International Journal of Software Engineering and Knowledge Engineering

A Comparative Study of Ensemble Techniques based on Genetic Programming: A Case Study in Semantic Similarity Assessment

Jorge Martinez-Gil
Software Competence Center Hagenberg GmbH
Softwarepark 32a, 4232 Hagenberg, Austria
jorge.martinez-gil@scch.at
http://https://www.scch.at

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

The challenge of assessing semantic similarity between pieces of text through computers has attracted considerable attention from industry and academia. New advances in neural computation have developed very sophisticated concepts, establishing a new state-of-the-art in this respect. In this paper, we go one step further by proposing new techniques built on the existing methods. To do so, we bring to the table the stacking concept that has given such good results and propose a new architecture for ensemble learning based on genetic programming. As there are several possible variants, we compare them all and try to establish which one is the most appropriate to achieve successful results in this context. Analysis of the experiments indicates that Cartesian Genetic Programming seems to give better average results.

Keywords: Ensemble Learning; Genetic Programming; Semantic Similarity Measurement

1. Introduction

Making a computer capable of accurately identifying semantic similarities between pieces of textual information is one of the significant challenges for the scientific community. The possibility of obtaining good results in this direction is fundamental in many disciplines spanning databases, information retrieval, natural language processing, and machine translation, among many others.

Numerous methods, strategies, and frameworks have been proposed for this purpose. Some of them have achieved extraordinary notoriety thanks to the results. However, as is often the case in the world of technology, solutions that perform excellently in one aspect do not do so well for other essential elements. A clear example of this is artificial neural networks. To date, this family of methods has outperformed the other families in proposing solutions for the automatic calculation of semantic similarity. However, while this performance has been outstanding, these solutions neglect essential aspects such as the interpretability or transferability of the generated models.

For this and many other reasons, researchers and practitioners have continued their research looking for more and better alternatives to solve complex situations. One of these alternatives consists of building learning-to-learn solutions, known as ensemble learning strategies. Among all the ensemble learning strategies proposed so far, we look at one in particular called stacking [1].

Ensemble learning refers to a strategy for aggregating several proven performance methods for obtaining a better one [2]. Stacking is the ensemble capable of aggregating existing heterogeneous approaches. There are several other methods for ensemble learning, for example, bagging and boosting [3]. However, they almost always work by first aggregating different configurations of the same type of method and then applying hard and soft voting techniques. In this work, we look at stacking which consists in putting to work several methods of heterogeneous nature with the idea of having diversity so that each one performs better in different areas of the search space to get better results overall.

The benefit of stacking is that they can outperform the results emitted by the base estimators with much higher accuracy than if only one of them were used. So each method in the ensemble should be able to make significant contributions in some regions of the search space. In the other areas where its performance is not as good, that performance is blurred by the performance of one of the other methods in the ensemble [4].

Furthemore, additional considerations can be considered, such as that such a stacking strategy should be easily interpretable [5] or easily transferable to another scenario [6]. The reason for such a choice is based on the excellent performance achieved with such strategies in the past.

Regarding the learning part, we must acknowledge that in their attempt to solve problems in the best possible way, people have delegated to computers the development of programs that can perform some tasks. One of the most significant efforts in this direction is Genetic Programming (GP). This evolutionary strategy attempts to automatically identify the best possible program for a given task from among all those in the search space (the space of all possible computer programs). The hypothesis we address in this paper is that GP techniques are ideal for implementing ensemble learning strategies.

The reasons are various, but it is commonly accepted that by implementing a stacking strategy, we should be able to achieve a performance equal to or superior to the best primary method considered [7]. This is of particular interest since the development of complex computational techniques has yielded outstanding results.

Therefore, we aim to build stacking functions that can guarantee a good performance concerning several factors (and not just one, as is currently the case in the literature). For this reason, we have focused on GP, whereby the fundamental idea is that the resulting models should not be restricted to any particular format since they should be malleable enough to accommodate a variety of uses. Therefore, the contributions of this work are:

- We explore the concept of ensemble learning via GP with a case study in semantic similarity assessment. Although the concept of stacking is widely studied in the literature, the novelty of our approach is twofold: a) on the one hand, GP techniques have not usually been considered in this domain, and b) ensemble learning techniques for solving problems associated with textual semantic similarity have not been addressed in as much depth as they should be.
- We submit our novel strategies to an empirical study. Since a priori, we do not know which of the existing GP techniques is the most suitable. We perform a comparative study to identify the most promising in terms of the development of ensemble learning strategies for measuring semantic similarity.

As for the rest of the paper, we have proceeded with an organization: Section 2 shows state-of-the-art ensemble learning techniques, the GP challenge, and its most promising variants. Section 3 explains how to design and implement stacking techniques using several variants of GP. Section 4 reports the results we have obtained after subjecting our novel stacking strategies to several experiments with datasets widely used by the semantic similarity community. Section 5 discusses clearly and concisely the most significant implications that can be derived from our research work. Finally, we draw and highlight the conclusions and lessons that can be drawn from all the research we have carried out.

2. State-of-the-art

The research community is intensely interested in designing solutions to the challenge of semantic textual similarity in an automatic way [8]. Research along these lines has grown exponentially in the last few years. The truth is that finding reasonable solutions can benefit a wide range of fields and domains [9]. However, some researchers have decided not to propose more and more measures of semantic similarity but to apply criteria of rationality that allow them to take advantage of all the work done so far, just in the way that AutoML operates [10]. For this reason, some of these researchers have turned their eyes to the notion of ensemble learning, and more specifically, to the concept of stacking to try to create new strategies to tackle the problem, but without starting from scratch [11].

Thus, we find ourselves in a situation where we have numerous semantic similarity measures, many of which have proven reliability and trustworthiness [12, 13, 14]. In a given scenario, many of them may yield diametrically different results. At first glance, this fact may seem undesirable. It is paramount to build a strategy that capitalizes on diversity as its strength and around which to design a much more accurate and robust semantic similarity problem-solving scheme in the long run.

Furthermore, recent research suggests that practical approaches suitable for choosing ensemble learning algorithms automatically should be advanced [15]. For these reasons, we will now review the concepts of ensemble learning, its application

to the solution of semantic similarity problems, and the existing GP techniques from which we can benefit to build accurate, interpretable, and transferable stacking strategies. In this way, it seems that ensemble learning will become increasingly crucial in semantic similarity research in the coming years.

2.1. *Semantic Similarity Measurement*

The goal of the community has been, for a very long time, to develop systems that can automatically evaluate the semantic similarity of multiple textual pieces [16, 17], even if those parts reflect the same true notion but have different lexical representations [18, 19].

The most significant issue is that an increasing number of fundamental methods can estimate semantic similarity [20, 21]. There is a profusion of appropriate approaches accessible, each of which is founded on quite distinct ideas and presumptions [22], and the knowledge engineer is at a loss as to which one to apply [23].

As a result, many researchers think acceptably aggregating several semantic similarity measures could prevent errors when creating solutions that work reasonably well in production environments [24, 25, 26, 27, 28, 29].

