Recommender System for Software Engineering using SQL Semantic Search

Astrit Desku, Bujar Raufi, Artan Luma, Besnik Selimi

Abstract*: Recommender Systems are software tools that can assist developers with a wide range of activities, from reusing codes to suggest developers what to do during development of these systems. This paper outlines an approach to generating recommendation using SQL Semantic Search. Performance measurement of this recommender system is conducted by calculating precision, recall and F1-measure. Subjective evaluations consisted of 10 experienced developers for validating the recommendation. A statistical test t-Test is used to compare the means of two approaches of evaluations.

Keywords: Recommender Systems, Semantic Search, Software Engineering.

I. INTRODUCTION

Recommender Systems for Software Engineering (RSSEs) are software tools that can assist developers with a wide range of activities, from reusing codes to suggest developers what to do during development [1]. These tools can be used also to guide and recommend a developer for next activities, based on codes write from other developers.

Software developers have always used tools to perform their work. In the earliest days of the discipline, the tools provided basic compilation and assembly functionality [2]. Then came tools and environments that increasingly provided sophisticated data about the software under development [2] [3]. Nowadays, the systematic and large-scale accumulation of software engineering data opened up new opportunities for the creation of tools or APIs (Application Programming Interfaces), that infer information estimated to be helpful to developers in a given context. One way for helping developers during daily development activities is reusing code from previous project that they have developed or reuse code from projects developed from other developers. Today, there are online platforms (like Stack. Overflow. GitHub etc.), in which developer posts their projects or snipped code from projects, these codes therefore can be used by other developers in their own projects.

Manuscript received on 31 March 2022.
Revised Manuscript received on 06 April 2022.
Manuscript published on 30 April 2022.

* Correspondence Author

Astrit Desku*, Faculty of Contemporary Sciences and Technologies, South East European University, Tetovo, North Macedonia. E-mail: ad23613@seeu.edu.mk

Bujar Raufi, Faculty of Contemporary Sciences and Technologies, South East European University, Tetovo, North Macedonia. E-mail: b.raufi@seeu.edu.mk

Artan Luma, Faculty of Contemporary Sciences and Technologies, South East European University, Tetovo, North Macedonia. E-mail: a.luma@seeu.edu.mk

Besnik Selimi, Faculty of Contemporary Sciences and Technologies, South East European University, Tetovo, North Macedonia. E-mail: b.selimi@seeu.edu.mk

II. RELATED WORK

There are a lot of studies that treat the Recommender engine and data analyses, as one of the most important part of recommender systems. In [4] was proposed a recommender system called Precise, which focuses on parameter call sites. In [5] mainly use identifiers to predict method calls. They extract all identifiers used in the source code and split names on camel-case humps. Even though control structures are not part of their model, they consider control structure keywords in the tokenization. They apply text-mining techniques such as stemming or removing stop words to unify the collected tokens. In [6] it was implemented a tool that learns correct API usage from interactions of developers in their IDE. When code completion is triggered in the IDE, they extract features from the structural context around the trigger point that include the type, definition, enclosing statement, expression type, enclosing method. In [7] it was presented Context Sensitive Code Completion - CSCC system that is built on a simple and efficient algorithm. In [8] is proposed CODEWEB, a tool that can be used to identify “reuse relationships” in source code. In [9] authors propose DMMC to detect missing method calls. They extract object usages that describe how an object instance is used. They encode the type of the object, the enclosing method, and all calls on it. In [10] authors present an approach to build finite-state-automatons that describe valid protocols for using a specific type. The approach is based on their own framework that can be used to mine method sequences [11]. In [12] authors present a data-driven approach to vulnerability detection using machine learning, specifically applied to C and C++ programs. During the build process, they extract features at two levels of granularity. At the function level, they extract the Control Flow Graph (CFG) of the function. Within the control flow graph, they extract features about the operations happening in each basic block and the definition and use of variables. In this paper it was presented an approach of generating recommendations for RSSE by using SQL Semantic Search. Data for simulating results are used from datasets presents in our previous work in [13].

III. RECOMMENDER MODEL USING SEMANTIC SIMILARITY

Semantic similarity is defined over a set of documents or terms, where the idea of distance between items is based on the likeness of their meaning or semantic content as opposed to lexicographical similarity [14]. The semantic similarity is often confused with semantic relatedness. Semantic relatedness includes any relation between two terms, while semantic similarity only includes "is a" relations. For example, "car" is similar to "bus", but is also related to "road" and "driving.

