Topic Modelling for Automatic Selection of Software Design Patterns

Abeer Hamdy
Faculty of Informatics and Computer Science
British University in Egypt
Electronics Research Institute
Egypt
Abeer.hamdy@bue.edu.eg

Mohamed Elsayed
Faculty of Informatics and Computer Science
British University in Egypt
Elshorouk city, Egypt
Mohamed.elsayd@bue.edu.eg

ABSTRACT
Design pattern is a high-quality and reusable solution to a recurring software design problem. It is considered an important concept in the software engineering field due to its ability to enhance some of the quality attributes of the software systems including maintainability and extensibility. However, novice developers need to be provided by a tool to assist them in selecting the fit design pattern to solve a design problem. The paper proposes a novel approach for the automatic selection of the fit design pattern. This approach is based on using Latent Dirichlet Allocation (LDA) topic model. The topic is a set of words that often appear together. LDA is able to relate words with similar meaning and to differentiate between uses of words with multiple meanings. In this paper LDA is used to analyze the textual descriptions of design patterns and extract the topics then discover the similarity between the target problem scenario and the collection of patterns using Improved Sqrt-Cosine similarity measure (ISCS). The proposed approach was evaluated using Gang of four design patterns. The experimental results showed that the proposed approach outperforms approach based on the traditional vector space model of Unigrams.

CCS Concepts
• Software and its engineering~Search-based software engineering

Keywords
Design pattern selection; DP recommendation; Gang of four, text mining, Information retrieval, topic modelling, LDA and vector space model.

1. INTRODUCTION
Software design is the most challenging activity in the software development life cycle. Design patterns are standardized and well documented best practices used by experienced software developers. Using patterns in software development leads to increasing the software reusability, quality and maintainability, in addition to reducing the technical risk of the project by not having to develop and test a new design [5], [15].

However, The existing of a large number of design patterns makes the selection of a fit design pattern for a given design problem a difficult task to the experienced developer, and makes it a challenging task for the inexperienced one who is not familiar with design patterns. To overcome this difficulty a supporting tool that automatically suggest to the developer the right design pattern for a given design problem during the design phase becomes a necessity.
Recently, a number of research studies were conducted to address this issue. Some of these studies developed techniques for suggesting the suitable pattern based on the UML design diagrams [3], [4]. Other techniques are based on question-answer [6], [12]. Some studies used text classification and text retrieval techniques [14], [22], [26], [27]. Others recommended design patterns based on anti-patterns detected in the design documents or the code [16], [23]. Some studies used Case Based reasoning CBR technique. Where, the fit design pattern is selected according to the previous experiences of pattern usages stored in a knowledge base in the form of cases [18], [29].
The contribution of this paper is proposing a novel approach based on Topic Model and Improved Sqrt-Cosine similarity measure (ISCS) for automating the process of selecting the fit design pattern (from a repository of patterns) to solve a given design problem. The motivation for this approach is the following:
1. The ability of topic models to analyze a collection of documents to find out the patterns of word-use and how to attach documents that share similar patterns.
2. Topic model where used to solve other similar problems in software engineering like mining bug report repositories for the purpose of automating bug triage [28]. Bug reports are short documents written in natural language same as design problem scenarios. Mining bug repositories is similar to mining a repository of design pattern descriptions.
The structure of the paper is as follows: Section 2 Explains Gang of four design patterns. Section 3 presents the literature survey in the field of design pattern selection. Section 4 explains the proposed approach and section 5 discusses the results. Finally, section 6 provides a conclusion and the possible future improvements.

2. DESIGN PATTERNS
The concept of design pattern was initiated in software development in 1994 when four software engineers published their book titled "Design patterns: Elements of reusable object oriented software" [5]. These authors are together popular with the title Gang of Four (GOF). GOF patterns are 23 patterns and categorized into three categories namely Creational, Structural and Behavioral patterns. GOF defined a template to describe the patterns. This template has

ICGDA '18, April 20–22, 2018, Prague, Czech Republic

ACM ISBN 978-1-4503-6445-4/18/04…$15.00

DOI: https://doi.org/10.1145/3220228.3220263

two counterparts which are the pattern's problem domain and the solution domain. The problem domain counterpart includes the intent of the pattern and the context where the pattern can be applied. While the solution domain includes mainly the class diagram that describes the static structure of the pattern, description to the consequences of applying the pattern and the antipatterns. Table 1 shows the description of the problem domain of the Class Adapter design pattern.

