Neural Computing and Applications
https://doi.org/10.1007/s00521-020-05522-1

# Recommendation of technological profiles to collaborate in software projects using document embeddings

Pablo Chamoso[1] · Guillermo Hernández[2] · Alfonso González-Briones[3] · Francisco J. García-Peñalvo[4]

Received: 17 July 2020 / Accepted: 10 November 2020


**Abstract**
The information technology sector is continuously growing, and there is a high demand for developers. In the area of software development projects, fixing bugs or solving issues is a task that could be optimized to improve the productivity of developers. Making an adequate allocation for bug fixing will save overall project development time. Moreover, the problem will last for the shortest possible time, minimizing any negative impacts in case the project is already in production. This research work's objective is to identify the most apt users (where the term "user" refers to any technology professional, for example a software developer, who has registered on any given platform), from a set of different user profiles, for fixing bugs in a software project. The study has been carried out by analyzing large-scale repositories of open-source projects with a large historical volume of bugs, and the extracted knowledge has been successfully applied to new, unrelated projects. Different similarity-based profile raking procedures have been studied, including neural-network-based incidence representation. The obtained results show that the system can be directly applied to different environments and that the selected user profiles are very close to those selected by human experts, which demonstrates the correct functioning of the proposed system.

**Keywords** Text analysis · Artificial Neural Networks · Large-scale repositories · Candidate selection · Software bugs · Solving software issues



## 1 Introduction

The most recent advances in computer science have made it possible to automate a large number of tasks in different job sectors. The increase in the processing capacity of computers and the techniques that allow for the execution of tasks on different machines (distributed computing or resource virtualization) have, over the last few years, increased the capacities of methodologies based on Artificial Intelligence (AI).

One of the job sectors where AI-based methodologies have much to contribute is precisely the technology sector. This is a sector with a high employability rate; the high demand for qualified personnel can be reduced by improving productivity.

Thus, the focus of the present research work is improving productivity. At present, one of the main activities of the technology sector is the production of software [1]. However, during this process a lot of time is

✉ Francisco J. García-Peñalvo
fgarcia@usal.es

Pablo Chamoso
chamoso@usal.es

Guillermo Hernández
guillehg@air-institute.org

Alfonso González-Briones
alfonsogb@ucm.es

[1] BISITE Research Group, University of Salamanca, Calle Espejo, 24.2, Salamanca, Spain

[2] AIR Institute, Paseo de Belén 11, Campus Miguel Delibes, 47011 Valladolid, Spain

[3] GRASIA Research Group, Complutense University of Madrid, 28040 Madrid, Spain

[4] GRIAL Research Group. Department of Computer Science and Automation, Faculty of Science, University of Salamanca, 37008 Salamanca, Spain

often lost on detecting and fixing bugs/issues; a task that is crucial to the success of the projects [2, 3].

There are different tools such as Jira [4] or GitHub [5] which have been designed to facilitate this task. They allow developers to keep track of all the issues that arise in a software project. In large projects, where teams with numerous developers are involved, locating and keeping track of all the issues is a considerable time-saver.

Normally, there are two teams, one in charge of developing and another in charge of fixing the issues, so maintaining a bug-free system is not an easy task due to multiple factors, including the very decision of who should be in charge of solving an issue.

Hence, the aim of this project is to help reduce the time involved in keeping software projects free of bugs or issues by automatically identifying the profile that is the most apt for fixing a software bug. These candidates can be members of the development team or independent developers looking to work on new challenges or collaborate with the open-source community.

In this research we have employed large-scale repositories of open-source projects, where there is a historical record of all the bugs/issues that have been reported. The proposal is focused on applying AI mechanisms to extract all the knowledge that these repositories encapsulate and thus be able to use it with smaller software projects. This knowledge will be used to extract the user profiles that are the best to participate in each project and thus that are capable of rapidly fixing the bugs or issues that appear in the project.

Thus, the main contribution of the article lies in the proposed system; it offers a new functionality for the automatic selection of the profiles that are most apt for fixing a specific bug/issue. Moreover, the system offers the ability to extract specific knowledge (associated with bugs/issues) from large-scale repositories of open-source projects. Good performance has been achieved in knowledge extraction, especially in terms of direct applicability to smaller projects, even if they are not directly related at the technological level. This is due to the capacity of abstraction and knowledge extraction of the proposed system.