In [15] define semantic similarity as the similarity between two classes of objects in a taxonomy. A class C_1 in the taxonomy is considered to be a subclass of C_2 if all the members of C_1 are also members of C_2. Therefore, the similarity between two classes is based on how closely they are related in the taxonomy.

There are different let say techniques for measure similarity as are: Dice coefficient, distance-based similarity, Resnik (IJCAI 1995), but much more popular is Wu & Palmer (ACL 1994) [15]. Wu and Palmer in [16] proposed the following similarity measure based on use of subclass links between classes:

$$sim(C_1, C_2) = \frac{2N_3}{N_1 + N_2 + 2N_3} \qquad (1)$$

Where N_1 and N_2 are the number of subclass edges from C_1 and C_2 to their closest common superclass; N_3 is the number of subclass edges from the closest common superclass of $C_1$and C_2 to the root class in the taxonomy.

In this paper we will use SQL Semantic Search solution to find recommendations for users. Semantic search tries to improve search by understanding the contextual meaning of the terms and tries to provide the most accurate answer from a given document repository. The technologies behind semantic search are mostly used to access unstructured data.

There are several techniques to implement semantic search [17]. One type of structured data are ontologies. An ontology formally describes available concepts in a specific domain. One possibility to query structured data is conceptual graph matching. In this method each query and the data are represented as trees of concepts called ontologies. The search engine compares the query with each tree in the database and finds the best matching tree [18]. Based on Microsoft documentations, the SQL Semantic Search supports only unigrams. A unigram is a single word, whereas n-grams are word combinations, also known as term and phrase extraction respectively [19]. Semantic Search provides deep insight into unstructured documents stored in SQL Server databases by extracting and indexing statistically relevant key phrases. Then it uses these key phrases to identify and index documents that are similar or related.

Semantic search builds upon the existing full-text search feature in SQL Server. While full-text search lets query the words in a document, semantic search lets query the meaning of the document. Solutions that are now possible include automatic tag extraction, related content discovery, and hierarchical navigation across similar content. In order to use semantic search in SQL Server, first step is to install Semantic Language Database, which is a database with statistical base data to classify the keywords found in the indexed documents. To use semantic search on a text column of a regular table it is needed to create a semantic search index for that text column. The importance of the keywords it is determined by using a variation of the TF-IDF.

To find similar or related documents in a specific column we will use a function called *SEMANTICSIMILARITYTABLE*.

It returns a table of zero, one, or more rows whose content in the specified column is semantically similar to the specified document [16].

In our case after querying a value *'aesDecrypt'* in our database the result will be as it is in the Fig1 below. The first column "*MatchedTile*" presents content similarity from queried data, "*MethodBody*" presents recommendations found in the dataset, "*Socre*" presents index score of the similarity and "*NoCommits*" presents number of the commits in the GitHub for that repository.

MatchedTitle	MethodBody	Score	NoCommits
this.aesKey = new MetroFramework.Controls.MetroTextBox()	this.aesKey = new MetroFramework.Controls.MetroTextBox(); this.aesMess	100%	15
this.aesMessage = new MetroFramework.Controls.MetroTextBox()	this.aesKey = new MetroFramework.Controls.MetroTextBox(); this.aesMess	100%	15
this.aesResult = new MetroFramework.Controls.MetroTextBox()	this.aesKey = new MetroFramework.Controls.MetroTextBox(); this.aesMes	100%	15
this.aesEncrypt = new MetroFramework.Controls.MetroButton()	this.aesKey = new MetroFramework.Controls.MetroTextBox(); this.aesMess	100%	15
v.AddOptional(controls)	Asn1EncodableVector v = new Asn1EncodableVector(certReqId, certTemplate);	87%	5
return controls	List<Control> controls = new List<Control>(); foreach (Control child in pare	67%	256
controls.AddRange(GetSelfAndChildrenRecursive(child))	List<Control> controls = new List<Control>(); foreach (Control child in pare	65%	256
this.groupBoxAlg.Controls.Add(this.radioButtonDES)	this.components = new System.ComponentModel.Container(); System.Com	63%	144
this.Controls.Add(this.progressBar1)	this.components = new System.ComponentModel.Container(); System.Com	63%	144

Fig. 1. Results after applying semantic search

IV. RECOMMANDER MODEL EVALUATION

One of the most challenge in the area of Recommendation Systems for Software Engineering is how relevant are the recommendations produced by tool and delivered to user developer. By which premises should base user developer to accept the relevant recommendation in cases when system tool recommend more than one recommendations. The evaluation of recommender systems is usually focused on comparing prediction quality of different algorithms, performance or mode size [20]. However, RSSE are designed for the use by humans on typical developer achiness with limited resources and evaluations of recommender systems should take this into account. In this paper we will present two approach of evaluation a) performance measurements by model and b) subjective evaluation by 10 experienced developer.