Table1. Problem domain description of Adapter design pattern

Intent:

Change the interface of a class into another interface. It let the classes work together without modifying their source code.

Applicability:

The Class Adapter pattern is used when:

- You want to reuse an existing class but its interface is not compatible with the interface you need.
- You have a class hierarchy and you need to use one or more subclasses. But you need to change their interfaces. It is impractical to subclass the subclasses to change their interface.
- You need to have classes with incompatible interfaces work together.

3. LITERATURE REVIEW

The following subsections summarize the literature review in the field of design pattern selection.

3.1. UML based approach

Kim and Khwand [3], Kim and Shen [4] generate a meta-model for each design pattern from their UML diagrams. However, this approach has two limitations which are: 1) the meta- models of some patterns will be similar as some patterns are similar in their structure but they have different intent for example State, Strategy patterns and Façade, Adapter patterns. 2) This approach is not scalable due to the overhead resulting from generating the meta-model; in addition to the more the number of patterns increases the more the similarity between the meta-models increases.

3.2. Question-Answer based Approach

In question-answer based approach the software developer is provided with some questions about the design problem. The most suitable patterns for this problem are recommended based on the designer answers [6], [12]. Palma et al. [6] constructed a Goal-Question-Metric model (GQM) from the question-answers to recommend patterns. In this GQM model, the defined goal is a pattern name. The system consists of two layers, the first layer has the conditions, where the second layer has the sub-conditions. The model evaluation has been done by a total of six graduate students along with two information technology professionals. The outcome of the evaluation resulted in a success ratio which reached 50%. While, Pavlie et al. [12] used the question-answers to build an ontology-based model for design patterns recommendations. However, constructing the questions in this approach is a challenge task especially with the large number of patterns. Furthermore, the set of questions are usually biased towards the specifics of the design patterns themselves rather than the software design problem.

3.3. Case Based Reasoning (CBR) approach

In CBR approach the fit design pattern is recommended based on previous experiences (cases) stored in a repository. Each case comprises two main parts which are: A description to the problem and the solution (fit design pattern). Gomes et sl. [18] built a repository of cases and retrieve the closest case from the repository for a user provided class diagram. While, Muangon and Intakosum [29] proposed a solution where both Case Based Reasoning (CBR) and Formal Concept Analysis (FCA) are integrated together forming a cohesive technique. This integration enabled the organization of indices to construct a complete design problem description which is used as aid to find more suitable design patterns. Both of the indices and case similarity are calculated using FCA. As argued by Gomes et al. [18], the core shortcoming of CBR based approach is the fact that its accuracy relies on both of the quality and diversity of the case repository.

3.4. Anti-patterns based approach

Nahar [16] identifies the anti patterns in the design diagrams then recommend the suitable design pattern. Smith and Smith and Plante [23] recommend patterns at the code-level, where patterns are recommended dynamically during the implementation phase. They identify anti-patterns using structural and behavioral matching in the code, and then the fit design patterns are recommended to overcome the identified anti-patterns. However, design pattern recommendation in the code development phase is too late as the software has already been designed and should be changed.

3.5. Text classification and retrival based approach

This approach is based on matching the design problem textual description against design pattern textual descriptions [14], [22],[26],[27]. Sanyawong et al. [14] developed classifiers to determine the design pattern category that fits the target problem. They used popular classification techniques: Naive Bayes, J48, k-NN, and SVM. They used 26 case studies for evaluation. Hasheminejad and Jalili also developed classifiers to determine the pattern category of the problem then apply cosine similarity to select a pattern from the predetermined category. Hussain et al. [27] used clustering (Fuzzy C-means) instead of classification in [26]. Suresh [22] proposed a framework for design pattern recommendation that depends on two approaches which are: text retrieval and question-answer. In this framework the problem is represented as a collection of unigrams and matched against the intent of each pattern. Then the intents of the top candidate design patterns are displayed for the designer to select the most suitable pattern. Nevertheless, they have partially implemented and tested their framework. In addition to measuring the similarity between the problem scenario and the pattern intent only will not produce a good performance.

