International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-11 Issue-5, June 2022

Recommender Model for Secure Software Engineering using Cosine Similarity Measures

Astrit Desku, Bujar Raufi, Artan Luma, Besnik Selimi

Abstract*: One of the essential components of Recommender Systems in Software Engineering is a static analysis that is answerable for producing recommendations for users. There are different techniques for how static analysis is carried out in recommender systems. This paper drafts a technique for the creation of recommendations using Cosine Similarity. Evaluation of such a system is done by using precision, recall, and so-called Dice similarity coefficient. Ground truth evaluations consisted of using experienced software developers for testing the recommendations. Also, statistical T-test has been applied in comparing the means of the two evaluated approaches. These tests point out the significant difference between the two compared sets.

Keywords: Recommender Systems, Cosine Similarity, Software Engineering.

I. INTRODUCTION

Today recommendation engines are used in many fields, starting from mini news headlines on various social media pages, online articles from various news portals, providing streaming services with watchable content, friendly requests to join groups or individuals, all the way to various e-commerce sites with their products available for purchase [1, 2]. During daily use, recommendation engines can play an important role in users' decision-making regarding whether they like the platform that serves the data [3, 4].

In the sector of healthcare, there are a lot of systems that help medical staff with decision-making based on the recommendations provided by the system. These systems will be able to predict patients health based on their lifestyle or health issues from their relatives [5, 6]. In the field of Software Engineering, usually, most software developers want to reuse their functional code or other developer's code in their daily development activity. This reduces development time and uses proven functional code. Therefore, a lot of developers when searching for useful code, they also get served by background applications that utilize some machine learning algorithms that increase their chances of finding the correct code [1]. One such approach is to use Recommendation Systems in Software Engineering (RSSEs). The RSSEs are useful in some dimensions, like improving system quality, helping developers with better decision-making, increasing efficiency, and decreasing expenses during the development processes and maintenance phase [7]. RSSE can be useful in any phase of the development process i.e. requirement engineering, designing, programming, and testing [8]. There are different approaches and techniques for modeling recommender systems, like collaborative filtering, content-based filtering, and knowledge-based recommendation [9]. This paper model a recommender system for secure software engineering using cosine similarity measurements continued from the datasets produced during our previous paper in [10].

II. RELATED WORK

Data analysis for recommendation engines is mainly statistical analysis [11] and those most relevant to this paper are mentioned below. Adaptation of the cosine similarity method as described by [12] was used to create product similarity evaluation leading to the creation of customer recommendation score applied for content-based filtering. Their model was applied to both the customers and products. Using customer's transaction behavior they proposed new products that the customer would more likely purchase. Authors calculated the accuracy of their recommendations through precision and recall for scores on product similarity, achieving 100% and 93.47%, respectively. In [13] it was presented a similarity function by using weighted distance similarity to find similarities between users in the Movie Lens dataset.

Three different algorithms (K-means clustering, hierarchical clustering, and cosine similarity) are used in [14] to find similarities between the description of movies and books. Furthermore, in [15] cosine similarity has been applied to classifying Economic Journal Articles in the Indonesian Language.

Authors used only the content from titles and abstracts. Similarly, authors in [16] have applied cosine similarity to the cancer category extracted from website-based news articles. In the programming area for code recommendations for C and C++, the authors of [17] used machine learning in a data-driven approach, extracting features at two levels of granularity to create their build. Functional level usage of Control Flow Graph (CFG) allowed them to extract features from various operations that appear in basic blocks, variable definition, and usage. Measuring and identifying similar documents/reports by authors in [18] is based on class-based indexing term weighting schemes such as TFIDF and TFICF.