This paper presents a case study where: i) a neural model based on document embeddings has been obtained using a collection of large-scale repositories as input (Eclipse, Mozilla and OpenOffice projects) for the purpose of encoding software issues; ii) information from users and public GitHub projects (which have employed well-known technologies; however, they vary from the technologies in the projects that have been used to train the model) is retrieved; and iii) a system that recommends which GitHub user is the best to solve the issues retrieved in ii) is built and is evaluated against human recommendations.

The results indicate that the system is capable of automatically selecting from unlabeled text the user profiles that are most apt for resolving specific issues in much the same way as human experts who have participated in the experiment would.

Besides, the system is easily adaptable and portable to all types of projects and technical profiles, so applying the solution on employability portals or portals used as repositories of open-source projects could provide great results in the selection of candidates, improving the quality and agility of employability processes.

The rest of the article is structured in such a way that Sect. 2 presents a review of background research, as well as the trends of the technologies applied to the technical proposal. Later, in Sect. 3, the solution designed to achieve the research objective is presented. Once the proposal has been explained, Sect. 4 presents the results obtained. Finally, in Sect. 5, the obtained conclusions and the future lines of research are presented.

## 2 Background

This section presents the background of this research work, starting by presenting the works related to the analysis of large-scale repositories that have been presented within the framework of software development. This is followed by a study of the AI techniques commonly used in related work, with a special focus on Artificial Neural Networks (ANNs), the main methodology selected to carry out this study, and finally, the section will end with the conclusions drawn from the analysis of the literature.

### 2.1 Related research works

Although there is a large amount of research presenting analyses of software projects from large-scale repositories, none of them have studied mechanisms for the detection of the best software developer profiles for incident solution. Thus, software code repositories have been used to carry out a review of related studies; their objectives and the techniques they had employed.

In this regard, the scientific community is using large-scale repositories with information on software development in well-known projects such as Eclipse, Mozilla [6], Jira [7], OpenOffice or MySQL [8]. The importance of bug reports is very high in open-source projects, especially if they are large projects. For this reason, different authors have been working on unifying the best-known projects and working with all of them at the same time. As a result, repositories have been created for carrying out projects that combine the use of the Eclipse, Mozilla and OpenOffice

repositories, such as those presented in [9] and [10], the latter being the most recent compilation of issues.

Among recent research, an approach that stands out has been presented in [11], where AI methodologies have predicted the appearance of bugs (an algorithm based mainly on two classifiers) from a set of incidents reported over the last 14 years in the Jira bug repository.

In [12], a methodology based on N-gram IDF (Inverse Document Frequency) has been presented for the proper classification of the received bug reports. In addition, recent studies have focused on detecting whether or not a bug has already been reported to avoid duplication and loss of time on fixing it [13]. Others focused on the severity of the reported bugs, as is the case of [14], where a study has been carried out on the results of the best-known machine learning techniques (Naive Bayes, Multinomial Naive Bayes, Support Vector Machine, Decision Tree, Random Forest, Logistic Model Trees, Decision Rules and K-Nearest Neighbors). Deep Neural Networks (DNNs) have been used in [15] to calculate the priority of the bug. Convolutional Neural Networks (CNNs) have also been used to address the problem of duplicate bug detection [16].

The most recent approaches try to improve the results obtained in previous studies through the use of complementary information, as in the case of [17], which incorporates the analysis of comments, or as in [18], where the developers' acquaintance with bug fixing is taken into account, evidencing the importance of properly selecting the developer for optimal solution of incidences.

However, bugs are just as important as the people who are able to fix them optimally. For the present work, the optimal person to solve an incidence will be considered to be the one whose characteristics are best adapted to the extracted characteristics associated with the corresponding project. By selecting the people that are most apt foe the task, the time it takes to fix the bug is considerably reduced, as well as the negative impact it may have on the project's end users. In addition, the possibility of new, associated problems emerging is minimized and there is greater confidence in that the bug has been definitively fixed.

In addition to the already mentioned approach presented by the authors of [18], other approaches analyze the profile of the users, such as the one presented in [19], which makes a pre-selection of the most suitable candidates for job offers from the information available on a social network. The research presented in [20] extracts all the information related to a person on the Internet and which can be very useful to have additional information about the candidates for a job position, beyond what they have described in their profiles.

