# Assigning-Tasks Method for Developers in Software Projects Using up Similarity Coefficients

Sergio Ruiz(✉), Daniel Escudero, Jair Cervantes, and Adrián Trueba

Universidad Autónoma del Estado de México, Atlacomulco, Estado de México, Mexico
jsergioruizc@gmail.com, piedraa0@hotmail.com, chazarra17@gmail.com, atruebae@yahoo.com

**Abstract.** In the Software industry, big software projects are carried out with hundreds of developers. The fast change in technologies and development environments increase the complexity. Usually, there are project teams with a project leader. However, it is very difficult to know the profile of each developer. The development tasks also have their profile. Hence, it is necessary to assign each task to the most suitable developer. The erroneous assigning of tasks can cause delays and increase the project costs. Thus, a bad assigning of tasks can cause stress and low productivity. Therefore, we make a proposal to enhance tasks assignment to developers regarding the task and developer profiles. The task profile includes characteristics such as: knowledge, kind of task, complexity, experience, etc., in other aspects as codification: paradigm, programming language, version, etc. Using algorithms with similarity coefficients, we look for the best match to assign the tasks to developers. In this work, we used five techniques of similarity coefficient in order to find results and to recommend the best solution for this problem. We conclude that with the Sokal and Sneath technique we obtain better results to solve the problem.

**Keywords:** Matching · Software industry · Similarity coefficient · Profile of tasks · Profile of developer

## 1 Introduction

The reason that motivated the development of this work is the complexity that exists on assignment of tasks to the developers; mostly, in organizations that develop large and complex projects. The developers have different capabilities and abilities, called Profile of Developer (PD). Therefore, there are increasingly complex tasks with different characteristics that require new specialists, called Profile of Task (PT). Additionally, at least in Mexico, there is a high rotation of personal in the software project development companies.

The objective of this work is to create a method to support the assignment of tasks to the developers by the leaders or project managers. To fulfill the method, it is necessary for the project leader to generate a profile of each one of the developers. Then, when a new task is created, you must generate your profile. Our Model created

J.C. Figueroa-García et al. (Eds.): WEA 2017, CCIS 742, pp. 119–128, 2017.
DOI: 10.1007/978-3-319-66963-2_12

for this purpose will do the matching by finding the most suitable developer for the task in question.

The proposed method for this work is the use of similarity coefficients to find the greatest similarity between PD and PT. Five similarity coefficients are used to constrain the results and determine the best alternative. An application has been created to help the project leader. This method is intended to achieve the assignment of tasks more optimally.

## 2 Similarity Coefficients

The similarity coefficients are more prolific than the similarity indexes. These coefficients are used to measure the association between samples. In contrast, with most distance coefficients, similarity measures are not metric, since two, *A* and *B*, samples may be found to be more similar to each other than the sum of their similarities to a distant *C* sample. Similarities cannot be used to locate samples relative to one another in a symmetric space. Also, they can be defined as those that measure the data of identical pairs between pairs of operative taxonomic units on a multi-state double character array.

The coefficient is a measure of similarity that satisfies the requirements of a scalar product between normalized vectors in a Euclidean space, which is called correlation coefficient.

The similarity coefficients are developed for binary data, of type presence-absence or 1-0. For binary data they are derived as a ratio that implies the number of attributes shared by a pair of entities related to the number of attributes involved in the comparison. In most of these coefficients, the values range from 0 (no similarity) and 1 (complete similarity) [1].

In binary coefficients, [2] affirm it is simple; the similarity between two samples is based on the presence or absence of certain characteristics in them. These descriptors may be environment variable or species. Therefore, the observations can be tabulated in a frequency table of 2 × 2. See Table 1.

**Table 1.** Frequency table of 2 × 2 [2].

| | Sample X1 | | | |
|---|---|---|---|---|
| Sample X2 | | 1 | 0 | |
| | 1 | 1, 1 (a) | 1, 0 (b) | a + b |
| | 0 | 0, 1 (c) | 0, 0 (d) | c + d |
| | | a + c | b + d | (n) = a + b + c + d |

From the Table 1, it gets the variables: *(a), (b), (c)* y *(d)*. The *(a)* when both values are 1's, *(b)* when the values are 1 and 0, *(C)* when the values are 0 and 1, *(d)* when both values are 0's. Finally, *(n)* is the total number of occurrences [2].

