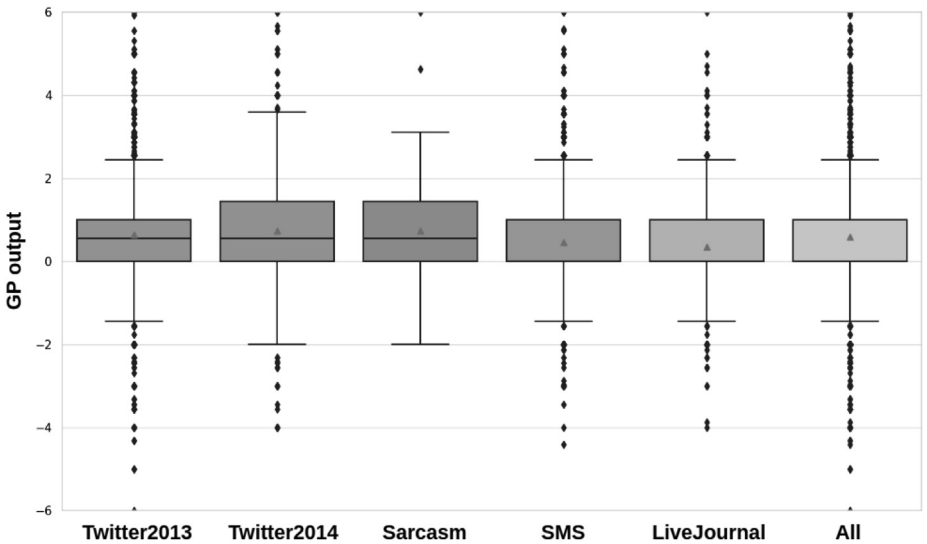

Fig. 4. Output values of p_{best} for each dataset.

6. Discussion

The results of this study indicate that the sentiment analysis classifier created with the GP is competitive compared to works that use the same benchmark. In regard to the LiveJournal dataset, which obtained the best values, the results revealed an F1-P/N (average F1-score considering the positive and negative classes) of 71.24, only 3.6 points inferior to the best method reported in [30] in SemEval 2014 and only 2.87 points inferior to the best method reported in [83] in SemEval 2016. In addition, GP reached 44.04 points above the majority baseline.

Compared to the classical algorithms of machine learning (SVM, NB, RF, LR, and SGD), the solution provided by GP was statistically equivalent to the two best algorithms (SVM and SGD) in the considered benchmark. GP was consistently better than RF and RL and was not worse than these methods in any test subset of the benchmark. GP was better than NB in all the subsets, except the Sarcasm subset, for which NB was the best overall method.

In addition to being very good at the classification task, as concluded above, GP is a very promising approach for resource selection and combination. We consider that this is the great potential of GP in the SA arena. As discussed in Section 1 and Section 5, SA is very sensitive to the combination of resources, and this fact is demonstrated, in our opinion, in competitions such as those of the SemEval Workshop. By including functions that are able to address different features and resource combinations, one can let GP evolve programs that are able to select the best resource and to combine these resources automatically for sentiment classification.

In this work, we experimented with 23 functions that allowed GP to explore text processing tasks, word polarities, lexicon combination, and neutral range selection, among others. In particular, the *polSumAVGW*() is an example of how GP can be used to appropriately combine eleven dictionaries, a task that is difficult to perform by hand. This function was very important in our case, being present in all the best programs in the 30 executions of GP. Additionally, the *neutralRange*() function is an example of the flexibility of GP in adapting the best range of polarity values to decide the class of the text messages. Despite being effective, as demonstrated in the experiments, the function set proposed here is by no means closed or definite. The function set can be extended to address many other features usually applied in competitions. For instance, extensions could include part-of-speech tagging, n-grams, and word embeddings, among others.

The developed source codes, generated models and configuration files used for the development of this paper are available under the GPL[20] license in GitHub[21]. The datasets and some lexicons are not available in the repository due to terms of use but can be found for download in the official pages.