## 2.2 Related tendencies in artificial intelligence

In the area of AI, a large number of researches have employed AI methodologies to solve problems similar to those addressed by the current research work.

One of the main trends in AI research is textual analysis, since in the vast majority of cases the information is not properly synthesized and labeled and therefore must be extracted from written texts in a variety of ways (as they are large-scale repositories). Once the required knowledge has been extracted from the input texts, it is necessary to apply methodologies that allow to choose the most apt users to fix specific bugs in the most optimal way. This requires a proper analysis of the user profile and the software project to determine the similarity between the type of problem and the profile of the user/software developer. This section presents a summary of the methodologies that have been applied to solve similar problems in existing works.

Firstly, an analysis has been carried out of the methodologies used in different researches to extract key knowledge from unlabeled text, for example software incidences, or the user profiles of candidates for a job.

As indicated previously, ANNs have been used for different purposes in the area of bug analysis (DNNs and CNNs have been described as examples). However, in the area of text analysis they are being widely used for many more purposes. For example, Deep Convolution Neural Networks (DCNNs) have been employed in sentiment analysis in [21], Recurrent Networks (RNNs) have been used in [22] for multilingual text analysis, C-LSTM (CNN Long Short-Term Memory) Neural Networks for automatic article tagging in [23] and text classification in [24].

However, the main ANN-based technique being used in textual analysis is Word2vec (W2V), which was created, published and patented by Google [25]. W2V encapsulates different models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. The result of applying W2V to a large corpus of text (in the case of this work, the whole set of events obtained from a large-scale repository) is a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located close to one another in the space [26].

The same idea can be extended to sentences (Sentence2Vec [27]) and complete documents (Doc2vec [28]) where instead of learning feature representations for words, the system learns it for sentences or documents.

Since its publication, W2V has been widely used and applied to analyze text with multiple functional approaches. For example, in [29] it has been used to classify articles and tweets, sentiment analysis has been carried out in [30] or topic extraction and user classification in [31]. The latter has provided satisfactory results. Although it has been applied in a completely different context, at a functional level its objective in the application of W2V is similar to that of the present research.

### 2.3 Conclusion

The conclusion drawn from the review of background literature is that there is a large amount of researches on the analysis of incidence repositories of open-source projects, but none of them have studied the factors that would have been relevant for this research, such as the extraction of knowledge from large repositories and its application in smaller projects, or the selection of the profile of the most suitable developer to address a given set of issues.

In this regard, reviewing state-of-the-art literature has been helpful in selecting the dataset that has been used for the training of the system, namely, the dataset proposed in [10]. This knowledge is intended to be applied in smaller open-source projects published on GitHub. The selected projects will be related to a variety of technologies.

At a technical level, some AI techniques enable the study of their results and the section of the one that renders the best results. However, these methodologies are not applicable 'as is' and specific adaptations must be made to this case study.

Therefore, it is necessary to propose a novel methodology with a scheme adapted to the needs of the proposed functionality, although existing text analysis techniques can be applied since, according to the literature, they provide excellent results.

## 3 Proposed system

The previous section has justified the main objective of this research and reviewed current trends in the analysis of open-source repositories of projects and in the use of AI methodologies such as ANNs. The technical details of the proposed solution are described in this section. However, it is necessary to first describe the methodology that has been followed.

Below, the methodology and subsequently the technical solution are described in detail.

### 3.1 Methodology

When analyzing large-scale repositories of open-source projects, it has been decided to start with the set proposed by [10] which, as already described in Sect. 2.1, presents a fairly complete and recent collection of issues from the Eclipse, Mozilla and OpenOffice projects (this dataset has been described in [10]). This dataset is to be used to create a system capable of extracting knowledge from software issues so that it can be reused for any other project.

To demonstrate the generalization capabilities of the resulting system, open-source projects published on GitHub will be used (which also has its mechanism for solving bugs/issues). Likewise, GitHub will be used to analyze the characteristics of different profiles and determine the best users to solve those issues. The selected projects are shown in Table 1.