Manuscript received on 30 May 2022.
Revised Manuscript received on 01 June 2022.
Manuscript published on 30 June 2022.
* Correspondence Author
Astrit Desku*, Faculty of Contemporary Sciences and Technologies, South East European University, Tetovo, North Macedonia. Email: ad23613@seeu.edu.mk
Bujar Raufi, Faculty of Contemporary Sciences and Technologies, South East European University, Tetovo, North Macedonia. Email: b.raufi@seeu.edu.mk
Artan Luma, Faculty of Contemporary Sciences and Technologies, South East European University, Tetovo, North Macedonia. Email: a.luma@seeu.edu.mk
Besnik Selimi, Faculty of Contemporary Sciences and Technologies, South East European University, Tetovo, North Macedonia. Email: b.selimi@seeu.edu.mk

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

III. RECOMMENDER MODEL USING COSINE SIMILARITY MEASURE

Today exists different algorithms for similarity measurements in text mining applications or information retrievals, such as Euclidean distance-based metric, Jaccard, Dice, Jensen- Shannon Divergence, and Cosine Similarity are one of the most used [19,20]. By applying this technique, different models can be generated with each different measure based on the distance was chosen or distance measure [21]. Based on the definition of the techniques of Cosine similarity, the measure is a result of the cosine of the angle between two vectors of an inner product space [22]. Two cosine vectors that can be aligned inside the identical orientation can have a similarity size of 1, while vectors aligned perpendicularly can have a similarity of 0 [23]. The cosine similarity for two vectors given is defined as:

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\| \, \|B\|} = \frac{\sum_{i=1}^{n} A_i \, B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \quad \sqrt{\sum_{i=1}^{n} B_i^2}} \quad (1)$$

Documents in datasets usually are long vectors that have a consideration of variables or attributes. In [11] to explain the measure of two vectors it was given a simple example with two vectors consisting of the following attributes: X (1,2,0,0,3,4,0) and Y (5,0,0,6,7,0,0).

Calculation of the cosine similarity index for the given example above is shown in Fig. 1, below

$$x \cdot y = \sqrt{1\times5+2\times0+0\times0+0\times6+3\times7+4\times0+0\times0} = 5.1$$
$$||x|| = \sqrt{1\times1+2\times2+0\times0+0\times0+3\times3+4\times4+0\times0} = 5.5$$
$$||y|| = \sqrt{5\times5+0\times0+0\times0+6\times6+7\times7+0\times0+0\times0} = 10.5$$
$$\text{Cosine similarity}(|x\cdot y|) = \frac{x\cdot y}{||x|| \;\; ||y||} = \frac{5.1}{5.5\times10.5} = 0.08$$

Fig. 1. The cosine similarity index calculation [4].

The following sections present a model of recommendation using cosine similarity. The proposed model will use datasets presented in [10] by our previous work. The model uses a dataset with three variants of them:

- CFG Dataset - contains data based on Control Flow Graph structure
- CSI Dataset - contains calculated Cosine Similarity Index for CFG Dataset
- Class Method Dataset – contains Methods of Classes separated from the original dataset.

The CFG Dataset is a dataset created by using the structure of Control Flow Graph (CFG). A control flow graph is a graph defined by the following formula:

$$G = (N, E), \quad (2)$$

Where the nodes are marked as N and represent statements of the procedure and the edges marked as E represents the transfer of the control between two statements. [24].

Using CFG will be able to track the flow of execution and variable states. A node in a CFG represents a block of code that will always run sequentially. Edges in a CFG represent possible paths of execution. We have to implement a .NET solution explained in detail in our previous work in [15], by which the dataset was transformed as a dataset with CFG structure The CSI Dataset is a dataset that was created after the calculated cosine similarity index for CFG Dataset. The Class Method Dataset is a dataset created by the original dataset after applying another .Net solution by which methods of the class are separated from the class body. This dataset structure has three columns: *Class*, *Method Header* and *Method Body*. We used a pattern based on regular expression or regex to identify the header of methods. A regular expression by definition is a pattern that the regular expression engine attempts to match in the input text. A pattern consists of one or more character literals, operators, or constructs.

Fig. 2 presents the relationship between the datasets explained above.