However, the use of GP to build ensemble strategies has been studied little or not at all in the past. We think it would be good to fill this gap since GP brings advantages from the idea of building the meta-models from the ground up. It is possible, for instance, to impose onto the models the requirement that they are accurate and interpretable or that they fulfill specific characteristics that make it possible for them to be transferred to other situations of a similar kind. As a result, potential advantages of exploring this alternative is to bring in methods that are much more efficient, interpretable, and require far fewer computational resources to be ready to operate.

2.2. *Ensemble learning*

Ensemble learning is a discipline for aggregating individual semantic similarity measures to generate a model with higher predictive capabilities. Ensemble learning techniques are used in the literature to solve classification and regression problems. However, it can also be used to create semantic similarity measures that are supposed to perform better than each of those measures considered individually [30].

In order to generate such an aggregation model, a search process must be available that tries to best fit the structure, rules, and coefficients of the aggregation model. This is almost always done against a loss function [31]. Therefore, generating the final model is quite similar to an optimization process that seeks to maximize the fitness function for a given metric [32]. Moreover, these search processes differ enormously from other existing techniques based on algebraic aggregations (i.e., which do not admit training), such as the calculation of arithmetic means, weighted averages, or even hard and soft voting models.

The architecture of a classical stacking strategy is composed of two layers: a lower layer formed by the basic similarity measures to be aggregated and another layer at a higher level, which must generate an aggregation model that considers the items of the lower layer. The values returned by the lower layer components are often called intermediate predictions and are usually the basis on which the upper layer predictive model is developed [33]. Since we work with real values, all the information handled by the stacking strategy comprises this type of numerical information.

Moreover, if an ensemble learning strategy is well constructed, there are theoretical guarantees about its performance; it can never be inferior to the performance of the best semantic similarity measure of the inferior capable one [34]. The reason is evident since, to equal design performance, it would suffice to cancel all semantic similarity measures except the best one. For this reason, this technique enjoys considerable popularity among the research and industrial community since it prevents unusual behavior when deployed in real environments.

We use the classical two-layer architecture in this work, but it would be theoretically possible to employ more layered architectures [35]. The problem arises in that each layer adds enormous extra complexity to the system and does not usually bring significant gains in aspects such as interpretability [5], transferability [36] or the appetite for resource consumption (both in the form of time and electric power). If it is strictly necessary to stacketize several layers, the community usually resorts to deep learning models that have proven reasonably reliable when processing information of this type.

2.3. *Genetic Programming*

GP attempts to emulate the natural selection process of animal species to solve a specific task [37]. It is a computational paradigm where computer programs can be encoded as individuals from a population that evolves based on well-defined criteria. Such coding is done through genes representing essential elements of a programming language, whereby a suitable arrangement of such vital elements results in a computer program. Traditionally, tree-based representation has been predominant in this context since most imperative programming languages can be instantiated by means of a tree. However, because it needs to keep many pointers, the tree-based representation may need to be more memory-efficient in high-performance environments. Nevertheless, there are ways to fix this issue. For example, if each function has a fixed number of arguments, the tree can be shown as a linear sequence.

A generation is an iteration from one set of individuals to the next. Throughout each iteration, each individual's fitness in the population is assessed. The best individual, i.e., the individual arrived at after numerous iterations, is the program that best solves the required task. In order to arrive at such an individual, the strategy performs several basic operations regularly: It combines individuals to give rise to a better one and produces mutations so that the search can be extended

over the entire search space. Each generation is guaranteed at least equal to or better results than the previous generation. There are currently several paradigms for implementing a GP scheme that we will review below.

In addition, another line of research called multiobjective genetic programming (MOGP) aims to build ensemble strategies that satisfy several orthogonal goals at the same time [38]. In this way, an operator may choose the alternative that most effectively meets its specific needs. It is impossible to maximize all of the orthogonal criteria simultaneously, but it is nearly always viable to pick a solution that optimizes two criteria. For instance, one can select a solution that maximizes both precision and ease of implementation, accuracy and execution time, or ease of implementation and execution time, respectively [39]. This can be achieved with the help of well-established methods such as [40, 41, 42]. We will not cover this variant, which remains as possible for future work.

We then explain the technical details of our approach, which consists of a comparative study to determine which variant of genetic programming best suits the challenge of generating ensemble techniques that allow the assessment of semantic similarity.

3. Ensemble learning based on genetic programming

Among all the existing ensemble learning strategies (bagging, boosting, stacking), we work here with stacking that consists of training a model to aggregate the predictions of multiple different methods. The reason for this is that most of the research work has found it to be appropriate for dealing with heterogeneous components such as in this case of semantic similarity [15]. A final prediction is made using the intermediate results from those methods to produce the final forecast so that:

$$S : [0, 1]^n \rightarrow [0, 1]$$

Each of the inputs is called a base estimator. A base estimator is $be : \mu_1 \times \mu_2 \rightarrow \mathcal{R}$ mapping two variables μ_1 and μ_2 to a value $s \in \mathcal{R}$ in $[0, 1]$, where 0 stands for no similarity at all, and 1 stands for absolute similarity.

To implement such a function S, we need to consider a number of $\mathcal{T}$ training examples so that $\{(x_1, y_1), ..., (x_N,\ y_N)\}$. Then, an ensemble learning strategy needs to build a function f so that $f : X \rightarrow Y$, being X the input and Y the output. The function f is just a specific instance from all the possible functions F. The most common criterion for selecting such an f is usually the best performance concerning a loss function, as explained below.

In our specific scenario, we seek to minimize empirical risk; therefore, we need a loss function to determine what solution best fits the training data. To do that, given a gold standard $\mathcal{GS}$, i.e., a dataset created and curated by humans, the goal is to maximize the correlation between the $\mathcal{GS}$ and the results from $\mathcal{S}$ as shown in Eq. 1.

$$S = arg\,max_S\ correl(\vec{\mathcal{GS}}(\mu_i, \mu_n), \vec{\mathcal{S}}(\mu_i, \mu_n)) \tag{1}$$

$\mathcal{S}$ can take different semantic similarity measures as input. These measures will function as base estimators to obtain intermediate predictions to learn a higher-level predictive strategy.

Concerning the objective to be accomplished, it is essential to remark that we seek to maximize a correlation coefficient since semantic similarity is typically evaluated based on its correlation with human judgment [18] instead of utilizing, for instance, an evaluation method founded on information retrieval. The type of correlation to consider is strongly dependent on the problem faced.

3.1. *Baseline*

As a baseline, it is reasonable to choose one of the best-performing methods for building stacking strategies, i.e., linear regression [43, 44]. The purpose of linear regression is to ascertain the functional link between the several semantic similarity metrics initially considered [45]. The connection may be described using a mathematical equation, which links the outcome to several semantic similarity measures as shown in Eq. 2.

$$\vec{\alpha} = arg\ min\,(D, \vec{\alpha}) = arg\ min \sum_{i=1}^{n} (\vec{\alpha} \cdot \vec{a_i} - b_i)^2 \tag{2}$$

This method aims to produce as little an error as possible, i.e., the measurement of the smallest distance that can be achieved between the points and the regression line.