A group of 10 evaluators (all of them holding a university degree in computer science, with at least two years of work experience) participated in the evaluation of the results. Each evaluator was given a double-entry table, matching the GitHub users that they follow (therefore, it is very likely that the evaluator knows the degree of suitability of each user to collaborate in the indicated projects) with the fixed set of popular repositories (Table 1). The evaluators were asked to provide a score of the potential adequacy of each repository–user pair using a 5-point Likert scale [32] with the following meaning:

1. Strongly disagree
2. Disagree
3. Neither agree nor disagree
4. Agree

**Table 1** Selected GitHub projects

| GitHub project | Description |
|---|---|
| numpy/numpy | Scientific calculations |
| tensorflow/tensorflow | Machine learning |
| mongodb/mongo | Databases |
| matplotlib/matplotlib | Visualization |
| pallets/flask | Python webserver |
| symfony/symfony | PHP framework |
| vuejs/vue | JS Framework |
| kubernetes/kubernetes | Container manager in Go |
| angular/angular.js | HTML enhanced for web apps |
| nodejs/node | JavaScript runtime |
| dotnet/aspnetcore | Cross-platform .NET framework |
| mozilla/geckodriver | WebDriver for Firefox |
| eclipse/eclipse | eclipse.platform project website |
| eclipse/mosquitto | An open-source MQTT broker |