7. Conclusion

The use of genetic programming applied to sentiment analysis has been underexplored in the literature. In this paper, the goal of generating a hybrid classifier for sentiment analysis using genetic programming and lexical combination has been achieved.

Among the main contributions of this research, we showed the importance of a good choice of resources (features and lexicons) and of a good combination of them to derive a competitive hybrid SA solution. We proposed a new hybrid approach using GP that can automatically combine preprocessing tasks and resources to derive an effective sentiment classifier. We showed examples of functions that can be used to compose individuals for GP. We provided a comparative study of GP and other sentiment classifiers that use different machine learning methods. The results showed the promising applicability of GP to SA.

As future improvements, the inclusion of new lexicons can further improve the results obtained by GP since they are essential resources to support the discovery of message polarities. The expansion of the function set proposed here should also be considered for future work. We also want to investigate the use of GP to combine sentiment classifiers, for instance, as an ensemble or as a stacking method.

CRediT authorship contribution statement

Airton Bordin Junior: Conceptualization, Methodology, Software, Writing - original draft. **Nádia Félix F. da Silva:** Data curation, Writing - original draft, Software. **Thierson Couto Rosa:** Visualization, Investigation, Writing - review & editing. **Celso G.C. Junior:** 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.

[20] https://opensource.org/licenses/GPL-3.0.
[21] https://github.com/airtonbjunior/opinionMining.

Acknowledgments

The authors would like to acknowledge the Brazilian Research Agency CNPq for their financial support.

