Computer Science and Information Systems 12(2):635–654 DOI: 10.2298/CSIS140905020L

# Tuning a Semantic Relatedness Algorithm using a Multiscale Approach

José Paulo Leal[1] and Teresa Costa[2]

[1] CRACS & INESC-Porto LA, Faculty of Sciences, University of Porto
Porto, Portugal
zp@dcc.fc.up.pt

[2] CRACS & INESC-Porto LA, Faculty of Sciences, University of Porto
Porto, Portugal
teresa.costa@dcc.fc.up.pt

**Abstract.** The research presented in this paper builds on previous work that lead to the definition of a family of semantic relatedness algorithms. These algorithms depend on a semantic graph and on a set of weights assigned to each type of arcs in the graph. The current objective of this research is to automatically tune the weights for a given graph in order to increase the proximity quality. The quality of a semantic relatedness method is usually measured against a benchmark data set. The results produced by a method are compared with those on the benchmark using a nonparametric measure of statistical dependence, such as the Spearman's rank correlation coefficient. The presented methodology works the other way round and uses this correlation coefficient to tune the proximity weights. The tuning process is controlled by a genetic algorithm using the Spearman's rank correlation coefficient as fitness function. This algorithm has its own set of parameters which also need to be tuned. Bootstrapping is a statistical method for generating samples that is used in this methodology to enable a large number of repetitions of a genetic algorithm, exploring the results of alternative parameter settings. This approach raises several technical challenges due to its computational complexity. This paper provides details on techniques used to speedup the process. The proposed approach was validated with the WordNet 2.1 and the WordSim-353 data set. Several ranges of parameter values were tested and the obtained results are better than the state of the art methods for computing semantic relatedness using the WordNet 2.1, with the advantage of not requiring any domain knowledge of the semantic graph.

**Keywords:** Semantic similarity, Linked data, Genetic algorithms, Bootstrapping, WordNet

## 1. Introduction

Consider a magazine, a pencil and a notepad. From these three items which is the most related pair? Is it magazine and pencil, magazine and notepad or notepad and pencil? People living in more individualistic societies tend to find the magazine and the notepad more related, since they are both made of paper, while people living in more collectivist societies tend to find the pencil and the notepad more related since they complement each other [13]. The differences are even more striking when people are asked to assign a value to relatedness [11]. These experiments reveal the lack of a standard definition of relatedness and the difficulty to measure the relatedness of two concepts.

This paper is an extended version of paper presented at the Symposium on Languages, Applications and Technologies [17] and presents ongoing work aiming the development of a new methodology to determine the semantic relatedness between a pair of concepts. This methodology is knowledge based, it is applied to a semantic graph and does not require any knowledge about the graph's domain. It uses a family of semantic relatedness algorithms based on the notion of proximity [16]. An algorithm from this family is parametrized by a semantic graph, where the resources are the graph nodes and the properties the arcs. Each type of arc has a specific weight value. Tuning these weights in order to improve the quality of the semantic relatedness is the current objective of this research.

There are other methods based on semantic graphs available in the literature [1, 21, 31] that measure the quality of their algorithms using as benchmark a standard data set [11]. This data set has the reference similarity of concept pairs that is the average similarity assigned by a group of persons. The relatedness computed with an algorithm is compared against those of the benchmark using a nonparametric measure of statistical dependence, such as the Spearman's rank correlation, and its quality is as high as the value of the correlation.

A measure of quality is essential for using a genetic algorithm in a tuning process. In this tuning approach, an assignment of values to weights is encoded as a set of genes of a chromosome. New chromosomes are obtained by crossover and mutation of chromosomes from the previous generation where the best are selected using a fitness function. The fitness function receives as input a weight assignment and returns the Spearman's rank order correlation for a subset of benchmark data.

The genetic algorithm has in turn its own set of parameters that need to be tuned. Bootstrapping is a statistical procedure that produces a larger number of samples that are used to explore the most promising settings of the genetic algorithm. The best setting is finally used to run the genetic algorithm a larger number of times with the complete data set.

The tuning approach was validated with the semantic graph of WordNet 2.1 [10], using as benchmark the WordSimilarity-353 [11] data set. The obtained results are better than the best result available in the literature for the same knowledge source and benchmark [31].

Due to its computational complexity, the tuning approach raises several technical challenges. Firstly, the semantic algorithms must collect a large number of paths connecting each pair of labels in the graph. Secondly, in order to compute the Spearman's rank order correlation, the semantic relatedness algorithm must be executed with several hundreds of pairs of concept labels. Thirdly, the genetic algorithm must compute a correlation for each chromosome (a set of weight assignments) in the genetic pool for hundreds of generations. And finally, the evolution process of the genetic algorithm has to be repeated hundreds of times as part of the bootstrapping method. This paper also presents approaches used to speedup the tuning process.

The rest of the paper is organized as follows. The next section surveys the state of the art on semantic relatedness and Section 3 presents the family of semantic relatedness algorithms designed in previous research. Section 4 describes the tuning methodology and Section 5 details its implementation. The experimental results and their analysis can be found in Section 6. Finally, Section 7 summarizes what was accomplished so far and identifies opportunities for further research.

## 2. Related Work

Semantic measures, such as semantic relatedness, are widely used to estimate the strength of a semantic relationship between elements. The information needed to estimate those measures can be extracted from two different source types.

Sources of the first type are unstructured or semi-structured texts. These texts have evidences of semantic relationships among their content. In order to estimate the strength of a semantic relationship it is possible to use simple assumptions regarding the distribution of words. Sources of the other type are structured and explicit knowledge representations. Semantic measures based on this source type rely on semantic graph processing techniques. The approaches of computing semantic measures are based on the type of the source: the distributional methods, the knowledge-based methods and the hybrid methods.

The distributional approaches use unstructured texts analysis. They compare words, sentences or documents and their occurrence. These approaches rely on the *distributional hypothesis* [14] which states that words occurring in similar contexts tend to be semantically close. This means that words surrounded by the same words (similar context) are likely to be semantically similar. There are several methods following this approach, such as the spatial/geometric methods, the set-based methods and the probabilistic methods.

The spacial/geometric methods assume a semantic space where a word, for instance, is a point in a multi-dimensional space that represents the diversity of the vocabulary used. Two words are compared regarding their location in the multi-dimensional space. An example of this approach is found in Ganesan [12] work.

The set-based methods compare words regarding their context, considering which are common and which are different, using classical set-based measures such as the Dice index or the Jaccard coefficient. The work of Bollegala [5] and Terra & Clarke [32] are examples of this type of methods.

The probabilistic methods express the semantic relatedness of words in terms of probability of co-occurrence. They consider both the contexts in which the compared pairs of words appear and the contexts in which the two words co-occur. These evidences are used to estimate the semantic relatedness. This type of methods were used by Dagan [7] in his work.

Distributional approaches do not require prior knowledge of the meaning and usage of words, but they have some limitations; they are highly dependent of the corpus used and words to compare must occur at least few times. They also require word disambiguation process prior to comparison.