The Simple Comparison Coefficient (SMC) establishes the similarity between the two samples by counting the number of variables they both possess and dividing by the total number of variables. See Formula 1. This coefficient assumes that there is no difference between double zero and double one, so that the variable or attribute can get the value of zero or one indistinctly, this is giving the same value to absences or presences [2].

$$S_1(x_{1,},x_2) = \frac{a+d}{n} = \frac{a+d}{a+b+c+d} \tag{1}$$

The coefficient of Rogers and Tanimoto establishes that differences are more important than similarities. See Formula 2. This index produces values between zero and one [3].

$$D_2(x_1,x_2) = \frac{a+d}{a+2b+2c+d} \tag{2}$$

The Sokal and Sneath coefficient is proposed four measures which take into account the double zeros, See Formula 3. In this coefficient the joint occurrences have double value. This index provides values from 0 to 1 [4].

$$S_3(x_1,x_2) = \frac{2a+2d}{2a+b+c+2d} \tag{3}$$

The Hamman coefficient is the simple equality index except that the numerator is reduced by inequalities. You can also find it by the name of Index G. See Formula 4. This index produces values between −1 and +1 [5].

$$S_4(x_1,x_2) = \frac{(\mathrm{a}+d)-(b+c)}{a+b+c+d} \tag{4}$$

Finally, the Jaccard coefficient gives the same value to all terms, and ranges from 0 to 1. See Formula 5 [6].

$$S_5(x_1,x_2) = \frac{a}{a+b+c} \tag{5}$$

The coefficient of similitude has been used in bio-informatics, pattern recognition, signal processing, file comparison and text correction. With the Distance of Levenshtein to know the distance between two words [7]. See the Table 2. The distance is 4, because among both words there are 4 different characters.

**Table 2.** Distance of Levenshtein [7].

| U | N | I | V | E | R | S | A | L | L | Y |
|---|---|---|---|---|---|---|---|---|---|---|
| = | = | = | = | = | = | = | x | x | x | x |
| U | N | I | V | E | R | S | E | – | – | – |

In this case, the distance is determined by the number of letters that need to be added, deleted or changed for getting two words are equal. When the two words or phrases are equal, then the distance is 0 [7].

We find the work of [8], where similarity metrics are applied for case-based project management, to find previous historical cases that provide knowledge.

The work of [9] seeks to detect the degree of similarity over short texts. For this, two different approaches are proposed to carry out this task. On one hand, they propose unsupervised methods based on algorithms that were originally created to solve the task of biological chain alignment. These algorithms are made to recognize a pair of sequences and find the main similarities between them. These algorithms are given semantic information to be used, not biological sequences, but in texts.

Another work [10] seeks to find similarity between leaves of plants from 150 characteristics. From a vector characteristic of a new leaf, it is compared with a set of 22767 vectors of the identified plants. The objective is to identify the new plant based on the characteristics of the leaf.

## 3 Proposed Method

### 3.1 Profile of Task

In this research, it is not intended to make a classification, but the application of a metric that allows finding the greatest similarity between the profile of a task and a developer.

Firstly, the PT was defined. The projects leader will create the profile of each task. Secondly, we have the profile of the task, we can search which developer has most similar profile and match them. The profile of some tasks can be seen in Table 3.

When the characterization exists the value is 1 and when the characterization does not exist the value is 0. Using a form, the project leader generates the task profiles, immediately it is captured on an application and the application makes the matching.

Thus, the first task has the next pattern: 01000100000001000010001010010100, and the second task: 00000100100100000000001000100101010. Then, each task has its own pattern.

### 3.2 Profile of Developer

The PD has 32 0's and 1's. The first developer has the next pattern: 00100110000000000010101010010101. Each developer will have their profile defined in the company and will be considered when assigning a task. See Table 4.