However the work presented in [14], [22], [26], [27] represented the patterns and the problems as vectors of unigrams only and did not consider the semantic similarity, so the performances of their models will be sensitive to the words used in describing the problems. This is the reason we used topic model in this paper. Moreover, we used Improved sqrt-cosine similarity (ISCS) measure instead of the popular Cosine similarity.

4. METHODOLOGY

Our proposed approach is based on analyzing the corpus of pattern descriptions to extract the topics. Then transferring each design pattern description and each problem scenario to a vector of features. These features are unigrams and topics. The selected pattern is the one whose vector is the nearest to the problem vector. Figure 1 illustrates the steps of the proposed approach which starts with the textual preprocessing to the corpus of pattern descriptions followed by two parallel processes which are: 1) Topics extraction

through training an LDA model. Then a vector space model of topics (LDA VSM) is generated for the patterns. 2) Building a vector space model of unigrams (Unigram VSM). Both of the unigram and the LDA vector space models are concatenated such that each pattern will be represented by a vector of features includes unigrams and topics. The built LDA model and the unigram VSM are used in generating a vector of features for each target design problem scenario after the scenario preprocessed. Finally, the similarity between the feature vector of the problem scenario and each design pattern feature vector is computed and the selected pattern is the closest to the problem scenario. The following subsections discuss these procedures in more details.

4.1. Preprocessing

Each design pattern description and each design problem scenario is pre-processed through three activities which are: Tokenization then Noise Removal then Stemming. Tokenization process splits each document at the delimiters into unigrams (tokens). Then all the tokens are transferred into the lower case such that words like "Object", "object" and "OBJECT" are treated the same. Noise removal stage disregards the non- descriptive words like linking verbs and pronouns. These non-descriptive words are considered noise as they increase the size of the Vector Space Model and do not contribute to the retrieval process itself. Finally, the words are normalized to their root forms through Stemming. For example a stemmer can reduce each of the words "creating" and "created" to the word "create". Porter stemming algorithm [11] was adopted in our work.

4.2. Unigram VSM

In this stage each pattern is represented as a vector of unigrams. All the vectors have the same size which is equal to the number of unique words in the corpus of pattern descriptions. To build the Unigram VSM, all the unique words in the corpus of patterns are collected and each word is given an index. The pattern vector will have zeros in the cells that correspond to the words that do not exist in the pattern description and ones in the cells that correspond to the existing words.

In order to enhance the performance of retrieving the correct pattern and clean the noises of the corpus; a feature weighting scheme was adopted, where the weight of each unigram in the Unigram VSM reflects the relative importance of this word for a specific design pattern description within the corpus. We adopted TF*IDF weighting scheme [7] as it is one of the widely used schemes in the field of information retrieval. TF*IDF stands for Term Frequency-Inverse Document Frequency. Term frequency TF (t,d) measures how many times a term (t) occurs in a document (d). While Document frequency DF (t,D) measures how many documents in a corpus (D) the term (t) appears in. Inverse document frequency IDF (t,D) equals to the inverse of DF(t,D). Classical TF*IDF is computed by equation (1) as follows:

$$TF * IDF(t,d,D) = TF(t,d) \times IDF(t,D) \quad (1)$$

Where, $IDF(t,D) = 1/DF(t,D)$

TF*IDF value copes with the fact that the repetitive words in a document usually carry a high level of information to that document, and that the less frequent a term is mentioned in a corpus the higher its importance to the document in which it appears. However, TF*IDF computed using equation (1) does not take the document length into consideration. Also, TF value indicates that a term occurs five times in a document is five times valuable than if it occurs once in the same document, which is not true. So, other forms to compute the TF*IDF were recommended in the literature to make the TF*IDF values correspond to user intuitions of the relevance of each term. In this work equation (2) is used to compute TF*IDF.

$$TF * IDF(t,d,D) = Sqrt\big(TF(t,d)\big) * IDF(t,D) * 1/ Sqrt(length)) \quad (2)$$

Where, $IDF(t,D) = \log(N/(DF(t,D)+1))$

The document length could be disregarded in equation 2, as each of the design pattern descriptions and the problem scenarios are short documents.