The knowledge-based approaches rely on structured data, such as semantic graphs. They consider the structural properties of the graph and elements are compared by studying their interconnections and the semantic of those relationships. Different methods have been defined to compare elements in a single knowledge base and also in multiple knowledge bases, such as structural methods, feature-based methods and Shannon's Information Theory methods.

The structural methods use the graph structure, nodes and arcs, to compare a pair of elements, expressing the semantic measure as a function of the strength of its connections. Most of these methods are based on the shorter path connecting both elements (the shorter the path the stronger is the semantic relationship) and only consider taxonomical relationships. Other structural methods have been proposed that consider all types of relationships connecting the elements. Examples of this approach are the work of Rada [26], Resnik [28] and Li [18, 19].

The feature-based methods estimate the semantic relatedness considering specific properties of the elements during the process. These methods need a function to characterize *features* of the elements that will be compared. The semantic relatedness is obtained by evaluating the number of features they share. Bodenreider [4], Stojanovic [30] and Ranwez [27] followed these methods.

There are also methods based on the Shannon's Information Theory. In these methods, the elements are compared using the amount of information they share and the amount of information they have distinct. Lin [20] and Pirró [25, 23, 24] developed their approaches based on this type of methods.

With knowledge-base approaches is possible to compare every pair of elements represented in the graph and also control which properties should be considered in the evaluation. These methods are easier to implement than the distributional methods and have a lower complexity. However they require a knowledge representation describing the elements.

There are also hybrid approaches that mix knowledge-base and distributional approaches. These methods take advantage of both texts and knowledge representations to estimate the semantic relatedness. Examples of these methods are given by Resnik [28], Banerjee & Pedersen [2, 3] and Patwardhan [22].

The methodology presented in this paper follows a knowledge-based relying in structural methods as detailed on the next section.

## 3. Previous Work

Concepts in semantic graphs are represented by nodes. Take for instance the music domain. Singers, bands, music genres, instruments or virtually any concept related to music is represented as nodes in a knowledge representation. These nodes are related by properties, such as `has genre` connecting singers to genres, and thus form a graph.

The semantic relatedness measure in development in this research uses a semantic graph to compute the relatedness between nodes. Actually, the goal is the relatedness between concepts, but concept nodes of semantic graphs typically have a label — a string representation or stringification — that can be seen as a term.

At first sight relatedness may seem to be the inverse of the distance between nodes. Two nodes far apart are unrelated and every node is totally (infinitely) related to itself. Interpreting relatedness as a function of distance has an obvious advantage: computing distances between nodes in a graph is a well studied problem with several known algorithms. After assigning a weight to each arc one can compute the distance as the minimum length of all the paths connecting the two nodes.

On a closer inspection this interpretation of relatedness as the inverse of distance reveals some problems. Consider the graph in Figure 1. Depending on the weight assigned to the arcs formed by the properties `has type` and `has genre`, the distances between Lady Gaga, Madonna and Queen are the same. If the `has genre` has less weight than `has type`, this would mean that the band Queen is as related to Lady Gaga as Madonna, which obviously should not be the case. On the other hand, if `has type` has less weight than `has genre` then Queen is more related to AC/DC than to Lady Gaga or Madonna simply because they are both bands, which also should not be the case.

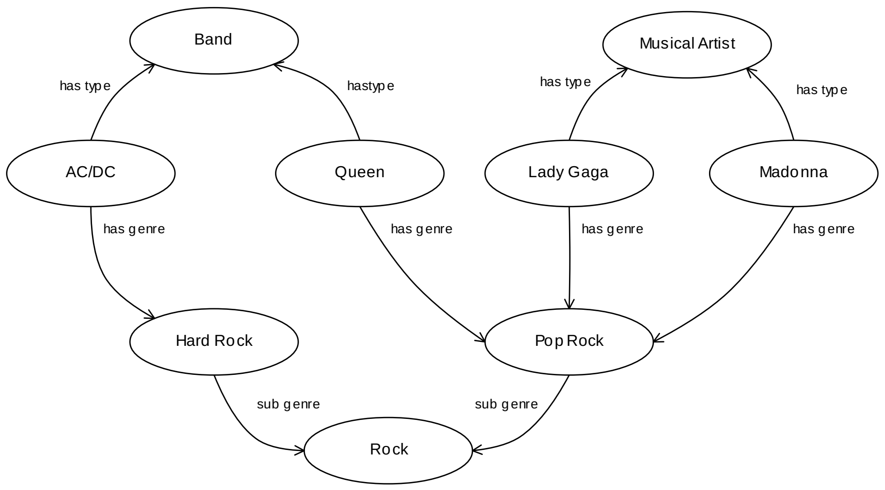

**Fig. 1.** RDF graph for concepts in music domain

In the proposed semantic relatedness measure, we consider *proximity* rather than distance as a measure of relatedness among nodes. By definition[3], proximity is closeness; the state of being near as in space, time, or relationship. Rather than focusing solely on minimum path length, proximity balances also the number of existing paths between nodes. As an example consider the proximity between two persons. More than resulting from a single common interest, however strong, it results from a collection of common interests.

With this notion of proximity, Lady Gaga and Madonna are more related to each other than with Queen since they have two different paths connecting each other, one through `Musical Artist` and another `Pop Rock`. By the same token the band Queen is more related to them than to the band AC/DC.

Our insight is that an algorithm to compute proximity must take into account all the relevant paths connecting two nodes and their weights. However, paths are made of several arcs, and the weight of an arc type should contribute less to proximity as it is further away in the path. In fact, there must be a limit in number of arcs in a path, as semantic graphs are usually connected graphs.

In this methodology, the estimation of a proximity value depends on exploring a graph. In a semantic graph a node can be linked to other node by several typed arcs. Although semantic graphs are usually characterized as directed multigraphs[4], for the purpose of this semantic measure the semantic graph can be interpreted as an undirected graph since, in

[3] `https://en.wiktionary.org/wiki/proximity`

[4] Take as example the RDF data model

general, it is possible to define an inverse relationship for each arc type in the graph. A domain graph can be defined as $G = (V, E, T, W)$ where $V$ is the set of nodes (concepts), $E$ is the set of edges connecting the nodes, $T$ is a set of edges types and $W$ is a mapping of edges types to weight values. Each edge in $E$ is a triplet $(u, v, t)$ where $u, v \in V$ and $t \in T$.