3.2. *Comparative study of GP variants*

GP is a subfield of machine learning that seeks to solve problems for which operators do not have straightforward solutions. GP's adaptable nature, untouched by human preconceptions or biases, enables it to deliver answers that are usually accurate and interpretable. Individuals are typically encoded by GP utilizing syntax trees. But since Koza's initial TGP [46], several research investigations have suggested different ways to encode GP individuals.

Nevertheless, almost all GP approaches have a few things in common; they perform operations to select the individuals that will pass to the next generation by crossover and then mutate some individuals to expand the search space. The crossover is achieved by swapping random parts of the individual. The mutation is also performed randomly on some part of the individual. This process is repeated generation after generation. The evolutionary process ends when one of the individuals reaches the desired fitness level or when a maximum number of iterations have been performed even if the desired result was not achieved.

In addition, several variants represent the information differently and look for optimization by playing with the rules of this representation. This can be done using binary trees, sequences of instructions, the Cartesian model of directed acyclic graphs, the evolution of grammars, employing instructions that represent the processing of the information through stacks, etc. In this work, we focus on the first three variants mentioned because they have given outstanding results in the past in a wide range of specific domains, both in terms of accuracy and low consumption of computational resources.

3.2.1. *Tree-based Genetic Programming*

When referring to Tree-like GP (TGP), we are talking about working with programs represented by tree-like structures, which significantly facilitates the recursive evaluation of multivariate expressions [47]. This type of GP is the classical one initially proposed by [46]. Representing programs as trees is very natural since it allows each node to be an operator and each terminal node (or leaf) to be an operand. This representation makes it easy to evolve the programs and calculate their associated fitness [48]. Moreover, the resulting model is easily exportable to a wide variety of modern programming languages.

Martinez-Gil and Chaves-Gonzalez [49] developed a method that aggregates semantic similarity measurements that is both successful and easy for people to grasp. The idea behind that approach is to compute the meta-model that provides the most significant match for a particular set of input data, which is then constructed using semantic similarity measures that are well-known and easily understood. This was made possible by the capacity to learn and improve the computation of the best viable tree through an evolutionary process. This made it possible for the tree to have the most optimal structure [49]. Figure 1 shows how a function is represented as a tree and how it is encoded in an individual to evolve toward finding the desired result.

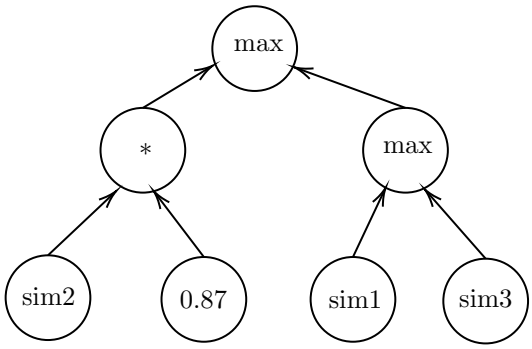

Fig. 1. Representation of the individual: max ((sim2 * 0.87), max(sim1, sim3))

In TGP, crossover and mutation operators are applied to the parents of the population. The crossover acts by exchanging randomly selected subtrees from each of the two parents and results in a new tree. The mutation is applied to a single randomly selected parent and results in a different tree. This work considers the classical **hoist mutation** that promotes a subtree as a result. In addition, it should be noted that the **parsimony rate**, or explicit penalty for large programs, is quite important as a means of controlling code growth in TGP. Otherwise, the size of the programs could get out of control and bring associated disadvantages such as over-fitting.

3.2.2. *Linear Genetic Programming*

Linear Genetic Programming (LGP) is a type of GP that allows programs to be represented by a sequence of instructions as in imperative programming languages [50]. The significant differences between the tree variant are how the data flow associated with the program is handled through the set of available registers and the existence of non-executable code items (called introns).

A LGP program consists of a variable number of functions and a terminal. It is possible to represent programs of variable size and that there can be many types of functions and terminals. In LGP, registers and constants play an important part in successfully completing this work, and the problem is that they must almost always be chosen based on experience. Listing 1 shows an example an aggregation strategy based on LGP.

Listing 1. Example of aggregation strategy using LGP

```
r[0] = r[0] - sim1;
r[1] = r[1] * r[1];
r[2] = r[0] + r[1];
r[3] = r[1] * sim2;
r[4] = r[3] - sim3;
r[5] = r[4] * 0.33;
```

Perhaps the two most important parameters in the context of the LGP are **register count**, and the determination of whether **if-then clauses** are allowed. The register count is limited in relation to the number of registers that the program being learned automatically can handle. There are many studies on how to determine the optimal number of registers [51]. Also, there are several studies that attempt to determine if the use of if-then clauses is beneficial [51]. In most situations it is, since it allows working with a greater diversity of solutions although the cost of computational resources is higher.

3.2.3. *Cartesian Genetic Programming*

Cartesian Genetic Programming (CGP) is a GP approach initially developed to aid in the automatic design of electronic circuits but later became popular as a

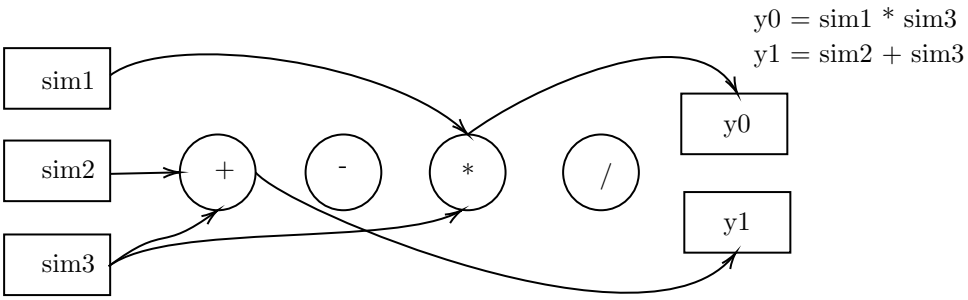

Fig. 2. Example of CGP. In contrast to what happens with TGP (where a tree is generated) and LGP (where an algorithm of linear nature is generated), in CGP, we seek to learn a directed acyclic graph that represents a computer program

variant of GP [52]. Recently, it has gained much relevance because it has proven its usefulness in various applications. Due to its origins, and unlike TGP or LGP, this paradigm aims to encode programs as directed acyclic graphs.

Directed graphs are represented in standard form by a two-dimensional grid (Cartesian grid). Coding programs as directed graphs has substantial advantages. For example, nodes can be used multiple times (whereas in TGP, they can only be used once). In addition, this format is very adaptable to a variety of situations. It can represent not only programs but systems of equations, electronic circuits, and even neural networks. Figure 2 shows a clear example of how the CGP paradigm operates.

In CGP, choosing the grid size on which the graph representing the final program will be learned is essential. This grid comprises **columns** and **rows**, so the graph is acyclic and directed. This means that a node may only have its inputs connected to either input data or the output of a node in a previous column. This way, the larger the grid one might choose to work with, the more computational resources will be needed to learn the program since the search space will be much larger. In return, finding better programs to solve the task would be technically possible. In addition, it is also necessary to choose the preferred **type of mutation**. The probabilistic mutation is the most typical in general-purpose contexts.