4.3. Topic Model

Topic model Latent Dirichlet Allocation (LDA) is a probabilistic generative model that allows the discovering of topics in a collection of documents [2], [1]. Each topic includes a group of words, from the corpus, that may occur together. The words belong to the topic with different probabilities. Each document could be represented by a vector includes the probabilities of topic distributions over the document. We called this vector LDA vector. So, LDA vectors hold beneficial features to the semantic similarity between the documents. The number of topics and number of words per topic are parameters should be selected during the experiments.

In this work LDA model is trained and topics are extracted from the corpus of design patterns descriptions. Then, LDA VSM is created where each pattern is represented by a vector includes the distribution of the topics over the description of this pattern. The size of the vectors is equal to the number of topics.

The topic distributions over the target design problem is generated (design problem LDA vector) using the previously trained LDA model. Further a vector of unigrams for the target problem is generated based on the previously built Unigram VSM. Finally, LDA vectors are concatenated with the Unigrams vectors for each design pattern and each design problem.

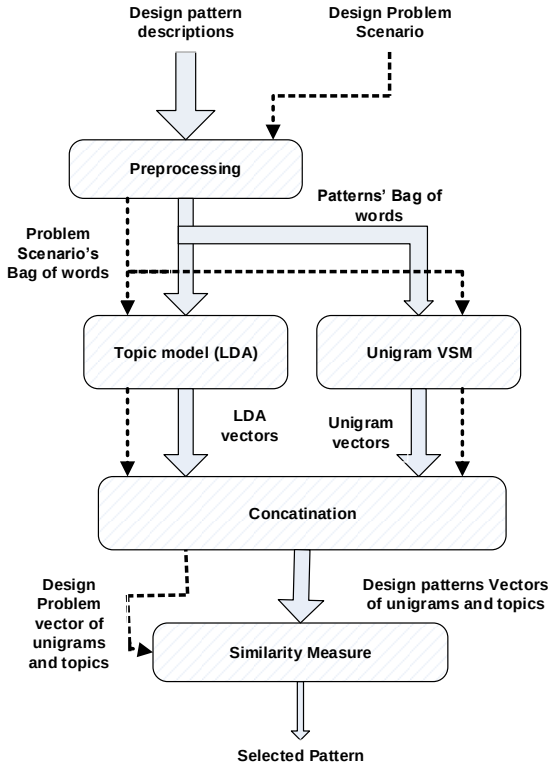

Figure 1. Proposed framework for automatic selection of design pattern

4.4. Similarity measure

The selected pattern for a given design problem is the pattern whose vector of unigrams and topics is the closest to the problem's vector. Cosine similarity (CS) is one of the popular measures in text mining as it measures the angle between the vectors, is given by equation 3.

$$CS(V,W) = \frac{\sum_{i=1}^{n} v_i w_i}{\sqrt{\sum_{i=1}^{n} v_i^2}\sqrt{\sum_{i=1}^{n} w_i^2}} \quad (3)$$

Where, V is the feature vector of one of the patterns, W is the feature vector of the given design problem and n is the size of the vectors.

However, Cosine similarity is derived from Euclidian distance which is not effective in text mining applications. So, Zhu et al. [25] proposed another similarity measure for information retrieval, called Sqrt-Cosine similarity (SCS). SCS is derived from Hellinger distance which is meant to measure the distance between two probabilities. They conducted text clustering experiments and proved that using Sqrt-Cosine similarity resulted in better performance than using Cosine similarity. Hellinger distance and SCS are given by equations 4 and 5. Sohangir and Wang [24] proposed a modified version of SCS called it Improved Sqrt-Cosine similarity (ISCS); is given by equation 6. They conducted experiments to compare the impact of using CS, SCS, and ISCS on the performance of each of the text classification and text clustering techniques. It was found that ISCS is the superior similarity measure, than CS then SCS. In this work we used both of Hellinger distance and ISCS.