The set $W$ defines a mapping $w : T \mapsto \mathbb{N}^+$ and the upper bound of weights for all types is

$$\Omega(G) \equiv max_{t_i \in T} w(t_i)$$

In order to retrieve the proximity between two concepts it is necessary to walk through the graph and build a set of distinct paths that connect them. A path $p$ of size $n \in \mathbb{N}^+$ is a sequence of unrepeated nodes in $\{u_0 \ldots u_n | (\forall i, j, n)((0 \leq i \leq j \leq n) \wedge (u_i \neq u_j))\}$ that are linked by typed arcs, that must have at least one edge and cannot have loops; paths are denoted as follows.

$$p = u_0 \xrightarrow{t_1} u_1 \xrightarrow{t_2} u_2 \ldots u_{n-1} \xrightarrow{t_n} u_n$$

The weight of a path $p$ is the sum of the weights of each edge and the weight of an edge depends on its type, $\omega(p) = w(t_1) + w(t_2) + \ldots + w(t_n)$. The set of all paths of size $n$ that connect two concepts is defined as follows.

$$P^n_{u,v} = \{u_0 \xrightarrow{t_1} u_1 \ldots u_{n-1} \xrightarrow{t_n} u_n : u = u_o \wedge v = u_n \wedge \forall_{i,j,n} 0 \leq i \leq j \leq n \wedge u_i \neq u_j\}$$

The weight of $P^n_{u,v}$ is the sum of all sub paths that connects $u$ and $v$. The algorithm used to calculate proximity is based on this definition but also considers the path length. The shorter the path the higher is its contribution to the proximity.

The proximity function $r$ is defined by the following formula, where $\Delta$ is the graph maximum degree.

$$r(u, v) = \begin{cases} 1 & \leftarrow u = v \\ \frac{1}{\Omega(G)} \sum_{n=1}^{\infty} \frac{1}{2^n n \Delta(G)^n} \sum_{p \in P^n_{u,v}} \omega(p) & \leftarrow u \neq v \end{cases} \tag{1}$$

Given a graph with a set of nodes $V$

$$r : V \times V \mapsto [0, 1]$$

The proximity function $r$ takes two nodes and returns a "percentage" of proximity between then. This means that the proximity of related nodes must be close to 1 and the proximity of unrelated nodes must be close to 0.

The main issue with this definition of proximity[5] is how to determine the weights of transitions. The first attempt was to define these weights using domain knowledge. For instance, when comparing musical performers one may consider that being associated with a band or with another artist is more important than their musical genre, and that genre is more important than their stylistics influences and even more important than instruments they play.

However this naïve approach to weight setting raises several problems. Firstly, this kind of "informed opinion" frequently has no evidence to support it, and sometimes is

[5] See [16] for a detailed description of the algorithm.

plainly wrong. How sure can one be that stylistics influences should weight more than the genre in musical proximity? Even if it is true sometimes, how can one be sure it is true in most cases? Secondly, this approach is difficult to apply to a large ontology encompassing a broad range of domains. Is a specialist required for every domain? How should an ontology be structured in domains? What domain should be considered for concepts that fall in multiple domains? To be of practical use, the weights of a proximity based semantic relatedness algorithm must be automatically tuned.

## 4. Multiscale Weight Tuning

This section describes an approach for tuning weights in a family of semantic relatedness algorithms. The proposed tuning approach operates at different scales. Although each scale has its own distinctive features, there are self-similar patterns common to all scales.

In this tuning approach, the common pattern is the concept of function, with input and output values, and a set of parameters that can be tuned. At the lowest scale this function computes the quality of a semantic relatedness algorithm. It takes as input the parameters of semantic relatedness algorithm and produces a numeric value, a semantic relatedness measure. This function takes as parameters a semantic graph and a set of weights that must be tuned.

Zooming out to the next scale, there is a genetic algorithm. A genetic algorithm can be seen as a function taking as input a fitness function and producing a result. It has also its own set of parameters that need to be tuned such as the number of generations and the mutation rate. In this case the fitness function takes as input a set of weights and computes its quality. Hence, each application of a genetic algorithm (seen as a function) aggregates thousands of applications of functions from the previous scale.

Continuing to zoom out to the next and final scale there is a statistical method, the bootstrapping method. This method measures the accuracy of the results obtained by the genetic algorithm with different parameter sets. It can also be seen as a function taking as input genetic algorithms, the candidates for producing a weight tuning, and producing an estimate of which is the best. Again, each application of the bootstrapping method (seen as a function) aggregates thousands of applications of the functions from the previous scale – the genetic algorithm – since each candidate configuration set is repeated hundreds of times. The following subsections detail each of these “scales” and describes the function that is used as input for its upper scale, identifying its parameters.

### 4.1. Quality Measure

The purpose of the inner layer of this weight tuning procedure is to compute the quality for a particular set of parameters of a semantic measure defined by (1). This definition resulted from previous work that originally assumed non-negative weights values. During the research described in this paper, it was noticed that better results are obtained if negative values are assigned to some weights, which lead to minor changes in the definition of the proximity function. To simplify the definition of the tuning process and optimize its implementation, it was also necessary to factor out weights from the proximity definition. This is achieved by using matrix and vector notation and results also in a terser definition. Finally, it was possible to define the quality of semantic relatedness measure as function of a set weight attributions. Each of the following subsections details these steps to define the quality of a semantic related measure.

**Negative weight values** To cope with the negative weight values a few adjustments must be made to the original definition. First of all the upper bound $\Omega(G)$, must be redefined using the absolute value of all weights.

$$\Omega(G) \equiv max_{t_i \in T} |w(t_i)|$$

The purpose of this upper bound is to constrain the semantic measure to a known interval (originally $[0,1]$). Since the definition in (1) relies on an infinite series one must ensure that it converges. Given that if $u \neq v$ then $\sum_{p \in P_{u,v}}^{n} w(p) \leq \Omega(G) n \Delta(G)^n$ hence $r(u,v) \leq \sum_{n=1}^{\infty} 1/2^n = 1$. For non-negative weight values it is trivial that $r(a,b) \geq 0$. With negative weights and considering the redefinition of $\Omega(G)$ results that $\sum_{p \in P_{u,v}}^{n} w(p) \geq -\sum_{p \in P_{u,v}}^{n} |w(p)| \geq -\Omega(G) n \Delta(G)^n$ hence the lower bound is $r(u,v) \geq -\sum_{n=1}^{\infty} 1/2^n = -1$. Thus, with negative weights and the new definition of $\Omega(G)$ the proximity function $r$ is now defined as

$$r : V \times V \mapsto [-1, 1]$$

It would be trivial to adjust the definition of function $r$ to maintain its original codomain. However, since the codomain itself is not important, as far as it has known boundaries, it was decided to preserve the original definition given in (1).