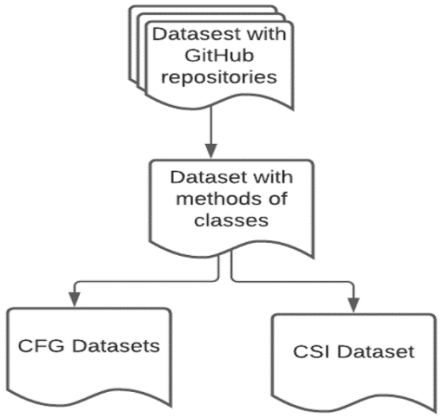

Fig. 2. The relationship between datasets

The model use as an input parameter a snippet code from developer users and return as output a list of class methods as a recommendation Developer user will be able to choose which proposed recommendation to use in their solution, in the case when the recommendation engine propose more than one recommendation.

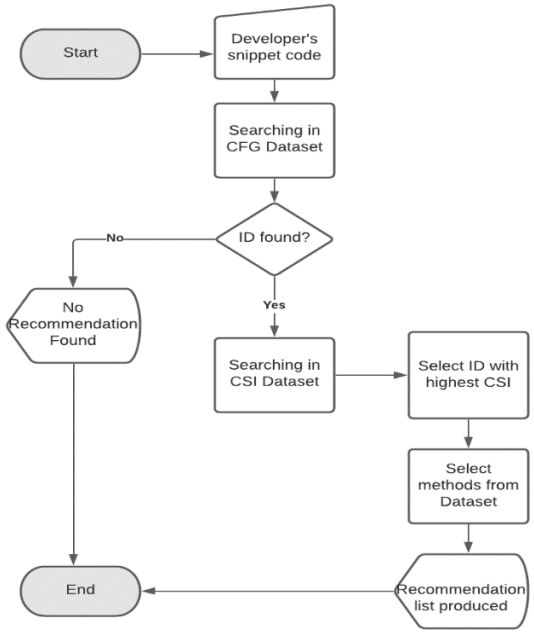

Fig. 3. Recommender Model using Cosine Similarity

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

A. How the algorithm works

In the following, it was the procedure of how the algorithm works. To explain how the algorithm of the model generates recommendations let's use as input parameter this value 'aesDecrypt'. At the same time, this value can be considered as developer user snippet code:

- For this value algorithm first search in 'CFG Dataset' to find the most similarity value.
- The next step is searching in 'CSI Dataset' to find data with the highest cosine similarity index.
- Because the algorithm will return a lot of values, we decide to get the top five selected values with the highest CSI. For these data, the algorithm will search in 'Dataset with methods class' to give the user full methods of classes as a recommendation. In this case, the recommender engine will propose a recommendation as it is shown in the figure below.

MethodBody	CosineSimIndex
this.aesKey = new MetroFramework.Controls.Metrc	67%
this.components = new System.ComponentModel.(	65%
this.components = new System.ComponentModel.(	65%
this.components = new System.ComponentModel.(	65%
this.metroLabel1 = new MetroFramework.Controls.	65%

Fig. 4. Results after applying the algorithm for recommendation

IV. RECOMMENDER EVALUATION OF OUR MODEL

The great challenge from a software engineering point of view is addressing the relevance of recommendations for a programmer's code by the recommendation system tools that will be used by the developer. Especially in cases when more than one recommendation is suggested for the developer.

Establishing the quality of a recommender system usually consists of comparison from predictions from different algorithms, performances, or various mode sizes [25, 26]. But since the RSSE are in the evolutionary stage done on limited hardware resources such as developer's machines, therefore these factors should be considered also, when evaluating these systems. Our proposed model for generation recommendation uses a cosine similarity approach and has two evaluation criteria. First is automatically measuring the effectiveness of the recommendation system through the combined use of precision, recall, and F-measure. The second is ground truth measurement of recommendations through the experienced developers Since the recommendation system tries to mimic how human uses the recommendation, from entering a query, the system returns the results in form of a recommendation, and the human then uses one or more recommendations based on how relevant it is for the task. The recommendation system tool could be adapted by code change to use the same query again and potentially get similar, better, or worse results [25].