$$SCS(v,w) = \frac{\sum_{i=1}^{n} \sqrt{v_i w_i}}{(\sum_{i=1}^{n} v_i)(\sum_{i=1}^{n} w_i)} \quad (4)$$

$$d_{Hellinger} = \sqrt{2 - 2\sum_{i=1}^{n} \sqrt{v_i w_i}} \quad (5)$$

$$ISCS(v,w) = \frac{\sum_{i=1}^{n} \sqrt{v_i w_i}}{\sqrt{\sum_{i=1}^{n} v_i}\sqrt{\sum_{i=1}^{n} w_i}} \quad (6)$$

4.5. Evaluation metric

In this paper, the Precision metric is used to assess the proposed approach and is defined by equation (7) as follows:

$$Precision = \frac{TP}{\mathrm{TP+FP}} \quad (7)$$

Where, TP is the number of correctly recommended design patterns while FP is the number of incorrectly recommended patterns.

5. RESULTS AND DISCUSSION

To evaluate the effectiveness of our approach, two corpus were created one of them includes the textual descriptions of the GOF design patterns. Each pattern document includes the intent, applicability in addition to part of the solution domain (participants, and collaborators). GOF book in addition to Wikipedia.com were used to prepare a rich description document to each pattern includes the pattern distinctive words. The other corpus includes 29 real design problem scenarios collected from various sources including various design patterns books, Wikipedia.com and Sourceforge.com. We label each design problem with the fit pattern manually. We meant to have some design problems written briefly or poorly to test the robustness of our approach. Seven samples of these design problems are defined as follows:

Problem #1: The Company class is the main class that encapsulates several important features related to the system as a whole. We need to ensure that only one instance of this class is present.

Problem #2: The system should has only one printer spooler although the system can identify many printers

Problem#3: The system has an interface named "MediaPlayer". This interface is implemented by a concrete class AudioPlayer. AudioPlayer has methods that play mp3 format audio files. There is another interface AdvancedMediaPlayer which is implemented by a concrete class AdvancedAudioPlayer to play vlc amd mp4 format files. It is required to have AudioPlayer class to use AdvancedaudioPlayer class to be able to play other formats.

Problem#4: It is required to use an existing user interface toolkit in developing software applications that work on different platforms. So it is important to include a portable window abstraction in the toolkit such that the user can create a window without being committed to certain implementation as the window implementation is related to the application platform.

Problem#5: The system approves purchasing requests. There are four approval authorities. The selection of the approval authority depends on the purchase amount. If the amount of the purchase is higher than one million dollar, the owner who approves. If it ranges from 500k to less than one million the CEO who approves, if it ranges from 25k to less than 500k the head of department approves, if less than 25k the vice who approves. The system needs to be flexible such that the approval authority for each amount of money can change at run time.

Problem #6: A menu consists of a set of choices and a mechanism for a user to specify which choice they want. There are a variety of styles of menus. One menu style is to print a number in front of each string (e.g., 1, 2, and so on) and let the user enter a number to make a choice. Another menu style prints a letter in front of each string (e.g., A, B, and so on) and lets the user enter a letter to make a choice. Still another menu style prints each string out, and lets the user type in the first few characters of the string to make that choice. In general, all of the menus must provide a way to add entries to the menu, delete entries, display the menu, and obtain the user's choice. It should be extremely easy for us to modify the program so that it uses a different menu style.

Problem # 7: The developer of a game desires the player to be able to pick up and drop off a variety of elements exist in the environment of the game. Two types of these elements are bags and boxes, each of them may contain individual elements as well as other bags and boxes.

Pre-processing was performed using the natural language toolkit NLTK [17]. While, Genism [8] was used for training the LDA topic model. Experiments were tried with number of topics set equal to 5, 10 and 20. Number of terms per topic was set equal to the number of topics. The best precision obtained during experiments is about 72 % when setting the number of topics equal to 10. Table 2 depicts samples of the results. It shows for each problem scenario listed above both of the correct and the first three recommended design patterns using our approach. Design problem scenarios of failed cases were reviewed and it was noted that these cases do not include descriptive words of the pattern or they are not well written. Although problem #2 is written briefly, the proposed approach was able to select the right pattern. Although we labeled problem#6 with Strategy pattern (correct pattern), the first recommended pattern (Adapter) somehow fits for this problem also.