### 3.3 Choice of Similarity Metrics

When a new task is generated the Method for Matching Tasks and Developers (MMTD) compares the task with all the profiles of developers. Its aim is to find a developer who has greater similarity with the task. The task is assigned to the developer with the greatest similarity. See Table 5.

**Table 3.** Profiles of tasks.

| | Profile of tasks | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 1 | Requeriments management | 0 | 0 | 0 | 0 | 0 |
| 2 | Analysis of requeriments | 0 | 0 | 1 | 1 | 1 |
| 3 | Requeriments definition | 0 | 1 | 0 | 0 | 0 |
| 4 | Architecture of software design | 0 | 1 | 0 | 0 | 0 |
| 5 | User interface design | 1 | 0 | 0 | 0 | 0 |
| 6 | Model of data desing | 0 | 0 | 1 | 0 | 0 |
| 7 | Modelled with UML | 0 | 0 | 0 | 1 | 1 |
| 8 | Front-end programming | 1 | 0 | 0 | 0 | 0 |
| 9 | Back-end programming | 0 | 0 | 0 | 1 | 0 |
| 10 | Structured paradigm | 0 | 0 | 0 | 0 | 0 |
| 11 | Programming OO paradigm | 1 | 0 | 0 | 1 | 0 |
| 12 | Functional programming paradigm | 0 | 0 | 0 | 0 | 0 |
| 13 | Logical programming paradigm | 0 | 0 | 0 | 0 | 0 |
| 14 | Unit tests | 0 | 0 | 1 | 0 | 0 |
| 15 | Integration testing | 0 | 0 | 0 | 0 | 0 |
| 16 | Testing of acceptance | 0 | 1 | 0 | 0 | 0 |
| 17 | Software installation | 0 | 0 | 0 | 0 | 0 |
| 18 | Software configuration | 0 | 0 | 0 | 0 | 0 |
| 19 | Desktop software | 0 | 1 | 1 | 0 | 1 |
| 20 | Mobile software | 1 | 0 | 0 | 0 | 0 |
| 21 | Web system | 0 | 0 | 0 | 1 | 0 |
| 22 | Minimal experience | 0 | 0 | 0 | 0 | 0 |
| 23 | Regular experience | 0 | 0 | 1 | 1 | 0 |
| 24 | High experience | 1 | 1 | 0 | 0 | 1 |
| 25 | English language - low | 0 | 0 | 1 | 0 | 0 |
| 26 | English language - regular | 0 | 0 | 0 | 1 | 0 |
| 27 | English language - high | 1 | 1 | 0 | 0 | 1 |
| 28 | Individual task | 0 | 0 | 1 | 1 | 1 |
| 29 | Collective task | 1 | 0 | 0 | 0 | 0 |
| 30 | Task on office | 0 | 1 | 1 | 0 | 0 |
| 31 | Task on home office | 1 | 0 | 0 | 1 | 0 |
| 32 | Software documentation | 0 | 1 | 0 | 0 | 1 |

## 3.4 Matching of Tasks and Developers

The Method for Matching Tasks and Developers has been created as a tool for this problem. The application compares the pattern of a task with all the patterns of the developers.

The MMTD takes the profile of the task and search a developer profile with the greatest similarity and it assigns the task to the developer, achieving the most suitable assignment.

**Table 4.** Profiles of developers.

| | Profile of developers | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Requirements management | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 2 | Analysis of requirements | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 3 | Requeriments definition | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | Architecture of software design | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5 | User interface design | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 6 | Model of data design | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 7 | Modelled with UML | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 8 | Front-end programming | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | Back-end programming | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 10 | Structured paradigm | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 11 | Programming OO paradigm | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | Functional programming paradigm | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | Logical programming paradigm | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | Unit tests | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 15 | Integration testing | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 16 | Testing of acceptance | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 17 | Software installation | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | Software configuration | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 19 | Desktop software | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 20 | Mobile software | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 21 | Web system | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 22 | Minimal experience | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 23 | Regular experience | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 24 | High experience | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 25 | English language - low | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 26 | English language - regular | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 27 | English language - high | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 28 | Individual task | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 29 | Collective task | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 30 | Task on office | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 31 | Task on home office | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 32 | Software documentation | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

## 3.5 Matching Technics

For this work, five techniques have been used: SMC, Roger and Tanimoto, Sokal and Sneath, Hamman and Jaccard. The objective is to find the best technique with the best results. For to use these techniques, we apply the algorithm as in Algorithm 1.