**Factoring out weights** The first step is to express proximity in terms of weights of *arc types*. Consider the set of all arc types $T$ with $\sharp T = m$ and the weights of its elements $w(t), \forall_t \in T$. The second branch of (1) can thus be rewritten as follows, where $c_i(a,b), i \in \{1..m\}$ are the coefficients of each arc type.

$$r(a,b) = \alpha \sum_{n_1}^{\infty} \beta \sum_{P_j \in \mathbb{P}} \sum_{t \in P_j} w(t) = \sum_{i=1}^{m} c_i(a,b) \cdot w(t_i)$$

It should be noted that the weights of transitions types are independent of arguments of $r$ but the coefficients that are factored out depend of these arguments. By defining a standard order on the elements of $T$ both the weights of transitions and their coefficients are representable as vectors, respectively $(w(t_1), w(t_2), \ldots, w(t_k)) = \boldsymbol{w}$ and $(c_1(a,b), c_2(a,b), \ldots, c_m(a,b)) = \boldsymbol{c}(a,b)$. This way the previous definition of $r$ may take as parameter the weight vector as follows

$$r_{\boldsymbol{w}}(a,b) = \boldsymbol{c}(a,b) \cdot \boldsymbol{w} \tag{2}$$

**Quality as function of weights** The usual method for estimating the quality of a semantic relatedness function is to compare it with a benchmark data set. The benchmark data set contains pairs of words and their relatedness.

The *Spearman's rank order coefficient* is commonly used to compare the relatedness values in the benchmark data set with those produced by a semantic relatedness algorithm.

Rather than the simple correlation between the two data series, the Spearman's rank order sorts those data series and correlates their rank.

Consider a benchmark data set with the pairs of words $(a_i, b_i)$ for $1 \leq i \leq k$, with a proximity $x_i$. Given the relatedness function $r_{\boldsymbol{w}} : S \times S \mapsto \Re$ let us define $y_i = r_{\boldsymbol{w}}(a_i, b_i)$. In order to use the Spearman's rank order coefficient both $x_i$ and $y_i$ must be converted to the ranks $x'_i$ and $y'_i$. The quality of the function is given by the Spearman's rank order $\rho$ defined as

$$\rho = \frac{\sum\limits_i (x'_i - \overline{x'})(y'_i - \overline{y'})}{\sqrt{\sum\limits_i (x'_i - \overline{x'})^2 \sum\limits_i (y'_i - \overline{y'})^2}}$$

The Spearman's rank order coefficient is thus defined in terms of $x_i$ and $y_i$, where $x_i$ are constants from the benchmark data set. To use this coefficient as the fitness function it must be expressed as a function of $\boldsymbol{w}$. Considering that $\boldsymbol{y} = (r_{\boldsymbol{w}}(a_i, b_i), \ldots, r_{\boldsymbol{w}}(a_n, b_n))$ then $\boldsymbol{y} = C\boldsymbol{w}$ where $C$ and $\boldsymbol{w}$ are the following matrices.

$$C = \begin{bmatrix} c_{t1}^{a_1 b_1} & \ldots & c_{tm}^{a_1 b_1} \\ & \ldots & \\ & \ldots & \\ c_{t1}^{a_n b_n} & \ldots & c_{tm}^{a_n b_n} \end{bmatrix} \qquad \boldsymbol{w} = \begin{bmatrix} w(t_1) \\ \ldots \\ \ldots \\ w(t_m) \end{bmatrix}$$

Matrix $C$ is a $n \times m$ matrix where each line contains the coefficients for a pair of concepts and each column contains coefficients of a single edge (transition) type. Vector $\boldsymbol{w}$ is a $m \times 1$ matrix with the weights assigned to each edge type. The product of these matrices is the proximity measure of a set of concept pairs.

Considering $\rho(\boldsymbol{x}, \boldsymbol{y})$ as the Spearman's rank order of $\boldsymbol{x}$ and $\boldsymbol{y}$, the quality function $q : \Re^n \mapsto \Re$ using the benchmark data set $\boldsymbol{x}$ can be defined as

$$q_{\boldsymbol{x}}(\boldsymbol{w}) = \rho(\boldsymbol{x}, C\boldsymbol{w}) \tag{3}$$

The next step in this tuning methodology is to determine a $\boldsymbol{w}$ that maximizes this quality function.

## 4.2. Genetic Algorithm

Genetic algorithms are a family of computational models that mimic the process of natural selection in the evolution of the species. These algorithms use the concepts of *variation*, *differential reproduction* and *heredity* to guide the co-evolution of a set of problem solutions. This type of algorithm is frequently used to improve solutions of optimization problems [33].

There are three necessary conditions for using a genetic algorithm. Firstly, the different candidate solutions must be representable as individuals (*variation*). This encoding of an individual solution is sometimes called a *chromosome* which are a collection of *genes* that characterize the solution. Secondly, it must be possible to compare a set of individuals, decide which are the fittest and allow them to pass their genetic information to the next generation (*differential reproduction*). At last, the representation of solutions as individuals must allow their recombination with other solutions (*heredity*) so that favorable traits are preferred over unfavorable ones as the population of solutions evolves.

In this weight tuning approach, the *individual* is a vector of weight values. Consider a sequence of weights $w(t_1), w(t_2) \dots w(t_k)$ (the genes) taking values in natural numbers up to a certain limit in a standard order of arc types. Two possible solutions are the vectors $\boldsymbol{v} = (v_1, v_2, .., v_k)$ and $\boldsymbol{t} = (t_1, t_2, \dots, t_n)$. They are easy to recombine by crossover, taking "genes" from both "parents" (e.g. $\boldsymbol{u} = (v_1, t_2, \dots t_{n-1}, v_k)$).

This representation contrasts with the binary representations typically used in genetic algorithms [9]. However it is closer to the domain and it can be processed more efficiently with large number of weights.

Genetic algorithms introduce variance also by *mutation*. There is a number of mutation operators, such as swap, scramble, insertion, that can be used on binary representations [9]. However, the approach taken to represent individuals in this methodology makes these operators inadequate. Since weights are independent from each other, swapping values among them is as likely to improve the solution as selecting a new random value. Hence, the genetic algorithm created for tuning weights has a single kind of mutation: randomly selecting a new value for a given "gene".

The fitness function plays a decisive role in selecting the new generation of individuals, created by crossover and mutation of their parents. In this case individuals are a vector of weight values $\boldsymbol{w}$, hence the fitness function is in fact the quality function defined by equation (3).

## 4.3. Bootstrapping

The genetic algorithm itself has a number of parameters that must be tuned. Generic parameters of a genetic algorithm include the number of generations and the mutation rate. In this particular case the range of values that may be assigned to weights must also be considered.

Several approaches to tuning parameters of genetic algorithms have been proposed and compared [29]. Although with different approaches, these methods highlight the advantage of using automated parameter tuning over tuning based on expert "informed opinions". In many cases the best solution contradicts the expert best intuitions.

The proposed approach relies on a single benchmark data set to compare alternative weight attributions. To repeat a large number of experiments using the genetic algorithm to co-evolve a set solutions one needs a larger number test samples. Bootstrapping [8] is a statistical method for assigning measures of accuracy to data samples using a simple technique known as *resampling* that solves this issue.

Resampling is applied to the original data set to build a collection of sample data sets. Each sample data set has the same size as the original data set and is built from the same elements. If the original data set has size $n$ then $n$ elements from that set are randomly chosen to create the sample set. When an element is selected it is not removed from the original data set. Hence, a particular element may occur repeatedly on the sample data set while other may not occur at all.

