Recommendation System for Assisting the Management of Information Technology

Taciano Balardin de Oliveira  
Lutheran University of Brazil  
Cachoeira do Sul, Brasil  
Email: taciano@ulbra.edu.br

Felipe Becker Nunes  
Computer Education Post Graduate Program  
Porto Alegre, Brasil  
Email: nunesfb@gmail.com

Gleizer Bierhalz Voss  
Computer Education Post Graduate Program  
Porto Alegre, Brasil  
Email: gleizer.voss@ufrgs.br

Roseclea Duarte Medina  
Federal University of Santa Maria  
Santa Maria, Brasil  
Email: roseclea.medina@gmail.com

Jose Valdeni de Lima  
Computer Education Post Graduate Program  
Porto Alegre, Brasil  
Email: valdeni@inf.ufrgs.br

Abstract—Management of the problems occurred in environments that make use of the Information Technology (IT), together with the need for the quick response from the support teams area, a challenge for today. With this, organizations require systems to manage these incidents, as a Service Desk to centralize these records. The objective of this work is to integrate a system of Mobile Service Desk to a recommendation system that stores past interactions and automatically suggests as a possible solution for new similar incidents in the managed environment. As a contribution of this work, an algorithm was compared for similarity analysis and it has been integrated to the tool that showed the best results.

Keywords—Mobile Service Desk; Recommendation System; Similarity Analysis Algorithms

I. INTRODUCTION

The increasing dependence of organizations on the use of Information Technology (IT) is making the management of IT services within these environments an increasingly important activity [1]. In case of any problem in these managed local places (e.g., computer, printer, software, networks, or any device that causes abnormal functioning of IT services), the expectation is that the user has a quick response of the team support, so the damage can be minimized [2].

The concept of service to users of IT service, which was originally named Help Desk [3], was created; this way, the problems could be centralized and subsequently solved by technicians responsible for these tasks. However, today, this area has absorbed other services and proceeded to call up of Service Desk, in the case of an extended version of the Help Desk and offering a greater amount of services [4].

A challenge that reaches those responsible technicians for these management environments is that, in many organizations, there is a high turnover of human resources. In 2010, according to the Research Institute Gartner [5], the turnover of IT personnel around the world was only 3%. In 2011, it jumped to 5%. Thus, the departure of an employee is a loss of human capital and generates a replacement cost (i.e., recruitment, selection, hiring and training) which can be high. In addition, there is the difficulty of transferring knowledge and experience among employees of this area [6].

A possible alternative to the problem of turnover is the integration of a Service Desk tool to a recommendation system, where the technical solutions applied in previous situations are retained in the database system, and when a new calling with similar features occurs, they are presented as a possible solution to the incident.

This work is part of a project that aims to design and implement a Service Desk tool mobile, called Mobile Service Desk, which has features of context awareness (e.g., geographical position, expertise and time), and it equips this tool with a recommendation system. Thus, the intellectual capital generated through the services performed by the support staff which is retained in the system and it is recommended as a possible solution for new similar callings. This paper outlines the design and validation of the recommendation system integrated into the tool.

The paper is structured as follows: Section 2 presents the related work in the field of Service Desk. Section 3 explains the concepts of Service Desk, recommendation systems and pre-processing text. Section 4 discusses the proposal for the relationship between calls, which serves as a basis for recommendation system. Section 5 presents the methodology employed in the development of this work. Section 6 treats the implementation of the recommendation system integrated into the Service Desk. Section 7 presents the results obtained with the work. Finally, Section 8 contains final considerations of this paper and future work.

II. BIBLIOGRAPHICAL REVISION

This section is a literature review of the terms related to this research and also serves as a basis for the development of this work, such as Service Desk systems, pre-processing techniques and algorithms analysis of similarity, that provides support to the system that recommends similar calls.