**Table 5.** Patterns of profiles.

| New task | Profiles of developers |
|---|---|
| 01000100000001000010001010010100 | 00100110000000000001010101010010101 |
| | 00001001001000000011100100101010 |
| | 11000010001000000010101001010100 |
| | 01110010000000010110100000100101 |
| | 01000000000001110110000100110000 |
| | 01000110000000000001010100100100 0 |
| | 11000010000000000011100110010001 |
| | 00010000100000010000100101000010 |
| | 00000000000000000010000000001000 |
| | 01001000010000010000001000100000 |

**Algorithm 1.** Similarity coefficient.

```
Repeat from Task=1 to n
   Read pattern of PT
   // Extraction the values of a, b, c. and d.
   Repeat from Developer=1 to n
    Read pattern of PD
    // It count the values of a, b, d, and d
    If  PT[t][d] = 1 and PD[t][d]=1
      a=1
    If  PT[t][d] =1 and PD[t][d]=0
      b=1
      If  PT[t][d] = 0 end PD[t][d]=1
      c=1
      If  PT[t][d]=0 and PD[t][d]=0
      d=1
      a++, b++, c++, d++
    End of Repeat Developer
  // with the values of a, b, c, and d. It get the
similarity
  SMC = a + d / (a + b + c + d)
  RT = a + d / (a + 2b + 2c + d)
  SS = 2a + 2c / (2a + b + c + 2d)
  Ham= (a + d) - (b + c) / (a + b + c + d)
  Jac = a / a + b + c + d
End of Repeat Task
```

The algorithm gets the PT and it compares it with each PD to find the best assignment using matching. The project leader can find the ideal developer for each task of the project.

## 4 Results

With a sample of ten PD and five PT, the first task was analyzed. The results are similar; however, the technique with the highest similarity is Hamman, with 0.951. The best Developer for the Task 1 is the PD2. See Table 6.

**Table 6.** Results of Similarity coefficients of Task 1.

| | a | b | c | d | SMC | R&T | S&S | Ham | Jac |
|---|---|---|---|---|---|---|---|---|---|
| PD1 | 0 | 8 | 10 | 14 | 0,438 | 0,280 | 0,609 | −0,125 | 0,000 |
| PD2 | 8 | 0 | 3 | 21 | 0,906 | 0,829 | 0,951 | 0,813 | 0,727 |
| PD3 | 1 | 7 | 9 | 15 | 0,500 | 0,333 | 0,667 | 0,000 | 0,059 |
| PD4 | 1 | 7 | 10 | 14 | 0,469 | 0,306 | 0,638 | −0,063 | 0,056 |
| PD5 | 2 | 6 | 7 | 17 | 0,594 | 0,422 | 0,745 | 0,188 | 0,133 |
| PD6 | 1 | 7 | 7 | 17 | 0,563 | 0,391 | 0,720 | 0,125 | 0,067 |
| PD7 | 2 | 6 | 10 | 14 | 0,500 | 0,333 | 0,667 | 0,000 | 0,111 |
| PD8 | 2 | 6 | 5 | 19 | 0,656 | 0,488 | 0,792 | 0,313 | 0,154 |
| PD9 | 1 | 7 | 1 | 23 | 0,750 | 0,600 | 0,857 | 0,500 | 0,111 |
| PD10 | 2 | 6 | 4 | 20 | 0,688 | 0,524 | 0,815 | 0,375 | 0,167 |

In Fig. 1 Developer 2 can be seen as the most ideal. On the other hand, developer 1 is the lowest.