The bootstrapping method is used for comparing different combination of parameters. Each combination is repeated a large number of times, typically 200, each time with a different sample set, being summarized by a statistics, such as the mean or the third quartile. In the end, these statistics are compared to select the most effective combination. Since the objective is to select the set of parameters that may lead to the highest quality, the third quartile is specially relevant since it is a lower bound of the largest solutions.

In this tuning approach, each combination corresponds to a particular setting of the genetic algorithm. Candidate settings include values for parameters such as the number of generations or the mutation rate. Another important parameter that is specific to this approach is the range of values that are used as possible values for weights. As these values have to be enumerated, this procedure considers only integer values bellow a certain threshold.

As the result of the bootstrapping method a particular setting of the genetic algorithm's parameters is selected. The final stage is to run the genetic algorithm with these settings, using the full benchmark data set in the fitness function. The selected genetic algorithm is repeated an even larger number of times, typically 1000, and the best result is selected as weights for the relatedness algorithm.

## 5. Implementation

The approach presented for parameter tuning has a high computational complexity. At its core it has to find all paths connecting two concepts to compute a single proximity. To test the quality of a vector of weights, the proximity has to be computed for each pair of concepts in a benchmark data set. Bootstrapping repeats 200 times the genetic algorithm for each setting, and this process is repeated for a large number of settings.

The proximity measure takes two strings as labels and builds a set with all the paths that connect both terms. There is a label stemming process that explores the meaning of related words or expressions. For instance, when one of the compared term is "*book*", then nodes having as labels "*bookish*" and "*book shop*" may also be explored. The stemming process is a special kind of transition and may improve the quality of the relatedness measure, but also increases the computational complexity of the tuning process.

Figure 2 provides an overview of the proposed semantic relatedness methodology. The process begins with the graph pre-processing. This action is not mandatory but is crucial to improve the efficiency, as detailed in the subsection 5.1.

The process itself is parametrized by a data set and begins with the bootstrap procedure that tests a user defined set parameters - mutation rate, number of generations and weight range - for different values. Each specific set of parameters is tested 200 times and is summarized by statistic measures that are decisive to choose the best set of parameters.

Each set of parameters is tested by the genetic algorithm. To generate a weight assignment, the genetic algorithm runs the number of generations defined by the parameters. This weight assignment is used by the proximity algorithm that returns the weight's quality. In the end, the best quality is selected.

A weight assignment is tuned by using the proximity algorithm. The semantic relatedness of each pair of words of the data set is tested using the proximity algorithm and its value is compared against the benchmark value using the Spearman's correlation, returning the quality of the given assignment.

The best set of parameters determined in the processing stage is used in the final stage of this methodology. Along with the data set, they parametrize the genetic algorithm, which is repeated 1000 times in order to obtain the best quality with those parameters.

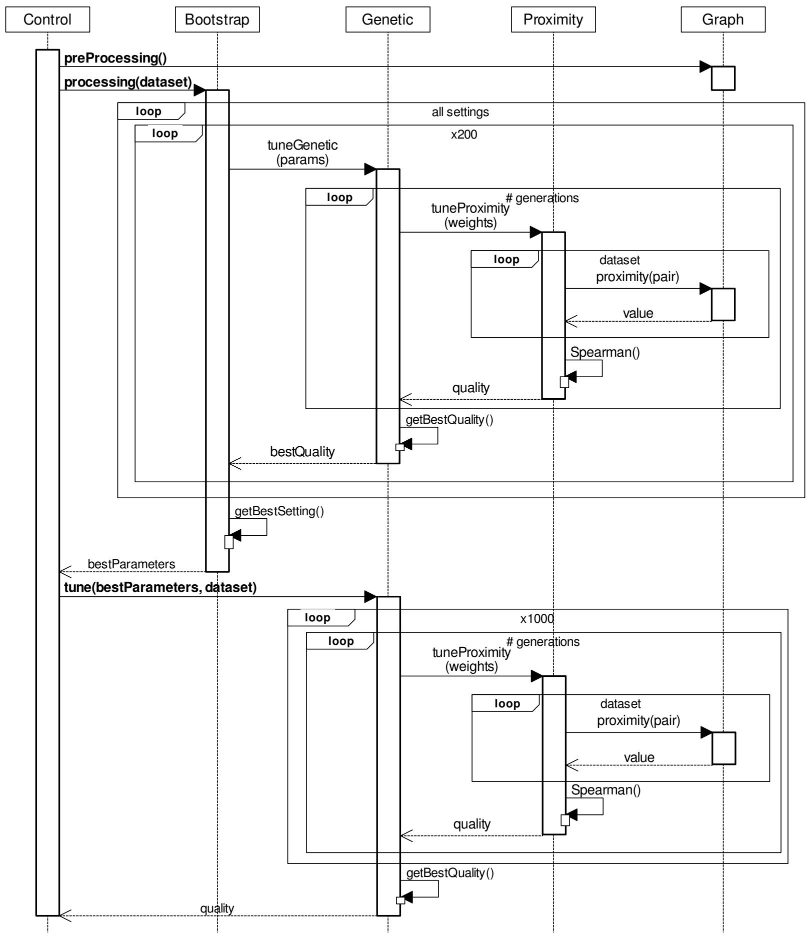

**Fig. 2.** Methodology diagram

The strategy used for improving the efficiency of the methodology has three main components: graph pre-processing, described in the subsection 5.1; factorization of the proximity algorithm and concurrent evaluation of the bootstrapping method, described in the subsection 5.2. The remainder of this section details each of these components.

## 5.1. Graph Pre-processing

The computation of the semantic proximity between two terms depends on a data graph search that finds all the paths that connect them. The data graph search is implemented in two different ways, supporting queries of remote and local data. The main difference is in the exploratory methods used by the semantic algorithm. One of these methods retrieves a set of nodes with a given label and the other method retrieves the transitions from a node.

Remote data is usually retrieved from SPARQL endpoints. A SPARQL endpoint is addressed by a URI to which SPARQL queries can be sent and which returns RDF as a response. Paths are built from data collected by the exploratory methods using SPARQL queries.

The initial step is to retrieve the set of nodes linked to each term of the process. Since this methodology estimates the semantic proximity between a pair of terms, this method is executed twice. For each node retrieved during all the process, the second exploratory method fetches the list of nodes that are connected to it. Whenever is possible to build a path between both terms using the expanded nodes and its transitions, that path is considered to measure the proximity. This traverse ends when there are no more nodes to explore or when the path size limit is reached.

This SPARQL approach raises a number of issues. Firstly, the endpoint or network may be under maintenance or with performance problems. Secondly, some endpoints have configuration problems and do not support queries with some operators, such as `UNION`. And thirdly, the SPARQL queries can have performance issues, mainly when using operators such as `DISTINCT`, and having a large amount of queries per proximity search can cause a huge impact at the execution time.