A. Performance measures

Assessing recommendation system effectiveness' is done using precision, recall, and F1 measures which are applied to the results of the queries. This provides a) all relevant recommendations and b) filter's out not relevant recommendations Precision quantifies the number of true positive predictions that belong to the positive class, i.e. Precision is defined as the number of useful (relevant) recommendations over the total number of recommendations,

$$\text{Precision} = \frac{\text{Recommendations}_{\text{relevant}}}{Recommendations_{total\ number}} \quad (3)$$

Recall quantifies the number of true positive predictions made out of all positive examples in the dataset, i.e. Recall is defined as the number of useful recommendations over the expected recommendations [25, 26]

$$\text{Recall} = \frac{\text{Recommendations}_{\text{relevant}}}{Recommendations_{\text{expected}}} \quad (4)$$

F-Measure or F-score provides a single score that balances both the concerns of precision and recall in one number. The F-Measure is interpreted as the harmonic mean of the precision and recall and has been widely used by the researchers [26, 25]:

$$\text{F} = \frac{(1+\beta^2) * precision * recall}{\beta^2 * precision + recall} \quad (5)$$

The F-Measure allows being put greater emphasis on either recall or precision by varying the assigning of the β parameter. In our case (β = 1) both precision and recall have the same importance therefore β = 1, which is also known as. F1 measure [27, 26].

In our case evaluating the cosine similarity for the recommendation system was done using the following rules for performance measures:

- Relevant recommendations: produced recommendations with GitHub commit of more than five and a cosine similarity index of more than 0.6
- Expected recommendations: produced recommendation with GitHub commits of more than five
- A Total number of recommendations: top 5 recommendations with the highest score on the cosine similarity index.

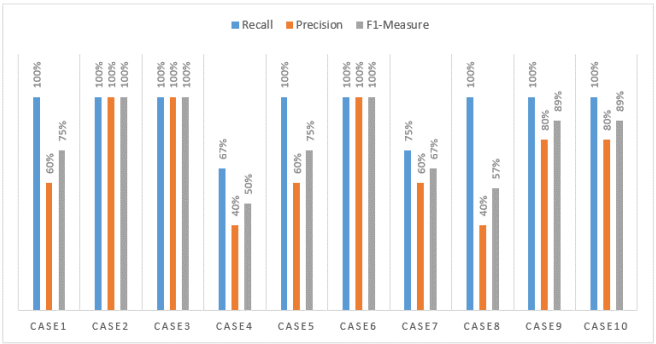

Fig. 5. Recommendation Performance measure

Average values of the performance measurements for data in the Fig. 5 above: Recall = 0.94, Precision = 0.72 and F1-measure = 0.80

B. Subjective Evaluation

As it was mentioned previously, the evaluation process of the RSSE simulates the usage of the tool by a human user, therefore ground truth from the user developer was used in this case. There are dozen user developers which will be evaluating the same cases to calculate performance.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

Their response is taken with a choice of relevant values from one to five, for each of the cases.

A value of one represents the lowest relevance, otherwise, the value of five represents the higher relevance. Then precision, recall, and F1-Measure are calculated for each evaluated case and these are shown in Fig. 6 below.

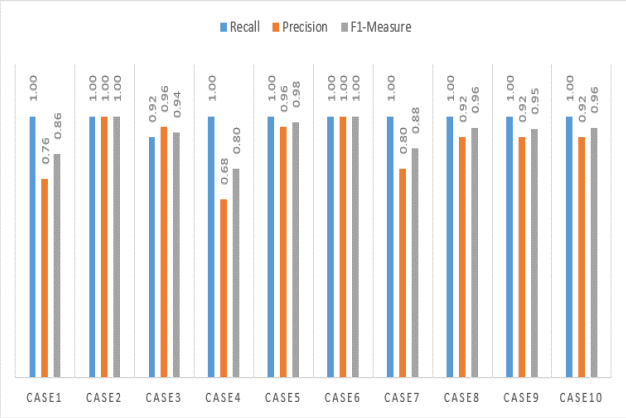

Fig. 6. The performance measure for subjective evaluation

Total average values of the performance measurements for data in the Fig. 6 above are: Recall = 0.99, Precision = 0.89 and F1-measure= 0.93.