Another of the most substantial advantages of CGP is that it does not suffer from the bloat phenomenon, whereby programs become larger and larger without any benefit in terms of improved fitness function throughout its evolution. This is significant because bloat is one of the major problems of other variants of GP [53]. However, this approach is also time-consuming and requires mastery of the grid configuration on which the network is calculated.

4. Empirical evaluation

We present here the results we obtained after testing the different variants of GP to the design and implementing a stacking strategy that allows us to derive a data-driven method of predictive capability superior to the basic measures that have been considered. Therefore, the section consists of a description of the datasets we are working with, an explanation of the chosen configuration, and the numerical results obtained in the experimental phase. Then, with the results, we can present the most significant milestones of this comparative study.

4.1. *Datasets*

We intend to compare the different variants of GP concerning the baseline and each other. For our evaluation to be complete, we have chosen three benchmark datasets oriented to the assessment of the semantic similarity. The criteria for this choice has been the popularity they enjoy among the community. In fact, it can be observed that they are recurrently mentioned in a large number of surveys [20, 21], as well as the coverage of the three different granularity levels of text: word, sentence, and paragraph. These are the particular characteristics of each of these datasets:

- The first dataset used in our experiments is the so-called **Miller & Charles** (MC) dataset [18]. This is the standard dataset community members use when evaluating research methodologies that concentrate on general cases. Therefore, this dataset aims to evaluate the semantic similarity between words that are components of a general-purpose scenario.
- The second dataset is the so-called **GeReSiD** dataset [54] and is drawn from the realm of geospatial research. It covers a pool of textual phrases, each of which has been grouped into one of 50 unique pairings. This pool of sentences includes over 100 different geographical expressions. On each of the 50 pairings, human opinions about the degree of semantic similarity were solicited and recorded individually. These 50 pairings include samples that are in no way comparable to one another and others that, in human view, are virtually indistinguishable from one another.
- The third dataset is the so-called **lawSentence200** (LS200) dataset[a], which is a well-known benchmark for comparing legal paragraphs of text. This dataset comprises 200 paragraphs (several sentences grouped) taken from legal sources and paired together. In these paragraphs, a panel of legal professionals carefully labeled the dataset by rating the use cases using a scale from 1 (absolutely not comparable) to 5 (very similar).

Therefore, the experiments we will perform will be run on 30 use cases grouping 60 individual words of general purpose, 50 use cases grouping 100 short sentences of specific intent (geographic domain), and 200 use cases grouping 200 paragraphs of

[a]https://github.com/jorge-martinez-gil/nefusi/tree/main/datasets

specific purpose (legal field). In total, we attempted to solve 280 different use cases on 560 pieces of textual information, so we have a great diversity of use cases.

4.2. *Experimental setup*

To facilitate comparisons, we must choose a fixed set of parameters for the three variants under study. However, this is insufficient since each variant has parameters affecting the final result. However, we need to do this so that all three variants are consistent in the key parameters they handle, therefore we use a classical configuration in the field of evolutionary strategies.

- Variables: {**Jiang & Conrath** [55], **Leacock & Chodorow** [56], **Lin** [57], and **Resnik** [58] for **Miller & Charles**} and {**BERT** [12], **ELMO** [13] and **USE** [14] for the other two datasets}.
- Operators: {**sum**, **subtr**, **mul**, **protected div**, **sin**, **cos**, **max**, **min**}
- Fitness measure: **Pearson** or **Spearman** Correlation
- Parameters: {Population: **1000**, Crossover: **0.70**, Mutation: **0.10**}
- Criterion for stop: **100%** accuracy reached or **5,000** generations
- Cross-validation: **5-crossfold**

We have to use two subsets of semantic similarity measures according to the benchmark we will face because of the diverse nature of the base estimators we need. While to meet a benchmark dataset composed of individual terms, we have to use classic dictionary-based base estimators that have proven to work very well; this would not be possible to work with sentences and paragraphs of a textual nature. For these last two cases, we must rely on the help of semantic similarity measures based on transformers that are capable of estimating the similarity of text strings (where, for example, the position of the words in the sentence or words that work as modifiers of the complete sentence are essential).

The individual parameters that are associated with each GP variant must also be taken into account. An extensive explanation of these parameters' technical meaning is out of this work's scope. However, a detailed description can be found in the seminal papers of each technique.

- TGP [46]: {Hoist mutation: **0.05**, Parsimony: **0.01**}
- LGP [50]: {Register Count: **6**, If-Then Allowed: **Yes**}
- CGP [52]: {Column: **200**, Rows: **1**, Mutation: **Probabilistic**}

Finally, it should be remembered that the linear regression that will act as a baseline in our experiments does not require the configuration of any parameters since it is always about minimizing the accumulated point-to-point distance between the functions to be compared.

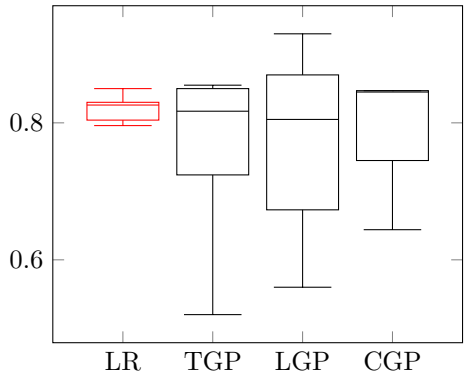

Fig. 3. Results for **PCC** over the **MC** dataset

4.3. *Results*

When speaking of correlation coefficients, we can always refer to what is perhaps the most commonly used: Pearson Correlation Coefficient (PCC) and Spearman Rank Correlation Coefficient (SRCC). The first one is used to estimate the absolute correlation between a gold standard and the results of the stacking strategy. While the second is usually used to check the relative (or ordinal) correlation between the predicted values by the stacking strategy and actual values. We will study the behavior of the different variants for the two correlation coefficients, one for each subsection.

4.3.1. *Evaluation based on absolute values*

PCC is a statistical method that may evaluate how strongly two variables are linked. It has a value that can vary from -1 to 1, where a value of -1 denotes a negative linear correlation and a value of 0 indicates no link between the two variables. If the value is +1, then the two variables have a positive correlation. As we are working with stochastic methods, i.e., the first population of individuals is always randomly generated, the results of each run will vary. In addition, we partition the datasets (training and testing) randomly. Therefore, the results have been calculated after 30 independent runs of each method under study.

Figure 3 shows us the results obtained by measuring the PCC on the MC dataset [18]. This picture also shows us that all approaches have obtained very similar results, especially if we look at the fact that the medians are pretty similar. However, the three GP variants have got much more variability in results.

Figure 4 show us the results obtained by measuring the PCC over the GeReSiD benchmark dataset [54]. As can be seen, the TGP variant is notoriously different from the other approaches, which achieve average results and quite similar variabilities.