In order to avoid those issues, this semantic relatedness methodology implements searches in local data. Knowledge bases often provide dumps of their data. Local data are pre-processed semantic graphs that are stored in the local file system with data retrieved from those dumps. Graph pre-processing begins with parsing the RDF data. RDF data can be retrieved in several formats, such as Turtle, RDF/XML or N-Triples. To simplify this process, all RDF data is converted to N-Triples, since this is the simplest RDF serialization.

This process takes some time to execute but it is only necessary to execute once. Also, the most used data is cached in memory which has a significant impact on performance.

The computation of the proximity of a single pair of concepts using the WordNet 2.1 SPARQL endpoint[6] takes about 20 minutes. With the pre-processed graph[7] that is executed once and takes 30 minutes, the same computation takes about 6 seconds.

## 5.2. Other Optimizations

Traversing the graph searching for paths connecting two labels is the most frequently executed part of this semantic relatedness methodology. Nevertheless, this procedure is almost the same for each pair of concepts, varying only on the weights that are used for each arc type. This computation is repeated many times since the exact same pair of concepts is used each time that the genetic algorithm is run.

The solution found was to alter the proximity algorithm to compute the set of coefficients that are multiplied to each weight. These coefficients are organized in a vector, using the same order of the weight vector used in the genetic algorithm. Thus, computing the proximity of a pair of concepts given a different weight vector is just the inner product of the weight vector and the coefficient vector, as shown in (2).

With this modification a single run of the genetic algorithm with 200 generations takes less than 1 minute and computing the coefficients for all the pairs takes about 30 minutes.

The final optimization was concurrent evaluation of the bootstrapping method. Each of the settings can be processed independently, hence they could be assigned to a different

[6] wordnet.rkbexplorer.com/sparql/

[7] The tests were executed in a 8 core machine at 3.5 GHz and 16Gb of RAM

processor of a multi-core machine[7]. Each run of the bootstrapping method takes about 200 minutes. It run 120 configurations that sequentially would have taken more than 16.5 days in about 2 days.

## 6. Validation

The validation of the proposed tuning approach consisted of tuning the weights of the relatedness algorithm for WordNet 2.1.

The WordNet[8] [10] is widely used in most of the mentioned knowledge-based approaches [6, 15, 1, 21, 31]. It is a large lexical knowledge base for English words. It groups nouns, verbs, adjectives and adverbs into *synsets* (set of cognitive synonyms) that express distinct concepts. *Synsets* are interlinked by lexical and conceptual-semantic relationships. This knowledge-base is well-known but lacks some specialized vocabularies and named entites, such as Diego Maradona or Freddie Mercury. WordNet 2.1 is a comparatively small knowledge base with 26 different types of properties that interlink 464.795 nodes and thus it is ideal for the inital tests of a tuning methodology.

This tuning process used as benchmark the WordSimilarity-353 data set [11]. It has 353 pairs of concepts with the mean of the relatedness values given by humans. Since WordNet 2.1 does not have all the words listed in this data set, the pairs with missing elements were removed, creating a new data set with the non-missing pairs. In total 7 pairs were removed.

The tuning was performed in two rounds. In the first round a large number of settings was explored to determine which were the most relevant. A second round was then performed to explore a wider range of values on those parameters that have more impact on performance.

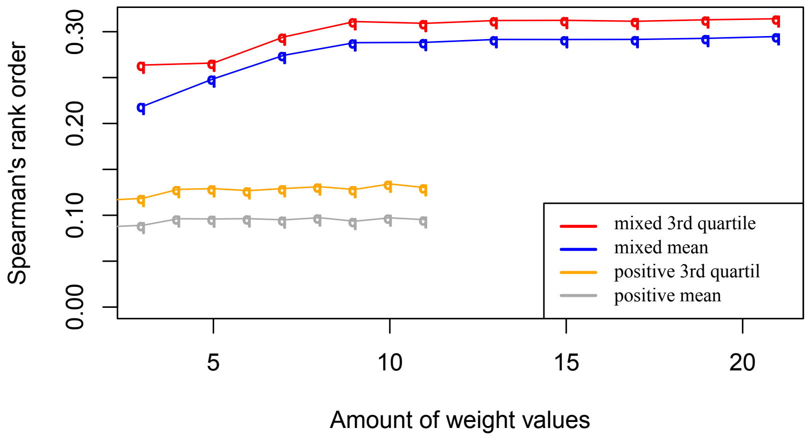

**Fig. 3.** Graph of weights distribution

[8] http://wordnet.princeton.edu/

In the first round the bootstrapping process tested three parameters: weight values, mutation rate, and number of generations. The weight values were divided in positive and mixed (positive and negative) values; the positive values ranged in $[0, n]$ with $n \in \mathbb{N}^+$ and $n \leq 10$. The mixed values ranged in $[-n, n]$ for the same values of $n$. The mutation rate took values in the set $\{0.3, 0.4, 0.5\}$ values and the number of generations in $\{100, 200\}$. Permutating these values, 120 different sets of parameters were tested. Each set of parameters was executed 200 times in the bootstrapping process. The results of those tests can be seen in the following graphs. These graphs show the statistics of the correlation as function of a single variable: number of weight values, mutation rate and number of generations.

Figure 3 shows how the correlation varies with different ranges of weight values. The correlation obtained when there are only positive values in the weight set is much lower than when positive and negative weights are used. The positive values also appear to reach a maximum value. However, the sets with positive and negative values do not show that stabilization, suggesting the need for more tests with a larger range of values. Third quartile and mean were used in order to verify if the variations were consistent.

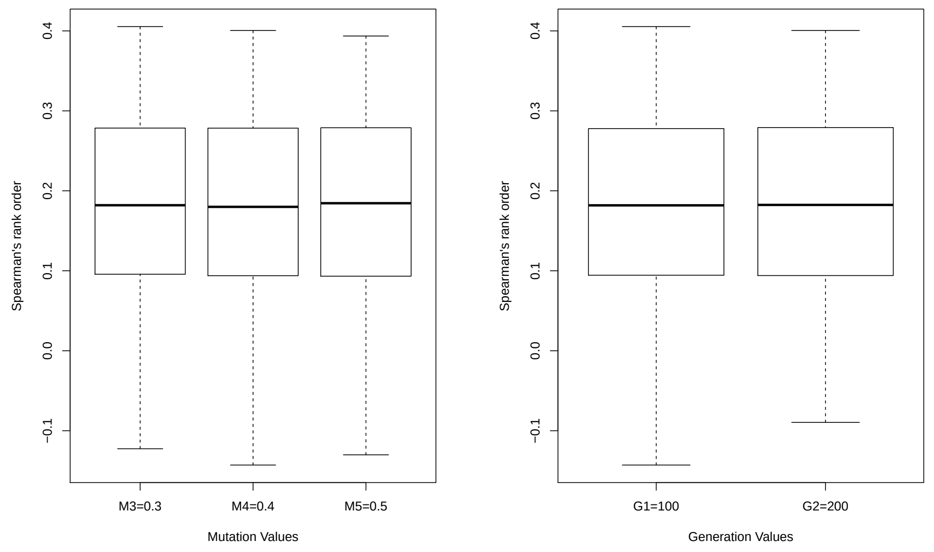

**Fig. 4.** Distribution of different mutation rates (left) and number of generations (right)

The graph on the left of Figure 4 shows the impact of changing the mutation rate. Despite the large overall variation, the mean and third quartile values are similar, showing that variations in this parameter have a small impact on the tuning process. Still, the variation of the maximums indicate the relevance of also testing lower mutation rates in the future.