A. Service Desk Systems

With the increasing business demand and globalization, more and more organizations need to ensure the quality of services performed to obtain better chances on the market. Thus, the goal of a Service Desk is to provide IT users with a Single Point of Contact (SPOC), vital to the realization of effective communication between users and teams that manage IT in an organization [7].
Its primary mission is to restore the normal operation of services of the users the fastest as possible, minimizing the business impact caused by IT failures [8]. In addition to it, the customer service keeps users informed about progress in solving incidents, changes and related events [4].

Operation of a Service Desk system occurs through the opening of calls or tickets. From this point on, whenever there is an open call, it is managed to be serviced. Moreover, these systems can also be based on some practical methodology for maintaining IT services, for example, the Information Technology Infrastructure Library (ITIL) [9].

B. Recommender Systems

A recommendation system combines computational techniques to select custom items based on users’ interests and as the context in which they live. According to Adomavicius and Tuzhilin [10], this issue has become an important area of research from the early work on collaborative filtering emerged in the 90s. According to the authors, the interest in this area remains high, because it has a large number of research problems and also for the abundance of applications that help users deal with information overload and provide recommendations, customized content and services to them.

Based on how the recommendations are made, these systems are generally classified as follows: (i) content-based, which seeks to recommend items similar to those that have been an interest to the user, (ii) collaborative filtering, which operates identifying users with similar preferences to present and recommend items that were of interest of that similar user, and (iii) hybrid-approach, resulting from the combination of collaborative and content-based methods [11].

The method based on content (i) calculates the utility of an item s for user c, based on the utility of “similar” items to the same user c. The calculation of similarity between items is performed through the use of a set of attributes that characterize each item [10]. To enable this type of recommendation it is necessary to find associations between these items [12].

The goal of collaborative filtering (ii) is to recommend new items or predict the utility of an item for a given user, based on the data from the similar users to it. Thus, the user will receive recommendations for items that people with similar preferences to it, preferred in the past. This method is divided into two categories: the first is called memory-based, and the second is called model-based. The calculation of the value of an item s in relation to a user c is made from the utility of the same item for other users c, similar to the user c [10].

The hybrid method (iii) is defined as a method that combines both strategies based on content recommendation, as for the collaboration-based strategies [11]. The advantage of an approach that unifies the others is to significantly increase the chances of getting correct answers on their recommendations and to eliminate the limitation of both approaches.

C. Pre-Processing Text

Text mining techniques, which can also be found in the literature as text data mining or knowledge discovery of textual data bases, in general, refer to the process of discovering knowledge in unstructured text documents. This technique can be seen as an extension of data mining or knowledge discovery in a structured databases [13].

In a Service Desk system, text mining techniques can be applied in the description of open calls by the user, in order to perform a pre-processing of texts to later analyze the similarity between them and identify similar cases of past interactions stored in database. This can be accomplished by using some text mining techniques, for example, by removing stopwords, and stemming algorithms for the similarity analysis.

A set of strings that compose a document consists of a few words (tokens) that have no semantic value, being useful only for the text that can be understood in general. In a system of data mining, such as words that are considered stopwords and belong to a stoplist. With a well organized stoplist is possible to eliminate up to 50% of the total words in a text [14].

An example can be seen through the phrase "I have problem in my printer.", by applying the technique of removing stopwords results in "Printer problem" resulting in around 66% of the words that compose the phrase being removed.

Stemming aims to reduce each word until its radical is obtained by removing suffixes that indicate variation in form of the word as plural, verb tense, adverbs, gender and accentuation. According to Krovetz [15], the use of stemming improved in 35% the recovery of information in some datasets.

Radicalization is a process that involves different algorithms according to the language in question, as there are differences in how words are formed, so that the application of a specific technique can produce mixed results according to the language of the texts [16].

According to Viera and Virgil [16], the approach that is best known for the Portuguese language is that of Orengo and Huyck [17]. There is also the algorithm of Porter [18], for the Portuguese, following the same rules developed by the same author for the English language.