Table 2. First three selected patterns and similarity values for each of the 7 design problems listed.

ID	Correct Pattern	1st Selected Pattern	2nd Selected Pattern	3rd Selected Pattern
1	Singleton	Singleton (0.258)	Prototype (0.184)	Adapter (0.162)
2	Singleton	Singleton (0.186)	Visitor (0.067)	Strategy (0.062)
3	Adapter	Adapter (0.407)	Bridge (0.307)	Composite (0.216)
4	Bridge	Bridge (0.422)	Facade (0.230)	Template (0.166)
5	Chain of Responsibility	Chain O.R. (0.380)	Observer (0.191)	State (0.185)
6	Strategy	Adapter (0.15)	Visitor (0.139)	State (0.137)
7	Composite	Composite (0.011)	Bridge (0.007)	Visitor (0.005)

Two extra experiments were conducted in the first experiment both of the design patterns and the problem scenarios are represented using vectors of unigrams only. While, in the second experiment patterns and problems are represented using vectors of topics only. ISCS was used in the first experiment while Hellinger distance was used in the second experiment. It was found that our proposed approach is superior to the other two approaches in terms of precision as illustrated by table 3.

TABLE 3. Comparison between the precision of our proposed approach, Unigrams only approach and Topics only approach

Approach	Similarity measure	Precision
Proposed (Topics and Unigrams)	ISCS	72%
Topics only	Hellinger distance	36%
Unigrams only	ISCS	60%

6. CONCLUSION AND FUTURE WORK

This paper proposed a novel methodology that automatically selects the fit design pattern to solve a given design problem. The problem scenario is expressed in natural language. For this purpose a vector space model was constructed using unigram features. LDA model was trained and topics were extracted from the textual description of the design patterns. This LDA model was used to generate LDA vector for each pattern and the target problem scenario. Each LDA vector represents the probability distribution of the topics over a pattern or a problem scenario. The LDA vectors are concatenated with the unigram vectors. Then, Improved Sqrt-cosine similarity measure was used to retrieve the fit pattern from the repository of patterns. To test our approach we had to build two repositories, one for the design pattern descriptions and the other for the design problems as there is no universal dataset, researchers can use. The experimental results illustrated the high efficiency of our proposed approach in terms of precision comparing to the traditional unigram vector space model used in information retrival.

Currently, we have been working on extending our dataset to include more catalogues of patterns which are patterns of concurrency and security. In addition to, building a repository to the design problem scenarios so we can integrate case based reasoning approach with our proposed approach.

7. REFERENCES

[1] C. Chemudugunta, P.S.M. Steyvers, "Modeling general and specific aspects of documents with a probabilistic topic model", Proceedings of the 20th Annual Conference on Neural Information Processing Systems, 2007 .

[2] D.M. Blei, J.D. Lafferty, A correlated topic model of science. Ann. Appl. Stat. pp.17–35,2007.

[3] D.K. Kim, C.E. Khawand, "An approach to precisely specifying the problem domain of design patterns." Journal of Visual Languages and Computing ,Elsevier 18, pp.560–591,2007.

[4] D.K. Kim, W. Shen, "Evaluating pattern conformance of UML models: a divide and conquer approach and case studies", Softw. Q. J. 16 (3) pp.329–359,2008.

[5] E. Gamma, R. Helm, R. Johnson, J. Vlissides, "Design Pattern: Elements of Reusable Object-Oriented Software." Addison-Wesley, 1995.

[6] F. Palma, H. Farzin, Y.-G. Gu´eh ,eneuc, and N. Moha, "Recommendation System for Design Patterns in Software Development: An DPR Overview," 3rd International Workshop on Recommendation Systems for Software Engineering, IEEE, pp. 1–5, 2012.

[7] F. Sebastiani, "Machine learning in automated text categorization." Journal of ACM Computing Survey (CSUR) 34,pp. 1–47, 2002.