References

[1] B. Liu, Sentiment analysis: a multifaceted problem, IEEE Intell. Syst. 25 (3) (2010) 76–80, https://doi.org/10.1109/MIS.2010.75.
[2] M. Atzeni, A. Dridi, D.R. Recupero, Using frame-based resources for sentiment analysis within the financial domain, Prog. AI 7 (4) (2018) 273–294.
[3] S. de Kok, L. Punt, R. van den Puttelaar, K. Ranta, K. Schouten, F. Frasincar, Review-aggregated aspect-based sentiment analysis with ontology features, Prog. AI 7 (4) (2018) 295–306.
[4] M. Lopez, A. Valdivia, E. Martinez-Camara, M.V. Luzon, F. Herrera, E2sam: evolutionary ensemble of sentiment analysis methods for domain adaptation, Inf. Sci. 480 (2019) 273–286.
[5] L. Becker, G. Erhart, D. Skiba, V. Matula, Avaya: sentiment analysis on twitter with self-training and polarity lexicon expansion, in: Second Joint Conference on Lexical and Computational Semantics (* SEM), vol. 2, 2013, pp. 333–340..
[6] H. Kanayama, T. Nasukawa, Fully automatic lexicon expansion for domain-oriented sentiment analysis, in: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, EMNLP '06, Association for Computational Linguistics, Stroudsburg, PA, USA, 2006, pp. 355–363. http://dl.acm.org/citation.cfm?id=1610075.1610125..
[7] N. Guimaraes, L. Torgo, A. Figueira, Lexicon expansion system for domain and time oriented sentiment analysis, in: Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016), 2016, pp. 463–471.https://doi.org/10.5220/0006081704630471..
[8] N. Kaji, M. Kitsuregawa, Building lexicon for sentiment analysis from massive collection of html documents, in: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Association for Computational Linguistics, Prague, Czech Republic, 2007, pp. 1075–1083. http://www.aclweb.org/anthology/D07-1115..
[9] S. Arora, E. Mayfield, C. Penstein-Rosé, E. Nyberg, 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, Association for Computational Linguistics, 2010, pp. 131–139..
[10] R. Feldman, Techniques and applications for sentiment analysis, Commun. ACM 56(4) (2013) 82–89.https://doi.org/10.1145/2436256.2436274..
[11] S. Vohra, J. Teraiya, A comparative study of sentiment analysis techniques, J. JIKRCE 2 (2) (2013) 313–317.
[12] A. Jurek, M. Mulvenna, Y. Bi, Improved lexicon-based sentiment analysis for social media analytics, Secur. Inf. 4(9) (2015).https://doi.org/10.1186/s13388-015-0024-x..
[13] B. Liu, Sentiment analysis and opinion mining, Synthesis lectures on human language technologies 5(1) (2012) 1–167..
[14] R. Poli, W. Langdon, N. McPhee, J. Koza, A Field Guide to Genetic Programming, Lulu Enterprises, UK Ltd, 2008. URL: https://books.google.com.br/books?id=3PBrqNK5fFQC..
[15] D.W. Hosmer Jr, S. Lemeshow, R.X. Sturdivant, Applied Logistic Regression, vol. 398, John Wiley & Sons, 2013..
[16] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[17] L. Bottou, Stochastic gradient descent tricks, in: Neural networks: Tricks of the trade, Springer, 2012, pp. 421–436..
[18] L. Breiman, Random forests, Mach. Learn. 45(1) (2001) 5–32..
[19] K.P. Murphy et al, Naive bayes classifiers, University of British Columbia 18 (2006).
[20] P. Nakov, A. Ritter, S. Rosenthal, F. Sebastiani, V. Stoyanov, SemEval-2016 task 4: sentiment analysis in twitter, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), Association for Computational Linguistics, San Diego, California, 2016, pp. 1–18.https://doi.org/10.18653/v1/S16-1001. https://www.aclweb.org/anthology/S16-1001..
[21] N.F.F.d. Silva, Análise de sentimentos em textos curtos provenientes de redes sociais, Ph.D. thesis, Universidade de São Paulo, 2016..
[22] W. Medhat, A. Hassan, H. Korashy, Sentiment analysis algorithms and applications: a survey, Ain Shams Eng. J. 5(4) (2014) 1093–1113..
[23] A. Go, R. Bhayani, L. Huang, Twitter sentiment classification using distant supervision, Processing 150 (2009)..
[24] O. Wijksgatan, L. Furrer, Gu-mlt-lt: Sentiment analysis of short messages using linguistic features and stochastic gradient descent, Atlanta, Georgia, USA 328 (2013)..
[25] T. Günther, J. Vancoppenolle, R. Johansson, Rtrgo: Enhancing the gu-mlt-lt system for sentiment analysis of short messages, in: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), 2014, pp. 497–502..
[26] Y. Miura, S. Sakaki, K. Hattori, T. Ohkuma, Teamx: A sentiment analyzer with enhanced lexicon mapping and weighting scheme for unbalanced data, in: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), 2014, pp. 628–632..
[27] H. Hamdan, P. Bellot, F. Bechet, Lsislif: Crf and logistic regression for opinion target extraction and sentiment polarity analysis, in: Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015), 2015, pp. 753–758..
[28] N. Malandrakis, M. Falcone, C. Vaz, J. J. Bisogni, A. Potamianos, S. Narayanan, Sail: Sentiment analysis using semantic similarity and contrast features, in: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), 2014, pp. 512–516..
[29] P. D. Turney, M. L. Littman, Unsupervised learning of semantic orientation from a hundred-billion-word corpus, arXiv preprint cs/0212012 (2002)..
[30] X. Zhu, S. Kiritchenko, S. Mohammad, Nrc-canada-2014: Recent improvements in the sentiment analysis of tweets, in: Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014), 2014, pp. 443–447..
[31] P. P. Balage Filho, L. V. Avanço, T. A. S. Pardo, M. d. G. V. Nunes, et al., Nilc_usp: an improved hybrid system for sentiment analysis in twitter messages., in: International Workshop on Semantic Evaluation, 8th, ACL Special Interest Group on the Lexicon-SIGLEX, 2014, pp. 428–432..
[32] R.-M. Karampatsis, J. Pavlopoulos, P. Malakasiotis, Aueb: Two stage sentiment analysis of social network messages, in: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), 2014, pp. 114–118..
[33] L. Barbosa, J. Feng, Robust sentiment detection on twitter from biased and noisy data, in: Proceedings of the 23rd international conference on computational linguistics: posters, Association for Computational Linguistics, 2010, pp. 36–44..
[34] E. Riloff, J. Wiebe, Learning extraction patterns for subjective expressions, in: Proceedings of the 2003 conference on Empirical methods in natural language processing, Association for Computational Linguistics, 2003, pp. 105–112..
[35] V.K. Jain, S. Kumar, An effective approach to track levels of influenza-a (h1n1) pandemic in india using twitter, Proc. Comput. Sci. 70 (2015) 801–807.
[36] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, R. Passonneau, Sentiment analysis of twitter data, in: Proceedings of the workshop on languages in social media, Association for Computational Linguistics, 2011, pp. 30–38..
[37] F. Aisopos, G. Papadakis, T. Varvarigou, Sentiment analysis of social media content using n-gram graphs, in: Proceedings of the 3rd ACM SIGMM International Workshop on Social Media, ACM, 2011, pp. 9–14..
[38] E. Kouloumpis, T. Wilson, J. Moore, Twitter sentiment analysis: the good the bad and the omg!, in: Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011, AAAI Press, 2011, pp. 538–541..
[39] H. Hamdan, F. Béchet, P. Bellot, Experiments with dbpedia, wordnet and sentiwordnet as resources for sentiment analysis in micro-blogging, in: Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013), Vol. 2, 2013, pp. 455–459..
[40] M. Graff, E. S. Tellez, H. J. Escalante, S. Miranda-Jiménez, Semantic genetic programming for sentiment analysis, in: NEO 2015, Springer, 2017, pp. 43–65..
[41] A. W. Almeida, Sentiment analysis in short messages using affective lexicons, Master's thesis, Pontifical Catholic University of Paraná, 2017..