1) Orengo Algorithm: The algorithm Orengo and Huyck is developed specifically for the Portuguese [17]. This algorithm consists of a series of eight steps, performed in accordance with a predefined order by the algorithm, such as the longest suffix that must be removed first. This algorithm was developed based on the most common suffixes found in Portuguese [19].

The Orengo algorithm presents eight steps that are as follows: (i) reduction in the plural, that removes the end s indicative of plural words that do not constitute exceptions to the rule, making modifications as necessary; (ii) reduction of the female, that removes the final a of female words based on the most common suffixes; (iii) reduction adverbial, that removes the final minded of words that do not constitute exception; (iv) reduction in grade, removes most common indicators of augmentative and diminutive; (v) nominal reduction, which are removed 61 suffixes for nouns and adjectives; (vi) verbal reduction, which reduces the number of verbal forms to their radicals; (vii) removing vowels, where it is removed the vowels a, e, o, of the words which were not addressed by the previous two steps; (viii) removal of accents, which are removed the diacritical signs of the words.

2) Porter Algorithm: Porter’s algorithm, originally proposed for English, is based on the idea that the suffixes in
English are mainly composed of a combination of smaller and simpler suffixes [18]. The algorithm consists of a series of five steps, in which some rules are applied to remove suffixes in each step. If a suffix combines with the word, the suffix is removed if the rules that were defined for that step are applicable [19].

In consonance with Viera and Virgil (2007), Porter’s algorithm adapted to the Portuguese language is also based on rules. Five steps are performed by him: removal of suffixes; removal of verb suffixes, if that first step is not carried out there isn’t any changes; removing the suffix (i) if it is preceded by (c); removal of residual suffixes os, a, i, o, á, í, ó; removal of suffixes (and), (is) and treatment of cedilla; after all, the nasalized vowels should be return to its original shape [16].

D. Similarity Algorithms between Strings

In literature, there are several techniques for calculating the similarity between strings, such as the inverted index model, Levenshtein Distance algorithm, natural language processing, algorithms of Boyer-Moore, Karp-Rabin, Jarro-Winkler, among other techniques that can be seen in [13][14][17][20]. In the next sections, some of these algorithms are presented.

III. RELATIONS BETWEEN TICKETS: PROPOSAL

One of the features of Service Desk proposed in this paper is that it presents solutions of past problems for the new incidents that the support will have to answer, in order to find a way to solve this new incident by applying a method already used in another similar call. Thus, the technical support team can have at your fingertips a possible solution to the problems that need to answer. For this to be done, when a user opens a call in Service Desk system, the system should automatically relate it with other previous records in the system that have already been solved.

In this work, five stages that compose the pre-processing text are applied, as proposed by Avila [21], which serve as an initial step for further analysis of similarity between strings. The first is the removal of invalid characters, such as quotation marks, brackets, parenthesis, among others, that need to be removed. After this, the replacement of accented character is performed, which is substituted by the respective non-accented character. The third step is the exclusion of repeated words, to avoid unnecessary comparison of a duplicate word several times. Also, a lowercase is applied, to prevent words with the same meaning that start with uppercase characters, will be differentiated from a similar word starting with lowercase characters.

The fourth step is the removal of stopwords, to prevent words like articles, adverbs, pronouns, prepositions, among others that have no semantic value, being only useful for text that can be understood, in general; Finally, stemming algorithm from Orenego and Huyck [17] is applied; this was chosen because it has been developed specially for the Portuguese language, to reduce morphological variants of the words, as singular forms, plural, verb conjugations, for its root or radical, by removing suffixes and prefixes.

After the execution of the steps for pre-processing in the opening text of the call, there will remain a set of strings that will be analyzed with previous case in order to determine the similarity between calls. For this, the similar_text() algorithm was used; the justification of using this algorithm is detailed in Section VI, which presents the comparative analysis of the algorithms for similarity between strings.