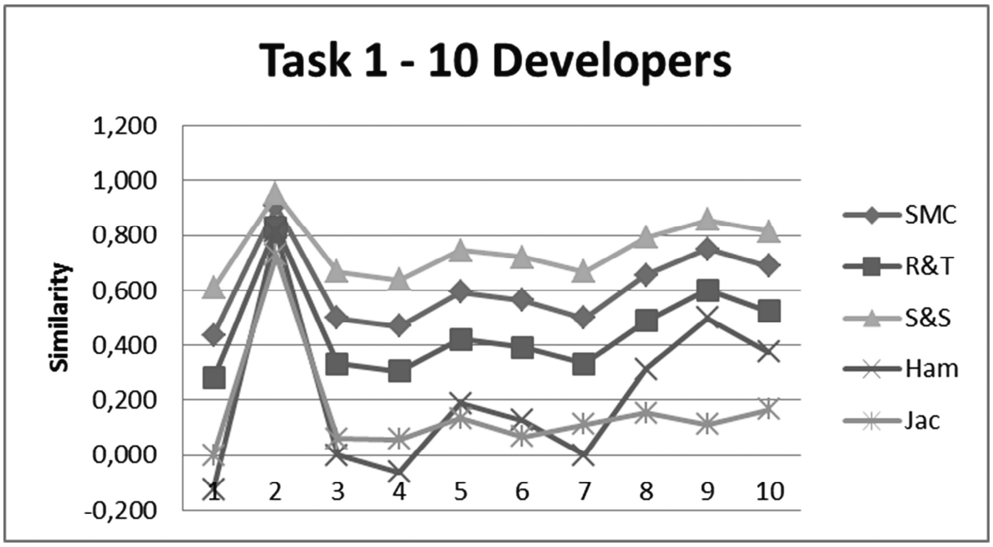

**Fig. 1.** Task 1 analysis.

For Task 2 analysis, Developer 4 is the most suitable. While, the Developer 1 and 6 are gets the lowest similarity. See Table 7. See also Fig. 2.

**Table 7.** Results of Similarity coefficients of Task 2.

| | a | b | c | d | SMC | R&T | S&S | Ham | Jac |
|---|---|---|---|---|---|---|---|---|---|
| PD1 | 4 | 4 | 6 | 17 | 0,656 | 0,512 | 0,808 | 0,344 | 0,286 |
| PD2 | 3 | 5 | 8 | 15 | 0,563 | 0,409 | 0,735 | 0,156 | 0,188 |
| PD3 | 2 | 6 | 7 | 16 | 0,563 | 0,409 | 0,735 | 0,156 | 0,133 |
| PD4 | 7 | 1 | 3 | 20 | 0,844 | 0,771 | 0,931 | 0,719 | 0,636 |
| PD5 | 4 | 4 | 4 | 19 | 0,719 | 0,590 | 0,852 | 0,469 | 0,333 |
| PD6 | 1 | 7 | 6 | 17 | 0,563 | 0,409 | 0,735 | 0,156 | 0,071 |
| PD7 | 4 | 4 | 7 | 16 | 0,625 | 0,476 | 0,784 | 0,281 | 0,267 |
| PD8 | 3 | 5 | 4 | 19 | 0,688 | 0,550 | 0,830 | 0,406 | 0,250 |
| PD9 | 1 | 7 | 1 | 22 | 0,719 | 0,590 | 0,852 | 0,469 | 0,111 |
| PD10 | 2 | 6 | 3 | 20 | 0,688 | 0,550 | 0,830 | 0,406 | 0,182 |