[42] M. Affenzeller, S. Wagner, S. Winkler, A. Beham, Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications, CRC Press, 2009..
[43] H. Keshavarz, M.S. Abadeh, Alga: adaptive lexicon learning using genetic algorithm for sentiment analysis of microblogs, Knowl.-Based Syst. 122 (2017) 1–16.
[44] A. Abbasi, H. Chen, A. Salem, Sentiment analysis in multiple languages: feature selection for opinion classification in web forums, ACM Trans. Inf. Syst. 26(3) (2008) 12..
[45] A.S.H. Basari, B. Hussin, I.G.P. Ananta, J. Zeniarja, Opinion mining of movie review using hybrid method of support vector machine and particle swarm optimization, Proc. Eng. 53 (2013) 453–462.
[46] M. Govindarajan, Sentiment analysis of movie reviews using hybrid method of naive bayes and genetic algorithm, Int. J. Adv. Comput. Res. 3(4) (2013) 139..
[47] F. Neri, C. Cotta, P. Moscato, Handbook of Memetic Algorithms, vol. 379, Springer, 2011..
[48] R. Tanabe, A.S. Fukunaga, Improving the search performance of shade using linear population size reduction, in: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2014, pp. 1658–1665..
[49] A. Moraglio, K. Krawiec, C.G. Johnson, Geometric semantic genetic programming, in: International Conference on Parallel Problem Solving from Nature, Springer, 2012, pp. 21–31..
[50] T. Hastie, R. Tibshirani, Classification by pairwise coupling, in: Advances in Neural Information Processing Systems, 1998, pp. 507–513..
[51] A.C. Lorena, A.C. de Carvalho, Estratégias para a combinaçao de classificadores binários em soluçoes multiclasses, Revista de Informática Teórica e Aplicada 15 (2) (2008) 65–86.
[52] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, USA, 1992..
[53] R. Prabowo, M. Thelwall, Sentiment analysis: a combined approach, J. Inf. 3 (2) (2009) 143–157.
[54] J. Leal, S. Pinto, A. Bento, H. G. Oliveira, P. Gomes, Cisuc-kis: tackling message polarity classification with a large and diverse set of features, in: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), 2014, pp. 166–170..
[55] J. Saias, Senti.ue: Tweet overall sentiment classification approach for SemEval-2014 task 9, in: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), Association for Computational Linguistics, Dublin, Ireland, 2014, pp. 546–550.https://doi.org/10.3115/v1/S14-2095. https://www.aclweb.org/anthology/S14-2095..
[56] L. Flekova, O. Ferschke, I. Gurevych, Ukpdipf: Lexical semantic approach to sentiment polarity prediction in twitter data, in: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), 2014, pp. 704–710..
[57] M. Jaggi, F. Uzdilli, M. Cieliebak, Swiss-chocolate: Sentiment detection using sparse svms and part-of-speech n-grams, in: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), 2014, pp. 601–604..
[58] D. Tang, F. Wei, B. Qin, T. Liu, M. Zhou, Coooolll: A deep learning system for twitter sentiment classification, in: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), 2014, pp. 208–212..
[59] S. Evert, T. Proisl, P. Greiner, B. Kabashi, Sentiklue: Updating a polarity classifier in 48 hours, in: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), 2014, pp. 551–555..
[60] K. Al-Mannai, H. Alshikhabobakr, S. B. Wasi, R. Neyaz, H. Bouamor, B. Mohit, Cmuq-hybrid: Sentiment classification by feature engineering and parameter tuning, in: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), 2014, pp. 181–185..
[61] B. Velichkov, B. Kapukaranov, I. Grozev, J. Karanesheva, T. Mihaylov, Y. Kiprov, P. Nakov, I. Koychev, G. Georgiev, Su-fmi: System description for semeval-2014 task 9 on sentiment analysis in twitter, in: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), 2014, pp. 590–595..
[62] S. Kiritchenko, X. Zhu, S.M. Mohammad, Sentiment analysis of short informal texts, J. Artif. Intell. Res. 50 (2014) 723–762.