With the determination of similarity between calls, the system recommends possible solutions to a new incident. Thus, at the time the technician will accomplish its service, the system must have the knowledge of solved cases where the opening text of the call has similarity to the current incident. This paper proposes a method for content-based recommendation system, where the contents and characteristics of the calls are analyzed in order to determine what level of similarity they have and then recommend cases already solved to a similar that has not been answered.

Figure 1 shows in flow chart form the operation of the recommendation system of solutions proposed in this work.

```
Figure 1. Operation of Recommendation Solution System
```

Then, the techniques of preprocessing text, as addressed in Section 4, are applied. These techniques analyze the similarity of the resulting text with cases already solved in order to obtain a possible solution to the incident and submit it to the technician at the time that it is to serve you.

Figure 2 shows the operation flow of the proposed system. From the opening of a call by the user, the system automatically links this with previous calls; with it, it is possible to suggest the support team possible solutions related to the call opened. To connect the calls, the use of text preprocessing algorithms and similarity analysis have been proposed.

User opens a call in the system after the system searches if there are similar calls. If there is any information relating to that call, it will be displayed by the system. Otherwise, it transforms this new information in context to the system and writes the solution of the problem to be used in the future.

IV. METHODOLOGY

The recommendation concerns the use of mobile devices and systems research in order to propose a system for Service Desk were stimulated by the observation of the behavior of the User Service Center (USC) of a federal university. During the observation period (i.e., between October and December 2012), an informal interview each month, with some technical supervisors and the USC was performed.

Through interviews, some questions were raised, such as: (i) operation of the current system of USC as the opening of a call; (ii) the existence of some kind of prioritization and classification of calls; (iii) the existence of a division of the technicians as to their knowledge; (iv) how is the distribution
Four algorithms that calculate the similarity were compared: (i) Jaro-Winkler Algorithm [22], (ii) Levenshtein Distance Algorithm [23], (iii) String Similarity Algorithm, which is a class used to calculate the similarity between two text strings, created from the "diff" algorithm, that compares the difference between files under GNU (GNU Operating System) [24], and (iv) The similar_text() function, that is native from PHP for calculating the similarity [25].

For system modeling the Unified Modeling Language (UML) was used, which allows to represent application objects through a standardized graphical notation and diagrams. For creating UML diagrams, the software Astah Community [26] version 6.6 was used, which is a free UML modeling tool.

The database was modeled with the support of DBDesigner software, in version 4, which is a free program that integrates creation and graphical modeling of data [27]. The database used for data persistence was MySQL [28], and the tool used to access and maintain the database was HeidiSQL [29], which is also free.

The module recommendation of the Mobile Service Desk solution has been implemented in NetBeans, a free software, which is an Integrated Development Environment (IDE), and can be used with different programming languages, such as Java, PHP, HTML, JavaScript, C/C++, etc. [30].

To validate the proposal, a free programming language that has features such as ease of handling strings and Web programming was sought. Beyond the initial requirements, the PHP Hypertext Preprocessor (PHP) was also chosen for having the following characteristics: (i) server-side language, which performs multi-functions, (ii) compatible with MySQL, (iii) cross-platform, (iv) functions available for use, and (v) documentation.

To work on many mobile devices, the system has a unified user interface, i.e., an unique implementation for all mobile devices and operating systems. Thus, to provide this unified interface, the jQuery Mobile framework was used, which is based on HyperText Markup Language 5 (HTML 5), and has libraries as the jQuery and jQuery UI, and has as characteristic to be optimized for touch interactions [31].

Finally, the Mobile Service Desk was installed on a server with the operating system Ubuntu, version 12.04, Long Term Support (LTS), using 32-bit architecture. In this, MySQL 5.5 database and Apache 2.2 web server, with support for PHP 5.3 programming language, were installed and configured.