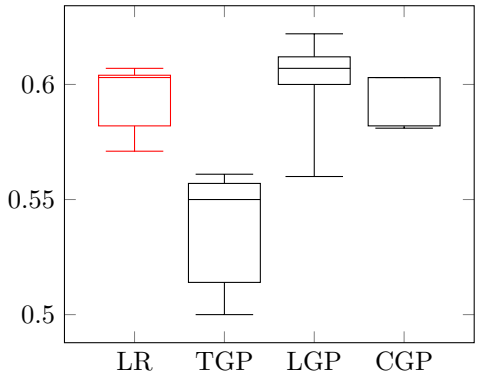

Fig. 4. Results for **PCC** over the **GeReSiD** dataset

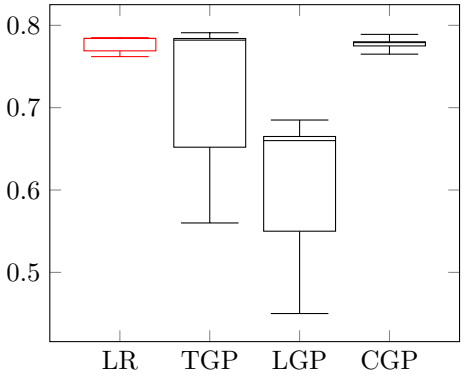

Fig. 5. Results for **PCC** over the **LS200** dataset

Figure 5 shows us the results obtained when measuring the PCC over the LS200 benchmark dataset. As can be seen, it is now the LGP variant that stands out negatively from the rest of the techniques. The other approaches again have results that are more or less in line. After all, the parameters and the base estimators these strategies use are very similar.

4.3.2. *Evaluation based on relative values*

The SRCC is a statistical method that may provide a concise summary of the relationship between two variables and the direction in which that association exists (positive or negative). In this regard, Figure 6 shows us the results obtained when calculating the SRCC over the MC dataset [18]. Here, instead of working with values representing an absolute numerical scale, we try to learn meta-models capable of leading to optimal results on an ordinal or relative numerical scale. As seen in the figure, the resulting median values are very similar. Only the LGP approach presents

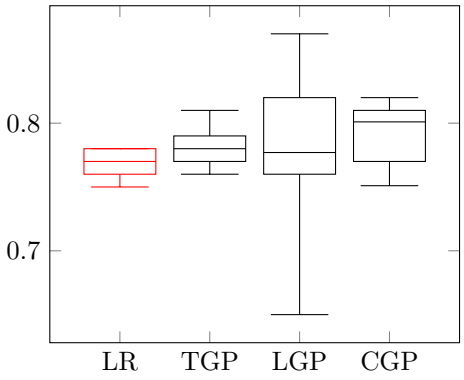

Fig. 6. Results for **SRC** over the **MC** dataset

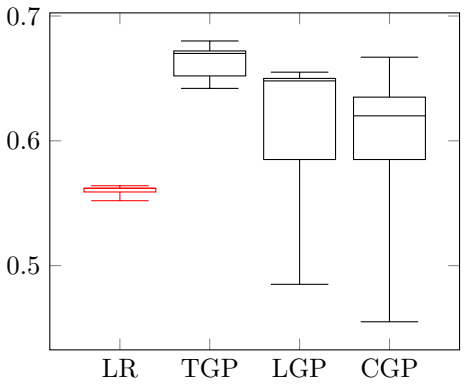

Fig. 7. Results for **SRC** over the **GeReSiD** dataset

a considerable variability that leads it to achieve the best maximum results with a tremendous difference in the relative (ordinal) numerical scale. This is similar to what happened in the previous experiments.

Figure 7 shows us the results obtained when measuring the SRCC over the GeReSiD benchmark dataset [54]. This figure shows that the three GP variants get better results. However, their variability is much higher than that of the GP variant.

Concerning our latest experiment, Figure 8 shows us the results achieved for the SRCC over the LS200 benchmark dataset. This time, the LGP technique stands out for the worse in relation to the other approaches.

We now turn to a quantitative and qualitative analysis of our empirical results. The objective is to compare the different techniques considered as LR (as baseline technique) and the three variants TGP, LGP, and CGP as strategies to build meta-models that allow the stacking of fundamental semantic similarity measures to achieve higher predictive performance.

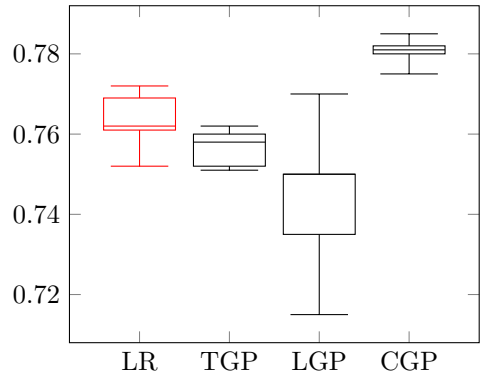

Fig. 8. Results for **SRC** over the **LS200** dataset

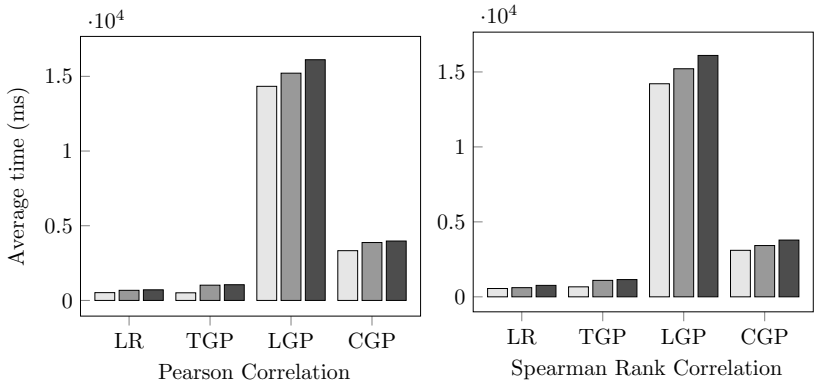

Fig. 9. Average execution times for the experiments performed

4.3.3. *Spacial and Temporal Analysis of the meta-models*

We now show a summary of the time needed as well as the final size of the generated meta-models using the different approaches that we have seen along the work. The experiments have been performed over a CPU 11th Gen Intel(R) Core(TM) i7-1185G7 at 3.00GHz with 64 GB RAM memory over Windows 10.

Figure 9 shows a summary of the the results obtained for the 6 experiments in terms of execution time. It is a visible fact that both the execution time spent to learn the meta-model and its size are correlated.

Figure 10 shows a summary of the results obtained for the 6 experiments in terms of size of the meta-model generated. We have measured the size in number of items, counting operators, variables and constants as one item in a homogeneous way. The generation of linear code by means of LGP is by far the most expensive operation. In contrast, the LR model is very inexpensive because only 5 coefficients are to be optimized. Therefore, both its time and size are the most efficient.