[63] T. Günther, L. Furrer, Gu-mlt-lt: Sentiment analysis of short messages using linguistic features and stochastic gradient descent, in: Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013), vol. 2, 2013, pp. 328–332..
[64] S. Bird, E. Klein, E. Loper, Natural Language Processing with Python, first ed., O'Reilly Media Inc, 2009.
[65] S. Mohammad, P.D. Turney, Crowdsourcing a word-emotion association lexicon, CoRR abs/1308.6297 (2013). arXiv:1308.6297. URL: http://arxiv.org/abs/1308.6297..
[66] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, M. Stede, Lexicon-based methods for sentiment analysis, Comput. Linguist. 37 (2) (2011) 267–307.
[67] S. Rosenthal, A. Ritter, P. Nakov, V. Stoyanov, Semeval-2014 task 9: Sentiment analysis in twitter, in: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), Association for Computational Linguistics, 2014, pp. 73–80.https://doi.org/10.3115/v1/S14-2009. http://aclweb.org/anthology/S14-2009..
[68] I. Steinwart, A. Christmann, Support Vector Machines, Springer Science & Business Media, 2008..
[69] O. Kolchyna, T.T.P. Souza, P.C. Treleaven, T. Aste, Twitter sentiment analysis, CoRR abs/1507.00955 (2015)..
[70] M. Hu, B. Liu, Mining and summarizing customer reviews, in: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2004, pp. 168–177..
[71] A. Esuli, F. Sebastiani, Sentiwordnet: a high-coverage lexical resource for opinion mining, Evaluation (2007).
[72] F. Å. Nielsen, A new anew: evaluation of a word list for sentiment analysis in microblogs, arXiv preprint arXiv:1103.2903 (2011)..
[73] C. J. Hutto, E. Gilbert, Vader: a parsimonious rule-based model for sentiment analysis of social media text, in: E. Adar, P. Resnick, M. D. Choudhury, B. Hogan, A. H. Oh (Eds.), ICWSM, The AAAI Press, 2014. http://dblp.uni-trier.de/db/conf/icwsm/icwsm2014.html#HuttoG14..
[74] L. Wu, F. Morstatter, H. Liu, Slangsd: Building and using a sentiment dictionary of slang words for short-text sentiment classification, CoRR abs/1608.05129 (2016)..
[75] Y. Choi, J. Wiebe, +/-effectwordnet: Sense-level lexicon acquisition for opinion inference, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1181–1191..
[76] S. Rosenthal, P. Nakov, S. Kiritchenko, S. Mohammad, A. Ritter, V. Stoyanov, Semeval-2015 task 10: Sentiment analysis in twitter, in: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Association for Computational Linguistics, Denver, Colorado, 2015, pp. 451–463. http://www.aclweb.org/anthology/S15-2078..
[77] P.J. Stone, D.C. Dunphy, M.S. Smith, D.M. Ogilvie, The General Inquirer: A Computer Approach to Content Analysis, MIT Press, Cambridge, MA, 1966..
[78] S.M. Mohammad, S. Kiritchenko, X. Zhu, Nrc-canada: building the state-of-the-art in sentiment analysis of tweets, arXiv preprint arXiv:1308.6242 (2013)..
[79] A.G. Prasad, S. Sanjana, S.M. Bhat, B. Harish, Sentiment analysis for sarcasm detection on streaming short text data, in: Knowledge Engineering and Applications (ICKEA), 2017 2nd International Conference on, IEEE, 2017, pp. 1–5..
[80] R. González-Ibánez, S. Muresan, N. Wacholder, Identifying sarcasm in twitter: a closer look, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers-Volume 2, Association for Computational Linguistics, 2011, pp. 581–586..
[81] S. Amir, R.F. Astudillo, W. Ling, M.J. Silva, I. Trancoso, INESC-ID at semeval-2016 task 4-a: Reducing the problem of out-of-embedding words, in: S. Bethard, D. M. Cer, M. Carpuat, D. Jurgens, P. Nakov, T. Zesch (Eds.), Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16–17, 2016, The Association for Computer Linguistics, 2016, pp. 238–242..
[82] H. Hamdan, SentiSys at SemEval-2016 task 4: Feature-based system for sentiment analysis in twitter, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), Association for Computational Linguistics, San Diego, California, 2016, pp. 190–197.https://doi.org/10.18653/v1/S16-1028. https://www.aclweb.org/anthology/S16-1028..