The graph on the right of Figure 4 presents the variation of the number of generations. As it occurs with the mutation rate, changes in the number of generations have no significant impact in the correlation values.

After the first round, changes in range of weight values appear to have a higher impact in the correlation values, specially if they allow negative values, increasing the correlation as the range size increases. New tests were conducted to investigate for how long the correlation continues to increase, if it converges to an asymptote, or if the correlation degrades after a certain threshold.

A new round of tests was made to investigate these hypothesis. This time only positive and negative values were used, with fixed values of mutation rate and number of generations. These new configurations uses ranges from $[-10 \times n, 10 \times n]$ with $n \in \mathbb{N}^+$ and $n \leq 10$. The mutation rate value was fixed at 0.4 and the number of generations was fixed at 200. The results are displayed in Figure 5. The values obtained by increasing the range of values show a maximum value at the range [-20,20]. Ranges with higher values seem to never exceed the Spearman's rank order obtained at that point, indicating that performance degrades after this threshold.

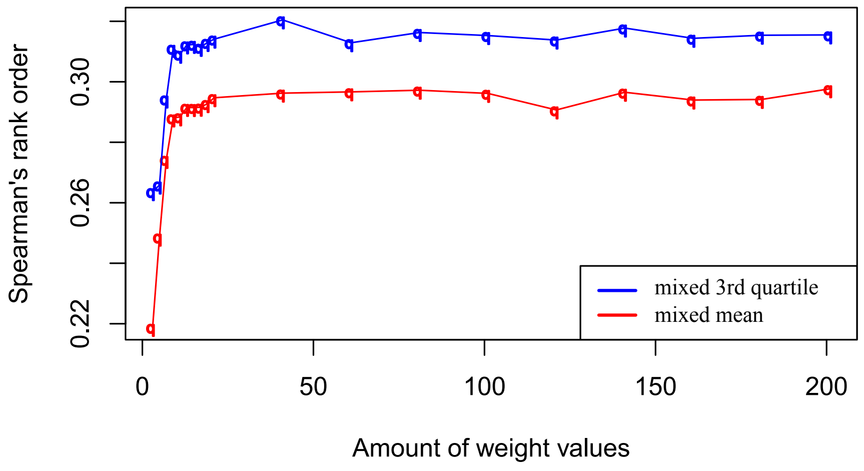

**Fig. 5.** Graph of weights distribution

Using the best configuration obtained by the bootstrap process the genetic algorithm was executed 1000 times aiming to obtain the best correlation value and the related configuration.

The best Spearman's rank order value obtained was 0.409 and the corresponding weight set is listed in the Table 1. The arcs with the prefix `wn` correspond to the WordNet 2.1 URI[9] and the prefix `rdfs` to the RDF Schema URI[10]. The arc type `:stemming` is the custom arc created in the stemming process.

Table 2 compares the results obtained by our semantic measure with weights tuned by the proposed methodology with other similar semantic measures described in the literature, that use also the WordNet 2.1 as semantic graph and that validate their results using WordSim-353.

[9] `http://www.w3.org/2006/03/wn/wn20/schema/`

[10] `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`

**Table 1.** Weight values obtained after tuning process

| Edge type | Weight | Edge type | Weight |
|---|---|---|---|
| :stemming | 1 | wn:classifiedByUsage | 18 |
| wn:memberMeronymOf | -2 | wn:tagCount | 1 |
| wn:participleOf | -6 | wn:sameVerbGroupAs | -8 |
| wn:antonymOf | -5 | wn:derivationallyRelated | 0 |
| wn:classifiedByTopic | 19 | wn:attribute | -15 |
| wn:partMeronymOf | 19 | wn:synsetId | 18 |
| wn:word | 12 | wn:seeAlso | 6 |
| wn:gloss | 19 | rdfs:type | -2 |
| wn:similarTo | -19 | entails | -1 |
| wn:containsWordSense | 4 | wn:classifiedByRegion | -9 |
| wn:causes | -17 | wn:adverbPertainsTo | 12 |
| wn:frame | 7 | wn:hyponymOf | 9 |
| wn:adjectivePertainsTo | 3 | wn:substanceMeronymOf | 18 |

**Table 2.** Previous work with WordNet and WordSim-353

| Method | Spearman's rank order |
|---|---|
| Jarmasz (2003) | 0.33 - 0.35 |
| Strube and Ponzetto (2006) | 0.36 |
| **Proposed methodology** | **0.41** |

## 7. Conclusion

The major contribution of the research presented in this paper is an approach for tuning a relatedness algorithm to a particular semantic graph, without requiring any domain knowledge on the graph itself. The results obtained with this approach for WordNet 2.1 are better than the state of the art for the same graph. A number of solutions to speedup graph processing and the evaluation of fitness functions are also relevant contributions.

The proposed approach performs a multiscale parameter tuning of a graph based semantic relatedness algorithm. The main feature of the base algorithm is the fact that it considers all paths that connect two labels and computes the contribution of each path as a function of its length and of the arc (properties) types. The main issue with this algorithm is the selection of parameters (weight values for each arc type) that maximize the quality of the relatedness algorithm.

The quality of semantic relatedness algorithms is usually measured against a benchmark data set. This data set consists of the relatedness of a set of words, defined as the mean of the relatedness attributed by a group of persons. In this approach the quality of the algorithm is computed as the Spearman's rank correlation coefficient between the relatedness produced by the algorithm and the relatedness given by the data set. Evolutionary algorithms in general, and genetic algorithms in particular, are popular choices for improving the quality of solutions. The proposed approach takes advantage of the genetic algorithms' use of variation and selection to improve the quality of the semantic relatedness algorithm. The statistical method of bootstrapping is used in this approach to measure the accuracy of different parameter settings used in the genetic algorithm.

The proposed approach for tuning the parameters of the semantic relatedness algorithm was validated with Wordnet 2.1. The tuning procedure was actually executed twice. In the first run several parameters of the genetic algorithm were tested to conclude that the range of weight values is the decisive parameter, in particular if it is allowed to contain negative values. The variation of some of the parameters, such as mutation rate and number of generations, had no impact on the quality. Based on these findings a second run of the tuning procedure explored a wider range of values. It showed that quality improves with the width of range values but also that a small degradation occurs after a certain threshold. The genetic algorithm was finally repeated a large number of times with the settings selected by this approach and the maximum Spearman's correlation obtained is significantly higher than the best result reported on the literature for the same graph.