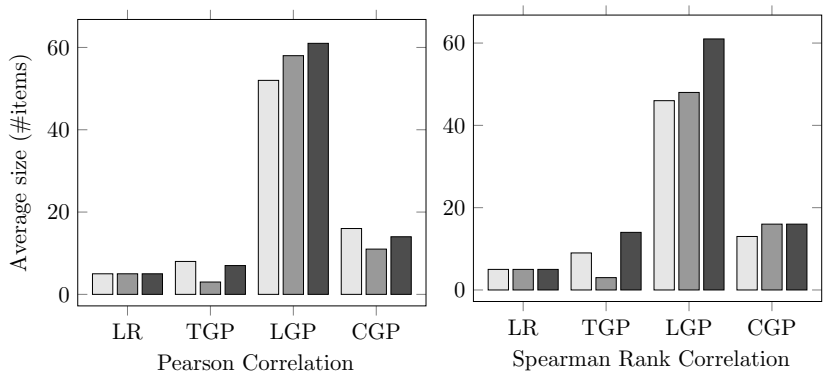

Fig. 10. Average size of the meta-models generated

We now proceed to analyze all the empirical results collected during the experiments to determine the best and worst performers, respectively.

4.3.4. *Analysis and lessons learned*

The goal is to extract the most significant lessons that can be learned from this research. To do that, we proceed to a double analysis: On the one hand, we are interested in knowing which approaches have performed better among all the experiments, and on the other hand, we are also interested in learning which approaches have performed worse.

Although we report the results here in summary format, the experiments performed have greatly expanded. We have tested cases in which only words were compared, cases in which only sentences were compared, and cases in which only paragraphs of text were compared. Moreover, some of these pieces of textual information were extremely specific, for example, those belonging to the LS200 dataset that pertains to the legal domain.

When comparing the results for each of the experiments we have performed, it is necessary to note that these results were obtained after testing 30 independent runs, where both the training sample and the cold start of the strategies were completely random. Therefore, the results reflect various situations concerning assessing semantic similarity.

We want to measure this behavior of the different approaches concerning five fundamental metrics: accuracy in terms of average results obtained for both PCC and SRCC, accuracy in terms of maximum results obtained for both PCC and SRCC, variability calculated as the difference between the best and worst results obtained, time calculated as the number of time units needed to generate the meta-models, and size which we have represented in bytes and which measures the amount of space (or size) required to store the meta-model, both in main memory and in secondary memory.

Table 1 shows the results of the different approaches to compute the best performances. As can be seen, the RL fails to excel in accuracy. However, it is the best approach when the other metrics are considered. TGP manages to be the best in terms of accuracy in some situations and still achieves decent results in the other metrics. LGP results in a somewhat chaotic behavior because although it fails in many metrics, it reaches the absolute best results in many cases. Finally, CGP performs quite well, with excellent results in calculating the median accuracy.

	LR	TGP	LGP	CGP
Accuracy (median)	0	2	1	3
Accuracy (maximum)	0	1	3	2
Variability	5	0	0	1
Time	5	1	0	0
Size (#items)	4	2	0	0

Table 1. Times each approach has been the **best** concerning the variables under study

Table 2 shows the summary of the results obtained for the different approaches but considering the worst performances only. As can be seen, LR gets the worst results in terms of accuracy on only two occasions, although it is never the worst in any of the other metrics. TGP and CGP present outstanding results, being the worst on a minimal number of occasions. Finally, LGP ranks as the worst approach on numerous occasions, especially regarding time and size variability and cost. Therefore, the lessons we can learn concerning the different alternatives to build meta-models from this comparative study are as follows:

	LR	TGP	LGP	CGP
Accuracy (median)	2	1	3	0
Accuracy (maximum)	2	2	1	1
Variability	0	2	3	1
Time	0	0	6	0
Size (#items)	0	0	6	0

Table 2. Times each approach has been the **worst** concerning the variables under study

- Concerning LR, it was always the lightest strategy in terms of the time needed to learn the model and the physical space required to store it. In addition, its results showed minor variability. As a disadvantage, it failed to excel in accuracy in any experiments.
- TGP turned out to be the lightest of the GP-based models. It obtained the best results on some occasions (both in the median and maximum values) and always had a lower cost than LGP and CGP.

- Concerning LGP, we have observed poor performance in most variables. For example, it is always the most space-consuming model (note that more than 50 lines of code were automatically generated in all attempts), the one that takes the longest time to learn the meta-model, and the one that obtains the most variable results. Even so, LGP has stood out for getting the best maximum results on a more significant number of occasions.
- CGP has performed exceptionally well compared to the other GP variants. It did not stand out negatively in almost any of the experiments performed and was the approach with the best performance in obtaining the best results (considering the median).

Finally, stacking techniques work best when more methods are aggregated (to some extent) and when there is diversity so that the weaknesses of each technique are blurred by the strengths of others in different regions of the search space. Designing a strategy based on size and diversity is always a good idea. It is certain that with the choice of other semantic similarity measures in which greater degrees of diversity could be found, the results would have been different.

5. Discussion

Ensemble learning has been a reasonable strategy to face the semantic similarity challenge over the last few years. To use it to generate a simple but robust aggregation strategy, we have focused our effort on exploring the GP-based family. However, even in this family, there are several variants of TGP, LGP, or CGP whose outcomes in a particular situation are often complicated to anticipate. For this reason, we have carried out this comparative study. The following are the lessons we have learned from this work as well as the threats to the validity of the research.

5.1. *Lessons learned*

The main benefit of ensembles is that they can take advantage of the capabilities of a wide range of proven methods to solve semantic similarity problems. Thus, it can often predict a more accurate result than the methods considered individually. We have seen how the use of GP techniques can lead to the learning of high-accuracy ensembles. In this regard, we have performed, for the first time to the best of our knowledge, a comparative study on the GP variants best suited to the challenge.

We should mention that, unlike what happens with other ensemble learning methods such as bagging or boosting, when working with stacking, we consider heterogeneous base estimators and what we are trying to do is to find an aggregation formula that can identify the parameters that should operate to obtain the best results.

As we assume that the search space is too large and that in some regions of this space, some semantic similarity measures will behave better than others, our goal is to use an evolutionary method that analytically identifies all these facts.

Therefore, we have emphasized diversity's critical role in completing a suitable strategy. Moreover, we always keep in mind additional but significant aspects, e.g., the interpretability of the resulting aggregation model, which will make people want to use these models in their daily activities.

5.2. *Threats to the validity*

We have been working with stochastic techniques, which means that each time we generate a model, we are likely to obtain a slightly different version from another model we may have generated previously. This is because the cold start of the methods requires the random generation of an initial state. In addition, the search method involves the random mutation of the solutions in search of other beneficial parameters in other significant parts of the search space. Therefore, given the same inputs, it is not assured that we will always have the same outputs. Our strategy to mitigate against the variability of results has been to run 30 instances of each program independently. The reported results have included statistics regarding the lower/upper whisker, median, and lower/upper quartile, which reflect very well the variability of the results obtained.

In addition, after building the ensemble in many cases, we may realize that its performance is not significantly superior to some of the semantic similarity measures considered fundamentally. In cases like these, and for simplicity and efficiency, it is usually a good idea to use only the semantic similarity measure that performed best within the ensemble. Therefore, it is assumed that the solution yielded by the ensemble can in no case be inferior to that of the best-represented method.

Furthermore, although throughout this paper, we have discussed the advantages of using GP to design and implement stacking strategies, it is also necessary to comment on some of its limitations. For example, stacked models consume much time and electrical energy during training, which is usually not sustainable in the long term. Not to mention, the base estimators also have to be trained simultaneously. In such a case, the resources required are usually vast. On top of that, stacking techniques are known to degrade quite frequently and are costly to recalibrate. These methods may not be optimal in a wide range of situations where sustainability is an integral part of the strategy to be deployed.