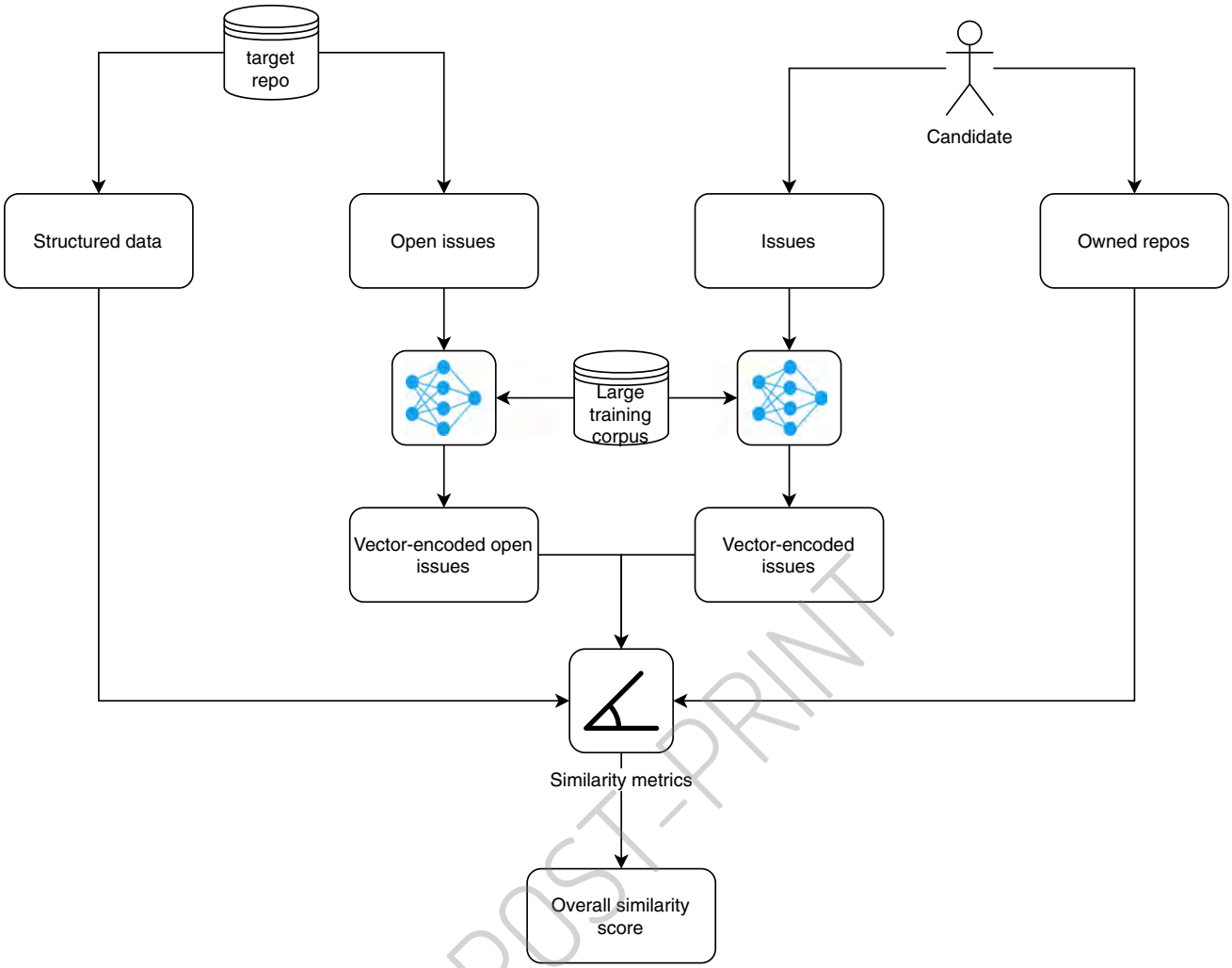


**Fig. 1** Workflow of user–repository similarity calculation

5. Strongly agree

The evaluators have also been asked to skip the evaluations they were unsure of, to avoid the central tendency bias.

Despite certain subjectivity inherent to the human component, these evaluations will serve as a reference when evaluating the accuracy of the recommendations made by the system.

The system's recommendations will consist in indicating the most suitable GitHub user (from the set of users analyzed by the evaluators) to solve the bugs/issues that arise in each of the projects selected to test the system.

Once a summary of the methodology has been presented, the following subsection proceeds to detail the technical approach of the proposed solution.

## 3.2 Technical approach

The process of user–repository similarity measurement is depicted in Fig. 1. A repository consists of both structured information, such as programming languages or its topics described as keywords, as well as textual information, such as the bugs or the descriptions. This information is recovered using a scrapper based on the GitHub API[1]. The same process is carried out for the users, whose information is extracted using the same approach.

Using the whole set of descriptions extracted from the incidences of the repositories presented in Sect. 3.1 [10], a Doc2vec model [33, 34] was trained to produce 100-dimensional vectors considering a 10-word distance, using a tokenized, lower-case version of the documents as the training set. The same preprocessing has been applied to the input whenever this vectorizer is used.

Initially, evidence for the usefulness of the constructed embedding in the representation of the textual content of the bugs can be found by making a reduced-dimensionality plot of the vectors associated with those bugs. A *t*-

[1] GitHub API - https://developer.github.com/v3/ (Accessed July 14, 2020).

distributed Stochastic Neighbor Embedding ($t$-SNE) [35] of the incidences of the repositories in Table 1 is shown in Fig. 2.

The similarity of the structured information can be measured using well-known metrics. Since a set of attributes is being considered, a neutral choice is the Jaccard similarity coefficient

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \tag{1}$$

where $A$, $B$ are the sets of tags being compared.

In the case of textual information, the Doc2vec model presented in Sect. 3.1 is used to obtain a vectorial representation. The overall bug similarity can then be extracted taking the cosine similarity of the aggregated vectors, defined by

$$s(A,B) = \frac{\sum_i \left(\sum_j A_i^j\right)\left(\sum_k B_i^k\right)}{\sqrt{\sum_i \left(\sum_j A_i^j\right)^2}\sqrt{\sum_i \left(\sum_k B_i^k\right)^2}}, \tag{2}$$

where $A_i^j$ ($B_i^j$) is the $i$-th component of the vector representing the $j$-th ($k$-th) incidence of the repository $A$ ($B$).

Once these user–repository similarities are calculated, they can be used to define a raking matching each of the users to an ordered list of repositories that the system has estimated will be most relevant to them.

Finally, the Kendall rank correlation coefficient is used to provide a quantitative measure of the agreement of the rank made by the human evaluators and the rank obtained from the designed algorithm. The so-called $\tau$-b version, which accounts for ties, is used as defined in [36], i.e.,

$$\tau = \frac{P - Q}{\sqrt{(P + Q + T_1) \cdot (P + Q + T_2)}}, \tag{3}$$

where $P$ is the number of concordant pairs, $Q$ is the number of discordant pairs and $T_1$ and $T_2$ are the numbers of ties in each of the compared rankings but not in the other one.

## 4 Results

A total of 38 distinct profiles were evaluated, collecting a total amount of 522 repository–profile evaluations. (Empty valuations have not been taken into account.) The distribution of the scores provided by the evaluators is shown in Fig. 3. As the figure shows, the data follow a typical flat bell-shaped curve.