A. Performance measures

To assess the effectiveness of a recommender system for a given query we use recall, precision and F1 measures that express the ability of the system to

- Find all relevant recommendations, and
- Filter recommendations not relevant for the given query

Recall is defined as the ratio between the relevant recommendations made by the system for the given query and the total number of expected recommendations that should have been made. Precision is defined as the ratio between relevant recommendations made and the total number of recommendations made by the system for a particular query. The F-Measure allows to emphasize either recall or precision by accordingly assigning the β parameter. For our evaluation, we equally weight precision and recall (β = 1). The resulting formula is called the F1 measure [20] [19].

In our case the performance measure it was calculated for 10 random querying cases in the model. For calculating performance measure for recommendation using SQL semantic search approach, we followed the below rules:

- As a relevant recommendations are classified recommenders which has number of GitHub commits more than 5 and semantic search score more than 0.6
- As an expected recommendations are classified recommenders which has number of GitHub commits more than 5
- A total a total number of recommendations are selected all recommenders provided from the model

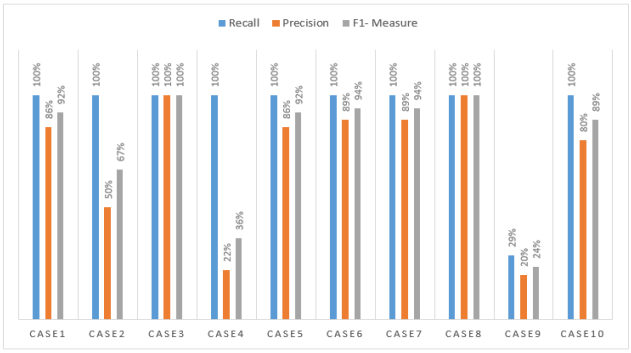

Fig. 2. Performance measure for recommendation using Semantic Search

Average values of the performance measurements for data in the Fig.2 above are: Recall = 0.93, Precision = 0.72 and F1-measure = 0.79

B. Subjective Evaluation

The evaluation process of the RSSE simulates the usage of the tool by a human user. Based on this reason for evaluation of the model we thought to hear the voice of the user developer. We chose 10 experienced developer and ask from them to evaluate the same cases which we calculated performance measurements. Developers had evaluated cases by answering a questionnaire with five question about the relevance of the recommendations for each of the cases:

- Not relevant,
- Poor relevant,
- Moderate,
- Satisfactory and
- Very Satisfactory.

Then we calculated performance measurement (precision, recall and F1-Measure) for each evaluated cases by developers.

Fig.3. Performance measure for subjective evaluation of recommendation using Semantic Search

Average values of the performance measurements for data in the Fig.3 above are: Recall = 0.99, Precision = 0.96 and F1-measure= 0.97.

C. Statistical significance between model performance and subjective evaluation

A t-test is a type of statistical test that is used to compare the means of two groups. It is one of the most widely used statistical hypothesis tests in pain studies [21]. *t-Tests* can be divided into two types: a). the independent t-test, which can be used when the two groups under comparison are independent of each other, and b). the paired t-test, which can be used when the two groups under comparison are dependent on each other. *t-Tests* are usually used in cases where the experimental subjects are divided into two independent groups [21].

The formula for the two-sample t-test is shown below.

$$t = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{\left(s^2\left(\frac{1}{n_1} + \frac{1}{n_2}\right)\right)}} \quad (2)$$

In this formula, t is the t-value, x_1 and x_2 are the means of the two groups being compared, s^2 is the pooled standard error of the two groups, and n_1 and n_2 are the number of observations in each of the groups. A larger t-value shows that the difference between group means is greater than the pooled standard error, indicating a more significant difference between the groups [21]. In our case we will calculate t-Test using type of t-Test independent groups: a). F1-Measure values of automatic performance measures of SQL Semantic Search recommendations and b). Subjective evaluation of SQL semantic search recommendations. Most statistical software includes a t-test function. This built-in function will take raw data and calculate the t-value. It will then compare it to the critical value, and calculate a p-value. In our case calculation of t-Test for three groups listed above will done using Excel data analysis tool. Results in the Fig.4 below present t- test calculation of *F1 Measure* between two these groups. Based on these results t value is -2.10 and p value is 0.06. Because of p-value is near 0.05, it means that can reject the null and assume that a significant difference exists between these groups and based on the *Mean* values, the second group performed significantly better with value 97.21 than the first group with mean value 78.88.