6. Conclusion

This paper has presented our research work on the design and implementation of ensemble learning techniques using different variants based on GP. These ensemble learning techniques aim to excel in solving problems related to the automatic assessment of semantic similarity.

We have studied the most promising variants of GP, including TGP, LGP, and CGP. Although the performance in terms of accuracy is similar, they offer different performances in additional aspects such as interpretability or transferability. In

addition, it is deduced that models with superior performance to existing, more advanced techniques can be achieved in some situations.

Among the lessons learned, although these GP techniques have been little studied, their performance is promising. Therefore, work in this direction could significantly contribute to the state-of-the-art. We are referring to the idea of gaining a few hundredths more in terms of accuracy and generating interpretable models that the people who use them can trust and solutions that can be reused in other situations. Alternatively, even allowing the incorporation of background knowledge makes the models much more helpful in practice.

We have also seen that several families allow the implementation of a stacking strategy. Besides those based on GP (either TGP, LGP, or CGP), other families are inspired by neural architectures [59] and architectures based on fuzzy logic [60]. Each family and subfamily has its characteristics that make them have associated advantages and disadvantages. However, more research needs to be done on the strategies most appropriate for each family and subfamily. In the end, many decisions are conditioned by the experience of the operator who uses them. The community shares that there is still much room for experimentation and adaptation of techniques. Therefore, this is a direction that should be explored in the future.

Acknowledgments

The author thank the anonymous reviewers for their help in improving the work. This work has been supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the State of Upper Austria through the COMET center SCCH. And by the project FR06/2020 - International Cooperation & Mobility (ICM) of the Austrian Agency for International Cooperation in Education and Research (OeAD-GmbH).

References

[1] D. H. Wolpert, Stacked generalization, *Neural Networks* **5**(2) (1992) 241–259.
[2] X. Dong, Z. Yu, W. Cao, Y. Shi and Q. Ma, A survey on ensemble learning, *Frontiers Comput. Sci.* **14**(2) (2020) 241–258.
[3] L. Breiman, Bagging predictors, *Mach. Learn.* **24**(2) (1996) 123–140.
[4] S. Dzeroski and B. Zenko, Is combining classifiers with stacking better than selecting the best one?, *Mach. Learn.* **54**(3) (2004) 255–273.
[5] I. Lopez-Gazpio, M. Maritxalar, A. Gonzalez-Agirre, G. Rigau, L. Uria and E. Agirre, Interpretable semantic textual similarity: Finding and explaining differences between sentences, *Knowl. Based Syst.* **119** (2017) 186–199.
[6] E. Haslam, B. Xue and M. Zhang, Further investigation on genetic programming with transfer learning for symbolic regression, in *IEEE Congress on Evolutionary Computation, CEC 2016, Vancouver, BC, Canada, July 24-29, 2016*, (IEEE, 2016), pp. 3598–3605.
[7] K. M. Ting and I. H. Witten, Stacked generalizations: When does it work?, in *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97, Nagoya, Japan, August 23-29, 1997, 2 Volumes*, (Morgan Kaufmann, 1997), pp. 866–873.

[8] J. J. Lastra-Díaz, A. García-Serrano, M. Batet, M. Fernández and F. Chirigati, HESML: A scalable ontology-based semantic similarity measures library with a set of reproducible experiments and a replication dataset, *Inf. Syst.* **66** (2017) 97–118.

[9] J. Martinez-Gil and J. F. Aldana-Montes, Semantic similarity measurement using historical google search patterns, *Inf. Syst. Frontiers* **15**(3) (2013) 399–410.

[10] X. He, K. Zhao and X. Chu, Automl: A survey of the state-of-the-art, *Knowl. Based Syst.* **212** (2021) p. 106622.

[11] M. Heilman and N. Madnani, HENRY-CORE: domain adaptation and stacking for text similarity, in *Proceedings of the Second Joint Conference on Lexical and Computational Semantics, *SEM 2013, June 13-14, 2013, Atlanta, Georgia, USA*, eds. M. T. Diab, T. Baldwin and M. Baroni (Association for Computational Linguistics, 2013), pp. 96–102.

[12] J. Devlin, M. Chang, K. Lee and K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, eds. J. Burstein, C. Doran and T. Solorio (Association for Computational Linguistics, 2019), pp. 4171–4186.

[13] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee and L. Zettlemoyer, Deep contextualized word representations, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, eds. M. A. Walker, H. Ji and A. Stent (Association for Computational Linguistics, 2018), pp. 2227–2237.

[14] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, B. Strope and R. Kurzweil, Universal sentence encoder for english, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, eds. E. Blanco and W. Lu (Association for Computational Linguistics, 2018), pp. 169–174.

[15] J. Martinez-Gil, A comprehensive review of stacking methods for semantic similarity measurement, *Machine Learning with Applications* **10** (2022) p. 100423.

[16] J. J. Lastra-Díaz and A. García-Serrano, A new family of information content models with an experimental survey on wordnet, *Knowl.-Based Syst.* **89** (2015) 509–526.

[17] J. J. Lastra-Díaz, A. Lara-Clares and A. García-Serrano, HESML: a real-time semantic measures library for the biomedical domain with a reproducible survey, *BMC Bioinform.* **23**(1) (2022) p. 23.

[18] G. Miller and W. Charles, Contextual correlates of semantic similarity, *Language and Cognitive Processes* **6**(1) (1991) 1–28.

[19] F. Hill, R. Reichart and A. Korhonen, Simlex-999: Evaluating semantic models with (genuine) similarity estimation, *Comput. Linguistics* **41**(4) (2015) 665–695.

[20] R. Navigli and F. Martelli, An overview of word and sense similarity, *Nat. Lang. Eng.* **25**(6) (2019) 693–714.

[21] D. Chandrasekaran and V. Mago, Evolution of semantic similarity - A survey, *ACM Comput. Surv.* **54**(2) (2021) 41:1–41:37.

[22] S. Harispe, S. Ranwez, S. Janaqi and J. Montmain, Semantic similarity from natural language and ontology analysis *Synthesis Lectures on Human Language Technologies* (Morgan & Claypool Publishers, 2015)

[23] J. Martinez-Gil and J. M. Chaves-Gonzalez, Semantic similarity controllers: On the trade-off between accuracy and interpretability, *Knowl. Based Syst.* **234** (2021) p.

107609.

[24] G. Pirrò, A semantic similarity metric combining features and intrinsic information content, *Data Knowl. Eng.* **68**(11) (2009) 1289–1308.

[25] P. Potash, W. Boag, A. Romanov, V. Ramanishka and A. Rumshisky, Simihawk at semeval-2016 task 1: A deep ensemble system for semantic textual similarity, in *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval-NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, 2016 pp. 741–748.

[26] J. Martinez-Gil, Coto: A novel approach for fuzzy aggregation of semantic similarity measures, *Cogn. Syst. Res.* **40** (2016) 8–17.