All of these profiles were evaluated concerning each of the repositories in Table 1 using the available information and then compared to the human evaluations using the Kendall rank correlation coefficient over the subset of repositories for which a non-null evaluation was provided. The results are shown in Fig. 4 using a box plot that summarizes the distribution of such a coefficient for each

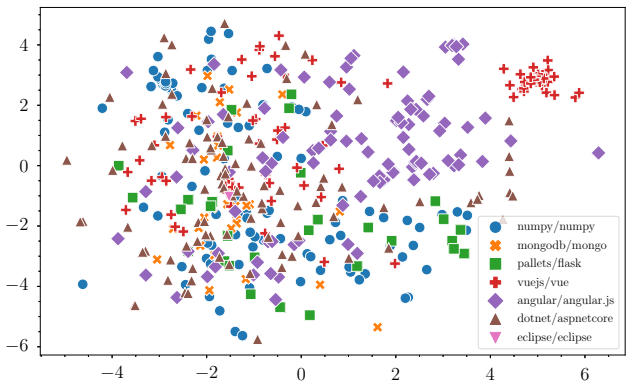


**Fig. 2** Scatter plot of a $t$-distributed Stochastic Neighbor Embedding of some of the open issues for repositories in Table 1. The number of repositories has been reduced on the plot for the sake of clarity

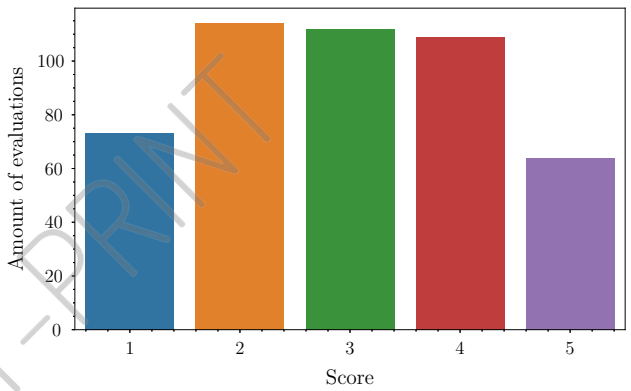


**Fig. 3** Distribution of the scores provided by the evaluators to the repository–profile pairs

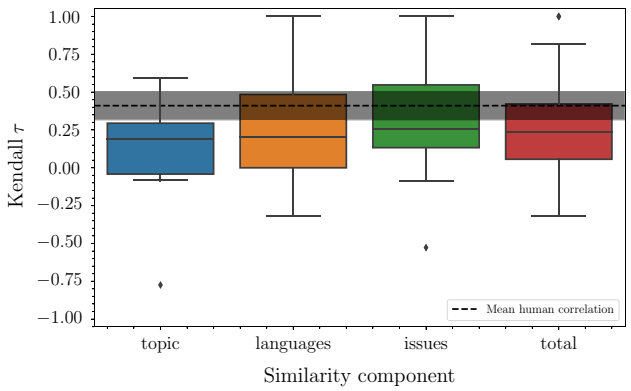


**Fig. 4** Evaluation results in terms of the distribution of the Kendall rank correlation coefficient. Each of the components (x-axis) is shown as a box plot where the whiskers delimit 1.5 times the inter-quartile range. For the sake of comparison, the mean human-to-human correlation coefficient is also shown as the horizontal line, surrounded by a shaded region defined by the standard error of the mean

of the components considered. The first two components ("topics" and "languages") are structural data, therefore the Jaccard coefficient (1) is used to build the ranking. The similarity of the third component ("issues"), is measured using (2), since it is a set of textual data. Finally, the "total" provides a similarity aggregation of the previous components with uniform weight.

When interpreting the results in Fig. 4, it should be noted that this metric is limited by human subjectivity that occurs when performing the evaluation. To have a quantitative measure of this component, the instances of the same profiles studied by more than one evaluator have been compared, thus obtaining a distribution of coefficients that indicate their level of agreement. Figure 4 includes as a reference the average value of such a distribution, as well as its standard error, represented by the shaded region.

The fact that the distribution of the Kendall coefficient is mainly located above zero, with values similar to those corresponding to the deviation detected for the human factor, shows that the behavior of the system is similar to that of the humans, and not a random permutation. Among the studied components, the one that has had the best performance has been the one extracted from the similarity of the issues. This may be because the languages and topics recorded in the repositories include less relevant components, but when taken into account in the similarity measures they introduce noise in them.