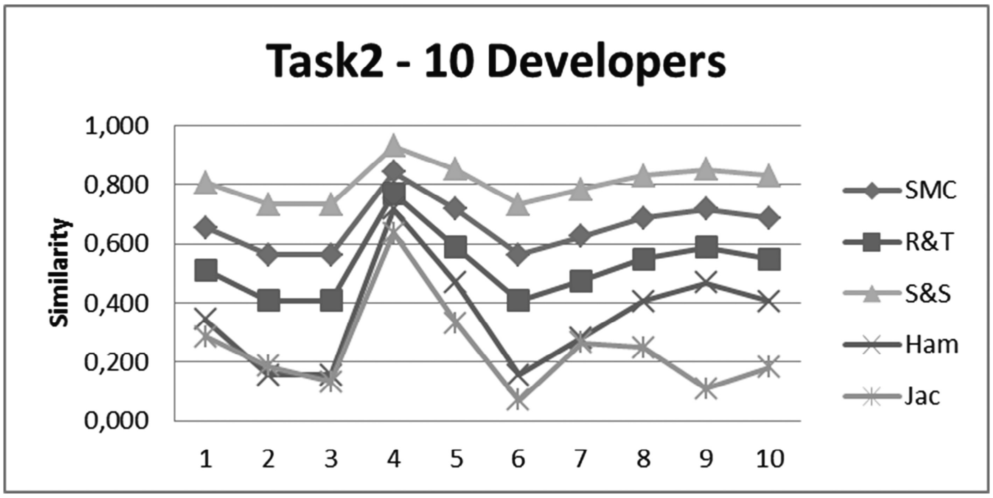

**Fig. 2.** Results of task 2.

Regarding the results of the five tasks and the ten developers the assigning it is as you see in Table 8. Specifying that, the corresponding tables and graphs the remaining tasks are not shown for reasons of space.

**Table 8.** Results of the five tasks

| Matching tasks and developers | | | | | |
|---|---|---|---|---|---|
| Task | 1 | 2 | 3 | 4 | 5 |
| Developer | 2 | 4 | 1 | 3 | 5 |

## 5 Conclusions

We conclude that MMTD can support project leaders on assignment of Tasks to Developers. Specially, in big development projects where hundreds or thousands of tasks take place. It can also support distributed projects. In this work, 32 characteristics have been included, but more characteristics could be added to the profiles to get a more precise matching. We recommended the technique Sokal & Sneath because sown similarity more high.

## References

1. Rodríguez-Salazar, M.E., Álvarez-Hernández, S., Bravo-Núñez, E.: Coefficients of similarity. ed. Plaza y Valdés S. A. de C. V (2001). ISBN 968-856-901-1
2. Legendre, L., Legrende, P.: Numerical Ecology. Elsevier, Amsterdam (1983)
3. Rogers, J.S., Tanimoto, T.T.: A computer program for classifying plats. Science **132**, 1115–1118 (1960)
4. Sokal, R.R., Sneath, P.H.: Principles of Numerical Taxonomy. W.H. Freeman and Company, San Francisco (1963)
5. Hamann U. Merkmalsbestand und Verwandtschaftsbeziehungen der farinosae. Ein Beitrag zum System der Monokotyledonen Willdenowia, pp. 639–768 (1961)
6. Jaccard, P.: Nouvelles recherches sur la distribution florale. In: Bulletin de la Sociète Vaudense des Sciences Naturelles, vol. 44, pp. 223–270 (1908)
7. García, J.F.: Métricas de Similitud para Búsqueda Aproximada. Revista de Tecnologia, Facultad de Ingenieria de Sistemas, Universidad del Bosque, pp. 1–11 (2007)
8. Rodríguez, G., Berdún, L., Soria, A., Amandi, A., Macelo, C.: Análisis de Métricas de Similitud en Razonamiento Basado en Casos para Administrar Proyectos, ASAI 2015, 16º Simposio Argentino de Inteligencia Artificial (2015)
9. Álvarez, C.M.A.: Thesis: Detección de similitud semántica en textos cortos, Instituto Nacional de Astrofísica, Óptica y Electrónica Tonantzintla, Puebla (2014)
10. Ruiz, C.J.S., Cervantes, J.C., Juárez, R.R.H., Trueba, A.E.: Métricas de Similitud SMC, Jaccard y Roger & Tanimoto en la Identificación de Plantas, CIINDET, Morelos, México (2016)