V. MOBILE SERVICE DESK INTEGRATED AT THE RECOMMENDATION SYSTEM

Faced with the high use of mobile devices, such as smartphones and tablet, it was decided to study ways to use these technologies to apply in IT management. The idea of using the location information of the technician is to speed up the service of the tickets which is already expressed in the work of Lobo [2], who also considered that not only the location was important, but also the definition of a mechanism that utilizes the experience and practice of technicians to decrease the occurrence of a second service.

In this work, these practices were maintained and improved in order to consider not only the latitude and longitude of the technical but also their altitude at the time of setting a priority of service. Since the cases where the building is composed of several floors may occur. Thus, the question of the altitude is an essential factor to set the distance from one point to another.

Furthermore, other problems found in these IT environments, such as the issue of staff turnover, led to the study of the techniques for storing the solutions and later on the treatment of these data, so for those new team members can get suggestions for possible solutions to similar problems that have already been resolved. In order to have suggestions of similar solutions could be aggregated to the tool, the study of techniques in the area of recommender systems was necessary.

The Mobile Service desk is a system that has specific functional features, such as run on a variety of mobile devices, treat context sensitivity, and keep a history of the solutions of the problems so that later this solution may be suggested to some other similar ticket through of a system of recommendation.

To perform the service of tickets, from the moment when a “Support” level user accesses the system, the open tickets are listed. At the moment in which informs that will start attending to the incident, the system will list the solutions of similar cases to the current problem, so that a solution can be reused. Even at the time the technician will attend some ticket, theses cases are related to the current treatment are presented. For this, the relationship of tickets was implanted, as addressed in Sections V-A and V-B.

The implementation of the relationship of tickets, to be further suggested as a possible solution of a problem to be treated, is divided into two stages: (i) pre-processing text; (ii) similarity analysis.
A. Pre-Processing Text

In this step, to each opening text of ticket, the following actions are applied: (i) removal of invalid characters, (ii) application of lowercase, (iii) removing stop words, (iv) exclusion of repeated words, and (v) application of stemming algorithm.

In the second line of Figure 3, the variable call "string" receives off a POST method, the value of the text to be processed; in "limpaTexto()" function the text is passed to lower case and accents and other punctuation characters, dashes, quotation marks, brackets, among others are removed. The fourth line is responsible for mounting an array with all the words that compose the text.

The “removeStopwords()” function in line 5 is responsible for the removal of the stopwords of the text, in turn, the “array_unique()” function has the task of removing the terms duplicated. In the line 7, the “arquivoStemming()” function generates the text file with the key words of the text, so that in line 8 to run the Python script, which applies the technique of stemming in Portuguese on the file previously generated. This, the result at 9, is stored in the "preprocessado" (pre-processed in English) variable.

Other information to be highlighted about the pre-processing text step is regarding to the stemming; Pstemmer script [32] was used because it possesses both the algorithms presented in [17] and [18], implemented for the Portuguese language. In this work, we opted to use the Orengo and Huyck [17] algorithm, because its rules were specifically created for the Portuguese language; in turn, the Porter algorithm is an adaptation of another language to Portuguese. In case the language should be applied to the system.

B. Similarity Analysis

To verify the similarity between the text of the ticket to be attended to and the texts stored in the system relating to, the comparing strings and returns the percentage of similarity between them was applied. This algorithm was chosen due to its superior performance in relation to the other algorithms tested. The complete comparison between the similarity algorithms is presented in the Section VI.

VI. RESULTS

In this section, the validation of the suggestion system integrated at the Mobile Service Desk with the test plan and results obtained through the evaluation of the analysis of similarity algorithms is described.

The tests of the similarity analysis algorithms were performed using real data captured from the database of User Service Center (USC) containing the tickets opened between January and November 2012, in a total of 7033 tickets. To test the running time of the algorithms as well as the quality of the results were opened several tickets in the system; however, some of them will not be shown in this paper; the others can be seen in Table I.