## 5 Conclusions and future work

The first conclusion that can be drawn from this research is that the use of this type of tool offers an advantage in the development of the software, since the automatic selection of the most suitable profile allows to fix the bug faster than if a manual pre-selection had to be carried out. Therefore, the use of this type of methodology is recommendable.

The system is easily adaptable and portable to all kinds of projects, regardless of the technologies used (knowledge has been extracted from 3 large-scale repositories and that knowledge has been tested with 14 GitHub projects using well-differentiated technologies) and technical profiles. This characteristic, together with the demonstration that the proposed methodology is capable of producing good results, can be considered as the main contribution of the research. This makes the designed solution usable on employability portals specialized in technology profile selection, analyzing on the one hand unlabeled job offers and on the other hand unlabeled user profiles. Similarly, it could be applicable to portals that facilitate the deployment of open-source software projects, as it could provide great results in the search for collaborators. Therefore, it can be said that the proposed system applied in environments such as those described, will improve the quality and agility of employability processes.

At a technical level, the approach of the proposed methodology is novel and has allowed to extract knowledge from large-scale repositories associated with three open-source software projects and to apply it efficiently in non-technically related projects, which validates that the proposed technical solution is applicable in multiple environments.

As explained before, the proposed methodology can be adapted to employability portals specialized in the search of technological profiles to participate in software projects, both initiated and non-initiated. The main drawback is that the information needed to make a system robust enough is owned by such portals and no repository with these characteristics has been found. Therefore, work is being done to find portals that have the characteristics to be able to carry out a new study based on this methodology, but with this new functionality.

At the technical level, research is being done on a supervised classifier that will operate more intelligently with similarity measurements to try to more accurately reproduce the human scale.

**Acknowledgements** The authors would like to express their gratitude to the profile evaluators.

## Compliance with ethical standards

**Conflict of interest** The authors declare that they have no conflict of interest.

## References

1. Biswas UN, Allard K, Pousette A, Härenstam A (2017) The information technology sector. In: Biswas UN, Allard K, Pousette A, Härenstam A (eds) Understanding attractive work in a globalized world. Springer, Berlin, pp 151–174
2. Chen TH, Thomas SW, Hemmati H, Nagappan M, Hassan AE (2017) An empirical study on the effect of testing on code quality using topic models: a case study on software development systems. IEEE Trans Reliab 66(3):806
3. Mohagheghi P, Jørgensen M (2017) What contributes to the success of IT projects? an empirical study of IT projects in the Norwegian public sector. JSW 12(9):751
4. Fisher J, Koning D, Ludwigsen A, et al. (2013) Utilizing Atlassian JIRA for large-scale software development management. In: 14th International conference on accelerator & large experimental physics control systems (ICALEPCS)
5. GitHub Inc., GitHub (2007) https://github.com/. Accessed 14 Jul 2020
6. Lamkanfi A, Pérez J, Demeyer S (2013) The eclipse and mozilla defect tracking dataset: a genuine dataset for mining bug information. In: 2013 10th working conference on mining software repositories (MSR) (IEEE), pp. 203–206
7. Ortu M, Destefanis G, Adams B, Murgia A, Marchesi M, Tonelli R (2015) The JIRA repository dataset: Understanding social