The Wordnet 2.1 graph used for the evaluation is comparatively small. It has just 26 different types of properties and 464.795 nodes. The next step is to investigate how this approach works with Wordnet 3.0 and with even larger graphs, such as the DBPedia or Freebase. Apart from the challenges of dealing with such large graphs, it will be interesting to compare the semantic relatedness potential of different graphs and try to combine them to improve the accuracy of the semantic relatedness algorithm.

**Acknowledgments.** This work is financed by the ERDF European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT — Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-037281. Project "NORTE-07-0124-FEDER-000059" is financed by the North Portugal Regional Operational Programme (ON.2 - O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by national funds, through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT). The authors would also like to thank Luis Torgo for his help in the use of statistical methods.

## References

1. Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., Soroa, A.: A study on similarity and relatedness using distributional and wordnet-based approaches. In: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics. pp. 19–27. Association for Computational Linguistics (2009)
2. Banerjee, S., Pedersen, T.: An adapted lesk algorithm for word sense disambiguation using wordnet. In: Computational linguistics and intelligent text processing, pp. 136–145. Springer (2002)
3. Banerjee, S., Pedersen, T.: Extended gloss overlaps as a measure of semantic relatedness. In: IJCAI. vol. 3, pp. 805–810 (2003)
4. Bodenreider, O., Aubry, M., Burgun, A.: Non-lexical approaches to identifying associative relations in the gene ontology. In: Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing. p. 91. NIH Public Access (2005)
5. Bollegala, D., Matsuo, Y., Ishizuka, M.: Measuring semantic similarity between words using Web search engines. Proceedings of the 16th international conference on World Wide Web 7, 757–766 (2007)
6. Budanitsky, A., Hirst, G.: Evaluating wordnet-based measures of lexical semantic relatedness. Computational Linguistics 32(1), 13–47 (2006)

7. Dagan, I., Lee, L., Pereira, F.C.: Similarity-based models of word cooccurrence probabilities. Machine Learning 34(1-3), 43–69 (1999)
8. Efron, B., Tibshirani, R.J.: An introduction to the bootstrap, vol. 57. CRC press (1994)
9. Eiben, A.E., Smith, J.E.: Introduction to evolutionary computing. Springer (2003)
10. Fellbaum, C.: WordNet. Wiley Online Library (1999)
11. Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E.: Placing search in context: The concept revisited. In: Proceedings of the 10th international conference on World Wide Web. pp. 406–414. ACM (2001)
12. Ganesan, P., Garcia-Molina, H., Widom, J.: Exploiting hierarchical domain structure to compute similarity. ACM Transactions on Information Systems (TOIS) 21(1), 64–93 (2003)
13. Gorodnichenko, Y., Roland, G.: Understanding the individualism-collectivism cleavage and its effects: Lessons from cultural psychology. Institutions and Comparative Economic Development 150, 213 (2012)
14. Harris, Z.S.: Distributional structure. In: Papers on syntax, pp. 3–22. Springer (1981)
15. Jarmasz, M.: Roget's thesaurus as a lexical resource for natural language processing. CoRR abs/1204.0140 (2012)
16. Leal, J.P.: Using proximity to compute semantic relatedness in rdf graphs. Comput. Sci. Inf. Syst. 10(4) (2013)
17. Leal, J.P., Costa, T.: Multiscale parameter tuning of a semantic relatedness algorithm. In: 3rd Symposium on Languages, Applications and Technologies, SLATE. pp. 201–213 (2014)
18. Li, Y., Bandar, Z.A., McLean, D.: An approach for measuring semantic similarity between words using multiple information sources. Knowledge and Data Engineering, IEEE Transactions on 15(4), 871–882 (2003)
19. Li, Y., McLean, D., Bandar, Z.A., O'shea, J.D., Crockett, K.: Sentence similarity based on semantic nets and corpus statistics. Knowledge and Data Engineering, IEEE Transactions on 18(8), 1138–1150 (2006)
20. Lin, D.: An information-theoretic definition of similarity. In: ICML. vol. 98, pp. 296–304 (1998)
21. Mazuel, L., Sabouret, N.: Semantic relatedness measure using object properties in an ontology. In: The Semantic Web-ISWC 2008, pp. 681–694. Springer (2008)
22. Patwardhan, S., Banerjee, S., Pedersen, T.: Using measures of semantic relatedness for word sense disambiguation. In: Computational linguistics and intelligent text processing, pp. 241–257. Springer (2003)
23. Pirró, G.: A semantic similarity metric combining features and intrinsic information content. Data & Knowledge Engineering 68(11), 1289–1308 (2009)
24. Pirró, G., Euzenat, J.: A feature and information theoretic framework for semantic similarity and relatedness. In: The Semantic Web–ISWC 2010, pp. 615–630. Springer (2010)
25. Pirró, G., Seco, N.: Design, implementation and evaluation of a new semantic similarity metric combining features and intrinsic information content. In: On the Move to Meaningful Internet Systems: OTM 2008, pp. 1271–1288. Springer (2008)
26. Rada, R., Mili, H., Bicknell, E., Blettner, M.: Development and application of a metric on semantic nets. Systems, Man and Cybernetics, IEEE Transactions on 19(1), 17–30 (1989)
27. Ranwez, S., Ranwez, V., Villerd, J., Crampes, M.: Ontological distance measures for information visualisation on conceptual maps. In: On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops. pp. 1050–1061. Springer (2006)
28. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: IJCAI. pp. 448–453 (1995)
29. Smit, S.K., Eiben, A.E.: Comparing parameter tuning methods for evolutionary algorithms. In: Evolutionary Computation, 2009. CEC'09. IEEE Congress on. pp. 399–406. IEEE (2009)
30. Stojanovic, N., Maedche, A., Staab, S., Studer, R., Sure, Y.: Seal: a framework for developing semantic portals. In: Proceedings of the 1st international conference on Knowledge capture. pp. 155–162. ACM (2001)

31. Strube, M., Ponzetto, S.P.: Wikirelate! computing semantic relatedness using wikipedia. In: AAAI. vol. 6, pp. 1419–1424 (2006)
32. Terra, E., Clarke, C.L.: Frequency estimates for statistical word similarity measures. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1. pp. 165–172. Association for Computational Linguistics (2003)
33. Whitley, D.: A genetic algorithm tutorial. Statistics and computing 4(2), 65–85 (1994)

**José Paulo Leal** is assistant professor at the department of Computer Science of the Faculty of Sciences of the University of Porto (FCUP) and associate researcher of the Center for Research in Advanced Computing Systems (CRACS). His main research interests are semantic web, automated evaluation and recommender systems.

**Teresa Costa** is a Ph.D. candidate in Computer Science and received the M.Sc. degree in Network and Information Systems with specialization in Communication Networks from Faculty of Sciences of University of Porto. Currently she is a researcher at Center for Research in Advanced Computing Systems (CRACS). Her main research interests are related with Semantic Web and Linked Data.

*Received: September 5, 2014; Accepted: May 21, 2015.*