<table>
<thead>
<tr>
<th>Nº</th>
<th>Texto do Chamado</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Meu computador está extremamente lento. Desconfio que seja algum vírus</td>
</tr>
<tr>
<td>2</td>
<td>Ocorre erro de spooler na impressão!!!</td>
</tr>
</tbody>
</table>

After pre-processing of texts opening tickets was obtained the results presented in Table II. This resulting text was used to compare the similarity between them and the USC tickets imported to the system.

<table>
<thead>
<tr>
<th>Nº</th>
<th>Preprocessed Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>comput extrem lent descontl viru</td>
</tr>
<tr>
<td>2</td>
<td>ocor err spool impressa</td>
</tr>
</tbody>
</table>

The results referring to the five cases with highest similarity obtained when running the test related to the first open ticket in the system “Meu computador está extremamente lento. Desconfio que seja algum vírus” (“My computer is extremely slow. There must be a virus” in English) are shown in Tables III, IV, V and VI.

In these tables, the first column shows the opening text for the ticket similar to how it was in the database of the USC (i.e., without spelling corrections or abbreviations), the second column is the percentage of similarity between these records and the case that has been verified.

Analyzing the data from the first test of similarity algorithms, it is possible to conclude that the five records with greater similarity captured by Jaro-Winkler algorithm are somewhat related to the problem of "computador lento" ("slow computer" in English). In the Levenshtein distance algorithm, two of the results are unrelated to the problem, since one it is just a slowness in browser and another is without network access. In the String Similarity, the result “Computador e retro-projetor desconfigurado” ("Deconfigured Computer and multimedia projector" in English) is unrelated to the problem. Finally, in similar_text() the top five results are related to the problem.

This way, it is possible to conclude that, in the first test, the Levenshtein distance and String Similarity algorithms do not return a good result, by seeing how these similar cases that were unrelated to the problem.

The second test was carried out by way of the so-called "Ocorre erro de spooler na impressão!!!", ("There are error of spooler at the printing!!!" in English); after the execution of the similarity algorithms, the results achieved are shown in Tables VII, VIII, IX and X.

Analyzing the data from the second test of the similarity algorithms, it is possible to conclude that the Jaro-Winkler algorithm does not obtain good results, given that the first
four records more similar that he found have no relation with the problem. Levenshtein distance, String Similarity and similar_text() algorithms, all the results has to do with printing problems, moreover bring a result related to the spooler problem.

To measure the execution time of each algorithm, the timestamp server was detected at the start of processing and, at the end, this value it was subtracted from the current
TABLE VIII. TEST OF TICKET 2 FOR LEVENSHTEIN DISTANCE ALGORITHM

<table>
<thead>
<tr>
<th>Text of Ticket</th>
<th>Preprocessed Text</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impressora configurada e não responde às solicitações de impressão.</td>
<td>impress configur respond solicitaçao impressa</td>
<td>75.47 %</td>
</tr>
<tr>
<td>Computador formatado ontem precisa ser colocado em rede.</td>
<td>comput format ont precis coloc red instal impress</td>
<td>74.06 %</td>
</tr>
<tr>
<td>Instalar impressora.</td>
<td>instal impress</td>
<td>73.33 %</td>
</tr>
<tr>
<td>Reinstalar e configurar na rede serplo uma impressora matricial Epson fx 2180</td>
<td>reinstal config red serp impress matric epson fx ob impressa relato continu</td>
<td>72 %</td>
</tr>
<tr>
<td>Obs. para impressão de relatórios continuos.</td>
<td>instal serv spool impress desativ pec retir sal instal comput coloc red impress</td>
<td>70.99 %</td>
</tr>
<tr>
<td>Instalar serviço de spooler, impressoras desativadas</td>
<td>instal serv spool impress desativ</td>
<td></td>
</tr>
<tr>
<td>Peço para retirar de uma sala e instalar em outra um computador e colocar em</td>
<td>dem p impress</td>
<td></td>
</tr>
<tr>
<td>rede o mesmo com a impressora.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE IX. TEST OF TICKET 2 FOR ALGORITHM STRING SIMILARITY