aspects of software development. In: Proceedings of the 11th international conference on predictive models and data analytics in software engineering , pp. 1–4
8. Lu S, Park S, Seo E, Zhou Y (2008) Learning from mistakes: a comprehensive study on real world concurrency bug characteristics. In: Proceedings of the 13th international conference on architectural support for programming languages and operating systems, pp. 329–339
9. Sun C, Lo D, Khoo SC, Jiang J (2011) Towards more accurate retrieval of duplicate bug reports. In: 2011 26th IEEE/ACM International conference on automated software engineering (ASE 2011) (IEEE), pp. 253–262
10. Lazar A, Ritchey S, Sharif B (2014) Generating duplicate bug datasets. In: Proceedings of the 11th working conference on mining software repositories , pp. 392–395
11. Zhang W, Challis C (2019) Software component prediction for bug reports. In: Asian conference on machine learning , pp. 806–821
12. Terdchanakul P, Hata H, Phannachitta P, Matsumoto K (2017) Bug or not? Bug report classification using n-gram idf. In: 2017 IEEE International conference on software maintenance and evolution (ICSME) (IEEE), pp. 534–538
13. Saad A, Saad M, Emaduddin SM, Ullah R (2019) Optimization of bug reporting system (BRS): artificial intelligence based method to handle duplicate bug report. In: International conference on intelligent technologies and applications. Springer, pp. 118–128
14. Baarah A, Aloqaily A, Salah Z, Zamzeer M, Sallam M (2019) Machine learning approaches for predicting the severity level of software bug reports in closed source projects. Mach Learn. https://doi.org/10.14569/IJACSA.2019.0100836
15. Zhang W, Challis C (2019) Automatic bug priority prediction using DNN based regression. In: The international conference on natural computation, fuzzy systems and knowledge discovery (Springer), pp. 333–340
16. Xie Q, Wen Z, Zhu J, Gao C, Zheng Z (2018) Detecting duplicate bug reports with convolutional neural networks. In: 2018 25th Asia-pacific software engineering conference (APSEC) (IEEE), pp. 416–425
17. Kaur A, Goyal S (2020) Comments-based analysis of a bug report collection system and its applications. from data gathering to data comprehension, intelligent data analysis
18. Wang C, Li Y, Chen L, Huang W, Zhou Y, Xu B (2020) Examining the effects of developer familiarity on bug fixing. J Syst Softw 169:110667
19. Chamoso P, Rivas A, Rodríguez S, Bajo J (2018) Relationship recommender system in a business and employment-oriented social network. Inf Sci 433:204
20. Chamoso P, Bartolomé Á, García-Retuerta D, Prieto J, De La Prieta F (2020) Profile generation system using artificial intelligence for information recovery and analysis. J Ambient Intell Humanized Comput 11:1–10
21. Chachra A, Mehndiratta P, Gupta M (2017) Sentiment analysis of text using deep convolution neural networks. In: 2017 Tenth international conference on contemporary computing (IC3), pp. 1–6
22. Zennaki O, Semmar N, Besacier L (2016) Inducing multilingual text analysis tools using bidirectional recurrent neural networks, arXiv preprint arXiv:1609.09382
23. Mukalov P, Zelinskyi O, Levkovych R, Tarnavskyi P, Pylyp A, Shakhovska N (2019) Development of system for auto-tagging articles, based on neural network. In: DCOLINS , pp. 106–115
24. Zhou C, Sun C, Liu Z, Lau F (2015) A C-LSTM neural network for text classification, arXiv preprint arXiv:1511.08630
25. Mikolov T, Chen K, Corrado GS, Dean JA (2015) Computing numeric representations of words in a high-dimensional space, Computing numeric representations of words in a high-dimensional space . US Patent 9,037,464
26. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781
27. Jebari C, Cobo MJ, Herrera-Viedma E (2018) A new approach for implicit citation extraction. In: International conference on intelligent data engineering and automated learning (Springer), pp. 121–129
28. Lau JH, Baldwin T (2016) An empirical evaluation of doc2vec with practical insights into document embedding generation, arXiv preprint arXiv:1607.05368
29. Jang B, Kim I, Kim JW (2019) Word2vec convolutional neural networks for classification of news articles and tweets. PLoS ONE 14(8):e0220976
30. Acosta J, Lamaute N, Luo M, Finkelstein E, Andreea C (2017) Sentiment analysis of twitter messages using word2vec. In: Proceedings of student-faculty research day, CSIS, Pace University **7**
31. Vargas-Calderón V, Camargo JE (2019) Characterization of citizens using word2vec and latent topic analysis in a large set of tweets. Cities 92:187
32. Joshi A, Kale S, Chandel S, Pal DK (2015) Likert scale: explored and explained. Current J Appl Sci Technol 7(4):396–403
33. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems , pp. 3111–3119
34. Řehůřek R, Sojka P ( 2010) Software framework for topic modelling with large corpora. In: Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks (ELRA, Valletta, Malta), pp. 45–50. http://is.muni.cz/publication/884893/en
35. Lvd Maaten, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9:2579
36. Kendall MG (1945) The treatment of ties in ranking problems. Biometrika 33(3):239–251