[27] G. Zhu and C. A. Iglesias, Computing semantic similarity of concepts in knowledge graphs, *IEEE Trans. Knowl. Data Eng.* **29**(1) (2017) 72–85.

[28] J. Martinez-Gil and J. M. Chaves-Gonzalez, Automatic design of semantic similarity controllers based on fuzzy logics, *Expert Syst. Appl.* **131** (2019) 45–59.

[29] J. Martinez-Gil, R. Mokadem, J. Küng and A. Hameurlain, A novel neurofuzzy approach for semantic similarity measurement, in *Big Data Analytics and Knowledge Discovery - 23rd International Conference, DaWaK 2021, Virtual Event, September 27-30, 2021, Proceedings*, eds. M. Golfarelli, R. Wrembel, G. Kotsis, A. M. Tjoa and I. Khalil *Lecture Notes in Computer Science* **12925**, (Springer, 2021), pp. 192–203.

[30] A. Ballatore, M. Bertolotto and D. C. Wilson, The semantic similarity ensemble, *J. Spatial Inf. Sci.* **7**(1) (2013) 27–44.

[31] Z.-H. Zhou, *Ensemble methods: foundations and algorithms* (CRC press, 2012).

[32] J. Martinez-Gil, Semantic similarity aggregators for very short textual expressions: a case study on landmarks and points of interest, *J. Intell. Inf. Syst.* **53**(2) (2019) 361–380.

[33] G. I. Webb and Z. Zheng, Multistrategy ensemble learning: Reducing error by combining ensemble learning techniques, *IEEE Trans. Knowl. Data Eng.* **16**(8) (2004) 980–991.

[34] A. L. V. Coelho, C. A. M. Lima and F. J. V. Zuben, Ga-based selection of components for heterogeneous ensembles of support vector machines, in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2003, Canberra, Australia, December 8-12, 2003*, (IEEE, 2003), pp. 2238–2245.

[35] A. I. Naimi and L. B. Balzer, Stacked generalization: an introduction to super learning, *European journal of epidemiology* **33**(5) (2018) 459–464.

[36] L. Torrey and J. Shavlik, Transfer learning, in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, (IGI global, 2010) pp. 242–264.

[37] N. I. Nikolaev, Genetic programming and data structures: Genetic programming+data structures=automatic programming, *Softw. Focus* **2**(4) (2001) 164–165.

[38] C. J. Hinde, N. Chakravorti and A. A. West, Multi objective symbolic regression, in *Advances in Computational Intelligence Systems - Contributions Presented at the 16th UK Workshop on Computational Intelligence, September 7-9, 2016, Lancaster, UK*, 2016 pp. 481–494.

[39] J. Martinez-Gil and J. M. Chaves-Gonzalez, Sustainable semantic similarity assessment, *Journal of Intelligent & Fuzzy Systems* **43**(5) (2022) 6163–6174.

[40] K. Deb, S. Agrawal, A. Pratap and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. Evol. Comput.* **6**(2) (2002) 182–197.

[41] S. Kukkonen and J. Lampinen, GDE3: the third evolution step of generalized differential evolution, in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005, 2-4 September 2005, Edinburgh, UK*, (IEEE, 2005), pp. 443–450.

[42] Q. Zhang and H. Li, MOEA/D: A multiobjective evolutionary algorithm based on

decomposition, *IEEE Trans. Evol. Comput.* **11**(6) (2007) 712–731.

[43] J. M. Chaves-Gonzalez and J. Martinez-Gil, Evolutionary algorithm based on different semantic similarity functions for synonym recognition in the biomedical domain, *Knowl. Based Syst.* **37** 62–69.

[44] J. J. Lastra-Díaz, J. Goikoetxea, M. A. H. Taieb, A. García-Serrano, M. B. Aouicha and E. Agirre, A reproducible survey on word embeddings and ontology-based methods for word similarity: Linear combinations outperform the state of the art, *Eng. Appl. Artif. Intell.* **85** (2019) 645–665.

[45] P. Greiner, T. Proisl, S. Evert and B. Kabashi, KLUE-CORE: A regression model of semantic textual similarity, in *Proceedings of the Second Joint Conference on Lexical and Computational Semantics, *SEM 2013, June 13-14, 2013, Atlanta, Georgia, USA.*, 2013 pp. 181–186.

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

[47] E. Vladislavleva, G. Smits and D. den Hertog, On the importance of data balancing for symbolic regression, *IEEE Trans. Evolutionary Computation* **14**(2) (2010) 252–277.

[48] M. Affenzeller, S. M. Winkler, G. Kronberger, M. Kommenda, B. Burlacu and S. Wagner, Gaining deeper insights in symbolic regression, in *Genetic Programming Theory and Practice XI [GPTP 2013, University of Michigan, Ann Arbor, USA, May 9-11, 2013].*, 2013 pp. 175–190.

[49] J. Martinez-Gil and J. M. Chaves-Gonzalez, A novel method based on symbolic regression for interpretable semantic similarity measurement, *Expert Syst. Appl.* **160** (2020) p. 113663.

[50] M. Brameier, W. Banzhaf and W. Banzhaf, *Linear genetic programming* (Springer, 2007).

[51] L. F. D. P. Sotto and V. V. de Melo, Comparison of linear genetic programming variants for symbolic regression, in *Genetic and Evolutionary Computation Conference, GECCO '14, Vancouver, BC, Canada, July 12-16, 2014, Companion Material Proceedings*, eds. D. V. Arnold and E. Alba (ACM, 2014), pp. 135–136.

[52] J. F. Miller and S. Harding, Cartesian genetic programming, in *Genetic and Evolutionary Computation Conference, GECCO 2008, Proceedings, Atlanta, GA, USA, July 12-16, 2008, Companion Material*, eds. C. Ryan and M. Keijzer (ACM, 2008), pp. 2701–2726.

[53] J. F. Miller, Cartesian genetic programming: its status and future, *Genet. Program. Evolvable Mach.* **21**(1-2) (2020) 129–168.

[54] A. Ballatore, M. Bertolotto and D. C. Wilson, An evaluative baseline for geo-semantic relatedness and similarity, *GeoInformatica* **18**(4) (2014) 747–767.

[55] J. J. Jiang and D. W. Conrath, Semantic similarity based on corpus statistics and lexical taxonomy, in *Proceedings of the 10th Research on Computational Linguistics International Conference, ROCLING 1997, Taipei, Taiwan, August 1997*, 1997 pp. 19–33.

[56] C. Leacock and M. Chodorow, Combining local context and wordnet similarity for word sense identification, *WordNet: An electronic lexical database* **49**(2) (1998) 265–283.

[57] D. Lin, An information-theoretic definition of similarity, in *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, 1998 pp. 296–304.

[58] P. Resnik, Using information content to evaluate semantic similarity in a taxonomy, in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelli-*

gence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes, (Morgan Kaufmann, 1995), pp. 448–453.

[59] Y. Goldberg, Neural network methods for natural language processing, *Synthesis lectures on human language technologies* **10**(1) (2017) 1–309.

[60] O. Cordón, A historical review of evolutionary learning methods for mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems, *Int. J. Approx. Reason.* **52**(6) (2011) 894–913.