V. CONCLUSION

This paper presented an approach to generating a recommendation for Software Engineering using recommender systems and applying the cosine similarity technique to a dataset composed of a Control Flow Graph structure. The results are measured using two types of evaluations a) automatic performance measure performed by model and b) subjective evaluation performed by experienced developers. Based on the results of values for precision, recall, and F1-measure, as expected the subjective evaluation performed better. This opens an opportunity to further explore model creation, potentially with API using MS Visual Studio integrated programming options.

REFERENCES

1. B. D. Vijay Kotu, "Recommendation Engines," in Data Science - Concepts and Practice (Second Edition), Morgan Kaufmann, 2019, pp. 343-394. [CrossRef]
2. M. Bruch, M. Monperrus and M. Mezini, "Learning from Examples to Improve Code Completion Systems," in International Symposium on the Foundations of Software, 2009. [CrossRef]
3. M. P. Robillard, W. Maalej, R. J. Walker and T. Zimmermann, Recommendation Systems in Software Engineering, Heidelberg New York Dordrecht London: Springer-Verlag, 2014. [CrossRef]
4. A. Whitten and J. D. Tygar, "Why Johnny can't encrypt: a usability evaluation of PGP 5.0," in Proceedings of the 8th conference on USENIX Security Symposium - Volume 8, Washington, 1999.
5. Bhatti, U. A., Huang, M., Wu, D., Zhang, Y., Mehmood, A., & Han, H. (2019). Recommendation system using feature extraction and pattern recognition in clinical care systems. Enterprise information systems, 13(3), 329-351. [CrossRef]
6. Sahoo, A. K., Mallik, S., Pradhan, C., Mishra, B. S. P., Barik, R. K., & Das, H. (2019). Intelligence-based health recommendation system using big data analytics. In Big data analytics for intelligent healthcare management (pp. 227-246). Academic Press. [CrossRef]
7. Gasparic, M., & Janes, A. (2016). What recommendation systems for software engineering recommend: A systematic literature review. Journal of Systems and Software, 113, 101-113. [CrossRef]
8. Pakdeetrakulwong, U., Wongthongtham, P., & Siricharoen, W. V. (2014, December). Recommendation systems for software engineering: A survey from software development life cycle phase perspective. In The 9th International Conference for Internet Technology and Secured Transactions (ICITST-2014) (pp. 137-142). IEEE. [CrossRef]
9. Felfernig, A., Jeran, M., Ninaus, G., Reinfrank, F., Reiterer, S., & Stettinger, M. (2014). Basic approaches in recommendation systems. In Recommendation Systems in Software Engineering (pp. 15-37). Springer, Berlin, Heidelberg. [CrossRef]
10. Desku, Astrit, et al. "Cosine Similarity through Control Flow Graphs For Secure Software Engineering." 2021 International Conference on Engineering and Emerging Technologies (ICEET). IEEE, 2021. [CrossRef]
11. M. P. Robillard, R. J. Walker and T. Zimmermann, "Development tools: Recommendation Systems for Software Engineering," I E E E S O F T WA R E www. c omp u t e r . o r g / s o f tw a r e.
12. Fiarni, Cut, and Herastia Maharani. "Product Recommendation System Design Using Cosine Similarity and Content-based Filtering Methods." IJITEE (International Journal of Information Technology and Electrical Engineering) 3.2 (2019): 42-48 [CrossRef]
13. Huang, B. H., & Dai, B. R. (2015, June). A weighted distance similarity model to improve the accuracy of collaborative recommender system. In 2015 16th IEEE International Conference on Mobile Data Management (Vol. 2, pp. 104-109). IEEE. [CrossRef]