	Variable 1	*Variable 2*
Mean	78.88	97.21
Variance	753.99	10.40
Observations	10	10
Hypothesized Mean Difference	0	
df	9	
t Stat	-2.10	
P(T<=t) one-tail	0.03	
t Critical one-tail	1.83	
P(T<=t) two-tail	0.06	
t Critical two-tail	2.26	

Fig.4. t-Test calculation of the third group

V. CONCLUSION

In this paper it was presented an approach of generating recommendation for recommender systems in Software Engineering by using SQL Semantic search in a dataset compiled by cryptographic data in C#.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

The effectiveness of this approach is measured using two type the evaluations a) automatic performance measure performed by model and b) subjective evaluation performed by 10 experienced developer Based the results of the t-Test calculation the subjective evaluation performed better with mean value: 97.21, than automatic evaluation by the model mean value: 78.88, when t value was -2.10 and p value: 0.05 it was concluded that there are a significant difference between these groups.

REFERENCES

1. 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.
2. M. P. Robillard, W. Maalej, R. J. Walker and T. Zimmermann, Recommendation Systems in Software Engineering, Heidelberg New York Dordrecht London: Springer-Verlag, 2014.
3. 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.
4. D. Bruschi, L. Martignoni and M. Monga, "Detecting selfmutating malware using control-flow graph matching," International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, no. Springer, p. 129–143, 2006.
5. D.-K. Chae, J. Ha, S.-W. Kim, B. Kang and E. G. Im, "Software plagiarism detection: a graph-based approach," in 22nd ACM international conference on Conference on information & knowledge management, 2013.
6. X. Sun, Y. Zhongyang, Z. Xin, B. Mao and L. Xie, "Detecting code reuse in android applications using component-based control flow graph," in IFIP International Information Security Conference, 2014.
7. K. Tsiptsis and A. Chorianopoulos, Data Mining Techniques in CRM, United Kingdom: John Wiley and Sons, 2009.
8. J. Quinlan, "Induction of decision trees," Machine Learning, vol. 1, no. 1, p. 81–106, March 1986.
9. L. Rokach and O. Maimon, "Data Mining with Decision Trees: Theory and Applications," in World Scientific Publishing, 2008.
10. J. Zurada, " Introduction to artificial neural systems," in West Publishing Co., St. Paul, MN, USA, 1992.
11. H. Kaur, G. Singh and J. Minhas, "A Review of Machine Learning based Anomaly Detection Techniques," International Journal of Computer Applications Technology and Research, vol. 2, no. 2, pp. 185-187, 2013.
12. S. J. H. Ch and S. Bae, "Evolutionary Neural Networks for Anomaly," IEEE Transaction on Systems, Man, and Cybernetic, vol. 36, no. 3, 2005.
13. 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.
14. D. Lin, "An Information-Theoretic Definition of Similarity," Icml, vol. 98, pp. 296-304, 1998.
15. Z. Wu and M. Palmer, "Verb semantics and lexical selection," arXiv preprint cmp-lg/940603, 1994.
16. Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi and L. Cazzanti, "Similarity-based Classification: Concepts and Algorithms," Journal of Machine Learning Research, vol. 10, pp. 747-776, 2009.
17. R. Mistry and S. Misner, Introducing Microsoft® SQL Server 2012, Microsoft Press, 2012.
18. Microsoft, "Semantic Search," Microsoft, 17 08 2020. [Online]. [Accessed 24 01 2022].
19. M. Bruch, M. Monperrus and M. Mezini, "Learning from Examples to Improve Code Completion Systems," in International Symposium on the Foundations of Software, 2009.
20. C. J. V. Rijsbergen, "A non-classical logic for information retrieval," The computer journal, vol. 29, no. 6, pp. 481-485, 1986.
21. K. H. Yim, F. S. Nahm, K. A. Han and S. Y. Park, "Analysis of Statistical Methods and Errors in the Articles Published in the Korean Journal of Pain," Korean Journal of Pain, vol. 23, no. 1, p. 35–41, 2010.

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