<table>
<thead>
<tr>
<th>Text of Ticket</th>
<th>Preprocessed Text</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instalar serviço de spooler, impressoras desativadas</td>
<td>instal serv spool impress desativ</td>
<td>57.14 %</td>
</tr>
<tr>
<td>Não consigo imprimir, erro na impressão.</td>
<td>consig imprim err impressa</td>
<td>57.14 %</td>
</tr>
<tr>
<td>Solucionar problema com impressora</td>
<td>soluc problem impress</td>
<td>54.90 %</td>
</tr>
<tr>
<td>Solucionar problema com impressora</td>
<td>soluc problem impress</td>
<td>54.90 %</td>
</tr>
<tr>
<td>Demora p/ impressão.</td>
<td>dem p impress</td>
<td>54.54 %</td>
</tr>
</tbody>
</table>

TABLE X. TEST OF TICKET 2 FOR ALGORITHM SIMILAR_TEXT()

<table>
<thead>
<tr>
<th>Text of Ticket</th>
<th>Preprocessed Text</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Não consigo imprimir, erro na impressão.</td>
<td>consig imprim err impressa</td>
<td>62.74 %</td>
</tr>
<tr>
<td>Instalar serviço de spooler, impressoras desativadas</td>
<td>instal serv spool impress desativ</td>
<td>62.06 %</td>
</tr>
<tr>
<td>problemas de impressão.</td>
<td>problemas impressa</td>
<td>60.46 %</td>
</tr>
<tr>
<td>Não dá a ordem para impressão</td>
<td>ord impressa</td>
<td>59.45 %</td>
</tr>
<tr>
<td>Reinstalar impressora.</td>
<td>reinstal impress</td>
<td>58.53 %</td>
</tr>
</tbody>
</table>

Comparing each of the opening texts of tickets with each record in the database system, the similarity analysis algorithms had very different performances. The lowest average time to perform all the calculations, as the graph shown in Figure 4, it was obtained by the similar_text() function algorithm with an average of 0.16 seconds. The Levenshtein Distance algorithm shown the worst performance, taking an average of 125.85 seconds to accomplish the task. In turn, Jaro-Winkler (8.45 seconds) and String Similarity (26.59 seconds) algorithms had intermediate performance.

Figure 4. Performance Analysis of Algorithms Similarity (in seconds)

VII. Conclusion and Future Work

The function of a Service Desk is to perform the process of incident management, dealing with all incidents of an organization linked to the IT area, such as software and hardware failures, communication networks or any device that causes the abnormal functioning of the IT services. The main objective of incident management is to restore the operation of the service as quickly as possible.

In addition, the Service Desk provides for IT users an IT SPOC, vital for an effective communication between users and teams that manage IT in an organization [7]. This helps managers, since it is not necessary to visualize numerous tools to access information concerning incidents of IT environment.

The results obtained by this research indicate that the similarity analysis algorithms and pre-processing text can be part of an integrated Service Desk to recommendation system solutions which, as seen in the Section VI, in the majority of cases return results referring to the problem in question. With these similar problems, it is possible that the solution adopted in the previous one should be reused by a technician in the new problem that needs to be solved; this way, by means of indications of the possible solutions, the system of recommendation may speed up to solve the incident.

Thus, the main contributions of this study are: (i) modeling of a recommendation system aggregate to the Service Desk tool; (ii) validation of the proposed solution by testing using several similarity analysis algorithms and using the algorithm with the best performance in the recommendation system.

In the light of the results, it can be considered that the use of recommendation for possible solutions might help the
technicians of the teams of assistance, especially for those that joined the team recently. It is further considered that these improvements also help to reduce the determining and resolution cost of incidents, which for Song [33] represent more than half of the operating IT costs.

REFERENCES