14. Nawar, A., Toma, N. T., Al Mamun, S., Kaiser, M. S., Mahmud, M., & Rahman, M. A. (2021, October). Cross-Content Recommendation between Movie and Book using Machine Learning. In 2021 IEEE 15th International Conference on Application of Information and Communication Technologies (AICT) (pp. 1-6). IEEE. [CrossRef]
15. Ristanti, Putri Yuni, Aji Prasetya Wibawa, and Utomo Pujianto. "Cosine similarity for title and abstract of economic journal classification." 2019 5th International Conference on Science in Information Technology (ICSITech). IEEE, 2019. [CrossRef]
16. Kohila, D. K. A. R. "Text Mining: Text Similarity Measure for News Articles Based On Global." Glob. J. Eng. Sci. Res. Manag. 3.7 (2016): 35-42.
17. J. A. Harer, L. Kim, R. L. Russell, O. Ozdemir, O. Ozdemir, E. Antelman and S. Chin, "Automated software vulnerability detection with machine learning," in ResearchGate, 2018.
18. Irananda, Syahroni Wahyu. "Measure the Similarity of Complaint Document Using Cosine Similarity Based on Class-Based Indexing." International Journal of Computer Applications Technology and Research, Volume 7–Issue 08, 292-296, 2018 [CrossRef]
19. Salton, G.: Automatic Text Processing: The Transformation, Analysis, and Retrieval ofInformation by Computer. (1989) Addison-Wesley Longman Publishing, Boston, MA.
20. Li M., Chen X., Li X., Ma B., and Vitanyi P. M.B. : The Similarity Metric. IEEE Transactions on Information Theory, 50(12): 3250-3264 (2004) [CrossRef]
21. K. Vijay and D. Bala, "Classification," in Data Science (Second Edition), organ Kaufmann, 2019, pp. 65-163. [CrossRef]
22. Li, B., & Han, L. (2013, October). Distance weighted cosine similarity measure for text classification. In International conference on intelligent data engineering and automated learning (pp. 611-618). Springer, Berlin, Heidelberg. [CrossRef]
23. B. Li and L. Han, "Distance Weighted Cosine Similarity Measure for Text Classification," in International Conference on Intelligent Data Engineering and Automated Learning, Berlin, Heidelberg, 2013.
24. K. Ottenstein and L. Ottenstein, "The program dependence graph in a software development environment," ACM Sigplan Notices, 1984. [CrossRef]
25. M. Bruch, T. Schäfer and M. Mezini, "On Evaluating Recommender Systems for API Usages," in Proceedings of the 2008 international workshop on Recommendation systems for software engineering, 2008. [CrossRef]
26. M. Bruch, M. Monperrus and M. Mezini, "Learning from Examples to Improve Code Completion Systems," in International Symposium on the Foundations of Software, 2009. [CrossRef]
27. Kohila, D. K. A. R. "Text Mining: Text Similarity Measure for News Articles Based On Global." Glob. J. Eng. Sci. Res. Manag. 3.7 (2016): 35-42.

© Copyright: All rights reserved.

AUTHORS PROFILE

Astrit Desku, currently is PhD student at the Faculty of Contemporary Sciences and Technologies, South East European University. Astrit does research in Software Development, Software Engineering, Recommender Systems in Software Engineering, Data Mining and Databases

Bujar Raufi, currently works as Associate Professor at the Faculty of Contemporary Sciences and Technologies, South East European University. Bujar does research in Adaptive User Interfaces, Semantic Web, Data Mining, Computer Graphics and Computer Communications (Networks)

Artan Luma, currently works as Associate Professor at the Faculty of Contemporary Sciences and Technologies, South East European University. Artan does research in Algorithms, IT Security, Computer Network Security and Cloud Computing

Besnik Selimi, currently works as Associate Professor at the Faculty of Contemporary Sciences and Technologies, South East European University. Besnik does research in Software Testing, Software Architecture, IT Security, Algorithms, Cloud Computing and Cryptography

© Copyright: All rights reserved.