[83] M. Rouvier, B. Favre, SENSEI-LIF at SemEval-2016 task 4: polarity embedding fusion for robust sentiment analysis, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), Association for Computational Linguistics, San Diego, California, 2016, pp. 202–208..
[84] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 26, Curran Associates Inc, 2013, pp. 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf..
[85] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation., in: EMNLP, vol. 14, 2014, pp. 1532–1543..
[86] J. Deriu, M. Gonzenbach, F. Uzdilli, A. Lucchi, V. De Luca, M. Jaggi, SwissCheese at SemEval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), Association for Computational Linguistics, San Diego, California, 2016, pp. 1124–1128..
[87] E. Palogiannidi, A. Kolovou, F. Christopoulou, F. Kokkinos, E. Iosif, N. Malandrakis, H. Papageorgiou, S. Narayanan, A. Potamianos, Tweester at SemEval-2016 task 4: Sentiment analysis in twitter using semantic-affective model adaptation, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), Association for Computational Linguistics, San Diego, California, 2016, pp. 155–163..
[88] B. E. Jahren, V. Fredriksen, B. Gambäck, L. Bungum, NTNUSentEval at SemEval-2016 task 4: Combining general classifiers for fast twitter sentiment analysis, in: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), Association for Computational Linguistics, San Diego, California, 2016, pp. 103–108.https://doi.org/10.18653/v1/S16-1014. https://www.aclweb.org/anthology/S16-1014..
[89] A. Lacerda, Uso de programação genética para propaganda direcionada baseada em conteúdo, Master's thesis, Universidade Federal de Minas Gerais, 2008..
[90] D. B. Fogel, The advantages of evolutionary computation, in: Biocomputing and Emergent Computation: Proceedings of BCEC97, World Scientific Press, 1997, p. 1–11..