[8] Gensim: https://radimrehurek.com/gensim/

[9] H. Kampffmeyer and S. Zschaler, "Finding the Pattern You Need: The Design Pattern Intent Ontology," Model Driven Engineering Languages and Systems, Springer Berlin Heidelberg , vol. 4735, pp. 211–225, 2007.

[10] I. Issaoui, N. Bouassida, and H. Ben –Abdallah, "A New Approach for Interactive Design Pattern Recommendation," Lecture Notes on Software Engineering, Vol. 3, No. 3, August 2015.

[11] K. S. Jones and P. Willet, "Readings in Information Retrieval," San Francisco: Morgan Kaufmann, 1997.

[12] L. Pavlic, V. Podgorelec, and M. Hericko, "A Question-based Design Pattern Advisement Approach,"Computer Science and Information Systems , vol. 11, no. 2, pp. 645–664, 2014.

[13] M. Melucci,"Vector-space model", Encyclopedia on Database Systems pp. 3259-3263,2009.

[14] N. Sanyawong, E. Nantajeewarawat, "Design Pattern Recommendation: A Text Classification Approach". 6th International Conference on Information and Communication Technology for Embedded Systems, IC-ICTES, 2015.

[15] N.L. Hsueh, J.-Y. Kuo, C.C. Lin, "Object-Oriented design: a goal-driven and pattern-based approach," J. Softw. Syst. Model. 8 (1), pp.1–18,2007.

[16] N. Nahar and K. Sakib, "ACDPR: A Recommendation System for the Creational Design Patterns Using Anti-patterns," IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Suita, pp. 4-7,2016.

[17] NTLK http://www.nltk.org

[18] P. Gomes, F. C. Pereira, P. Paiva, N. Seco, P. Carreiro, J. L. Ferreira, and C. Bento, "Using CBR for Automation of Software Design Patterns," Advances in Case-Based Reasoning, Springer Berlin Heidelberg , vol.2416, pp. 534–548, 2002.

[19] P. Castells, M. Fernandez, D. Vallet , "An adaptation of the vector-space model for ontology-based information retrieval", IEEE Trans. Knowl. Data Eng. 19 (2), pp.261–272,2007 .

[20] P. Tonella, G. Antoniol, "Object Oriented Design Pattern Inference." In: Proceedings of the IEEE International Conference on Software Maintenance, pp. 230–238 ,1999.

[21] R. M. Alguliyev, R. M. Aliguliyev, N. R. Isazade "An unsupervised approach to generating generic summaries of

documents, " Applied Soft Computing Volume 34, pp. 236-25, 2015.

[22] S. Suresh, M. Naidu, S. A. Kiran, and P. Tathawade, “Design Pattern Recommendation System: a Methodology, Data Model and Algorithms,” in Proceedings of the International Conference on Computational Techniques and Artificial Intelligence (ICCTAI) , 2011.

[23] S. Smith and D. R. Plante, “Dynamically Recommending Design Patterns,” in Proceedings of the 24th International Conference on Software Engineering and Knowledge Engineering (SEKE) , pp. 499–504, 2012.

[24] S. Sohangir and D. Wang, “Improved sqrt‐cosine similarity measurement”, Journal of Big Data , Springer, 2017.

[25] S. Zhu , Lizhao Liu, Yan Wang, Information Retrieval using Hellinger Distance and Sqrt-cos Similarity, The 7th International Conference on Computer Science & Education (ICCSE 2012) July 14-17, 2012. Melbourne, Australia

[26] S. M. H. Hasheminejad, S. Jalili, “Design patterns selection: An automatic two-phase method”, Journal of systems and software, elsevier, pp. 408-424, 2012.

[27] S. Hussain, , J. Keung, A. A. Khan, “Software design patterns classification and selection using text categorization approach” Applied soft computing, pp. 225-244, 2017.

[28] T. Zhang, J. Chen, G. Yang, B. Lee, X. Luo , “Towards more accurate severity prediction and fixer recommendation of software bugs”, Journal of systems and software, elsevier,2016.

W. Muangon and S. Intakosum, “Case-based Reasoning for Design Patterns Searching System,” International Journal of Computer Applications , vol. 70, no. 26, pp. 16–24, 2013.