



Run Run Shaw Library

香港城市大學
City University of Hong Kong

Copyright Warning

Use of this thesis/dissertation/project is for the purpose of private study or scholarly research only. ***Users must comply with the Copyright Ordinance.***

Anyone who consults this thesis/dissertation/project is understood to recognise that its copyright rests with its author and that no part of it may be reproduced without the author's prior written consent.

SOFTWARE DESIGN PATTERNS
CLASSIFICATION AND SELECTION
USING TEXT CATEGORIZATION
APPROACH

SHAHID HUSSAIN

DOCTOR OF PHILOSOPHY
CITY UNIVERSITY OF HONG KONG
SEPTEMBER 2017

CITY UNIVERSITY OF HONG KONG

香港城市大学

Software Design Patterns Classification and
Selection using Text Categorization Approach

基於文本分類方法的軟件設計模式
分類與選擇

Submitted to
Department of Computer Science
電腦科學系

In Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
哲學博士學位

by

Shahid Hussain
沙希德·侯赛因

September 2017
2017 年 9 月

List of Publications

Published and Accepted Journal Articles

1. **S. Hussain**, J. Keung, and A. A. Khan, "Software design patterns classification and selection using text categorization approach," **Applied Soft Computing**, vol. 58, 2017.
2. **S. Hussain**, J. Keung, A. A. Khan, A. Akhunzada, A. Ahmad, J. Geon, S. Coumo, "Implications of Deep Learning for the Automation of Design Patterns Organization," *Journal of Parallel and Distributed Computing*, 2017.

Under Review Journal Articles

1. **S. Hussain**, J. Keung, and A. A. Khan, "Automated Framework for the Classification and Selection of Software Design Patterns," *Journal of Data and Knowledge Engineering*, Jun 2017.
2. **S. Hussain**, J. Keung, and A. A. Khan, "Impact of Pattern and Anti-Patterns Correlation on the Software Quality," *Software Quality Journal*, Jun 2017.
3. **S. Hussain**, J. Keung, and A. A. Khan, A. Akhunzada, R. Choo, "Towards Performance Enhancement of Text Classification based automated system," *New Generation Computing*, (Review since May 2017)

Book Chapter

1. **S. Hussain**, J. Keung, and A. A. Khan, "A framework for ranking of software design patterns," *Proceedings of 11th International Conference on Complex, Intelligent, and Software Intensive Systems, CISIS IEEE/Springer*, 10-12 July 2017 Torino, Italy.

Conference Proceedings

1. **S. Hussain**, J. Keung, and A. A. Khan, "Empirical Investigation of Correlation between Gang-of-Four Design Patterns Usage and Structural Complexity" *Proceedings of 24th Asia-Pacific Software Engineering Conference (APSEC)*, Dec 2017 (Accepted)
2. **S. Hussain**, J. Keung, and A. A. Khan, "The effect of Gang-of-Four design patterns usage on design quality attributes," *Proceedings of the IEEE International Conference on Software Quality, Reliability and Security (QRS)*, 27-29 July 2017.
3. **S. Hussain**, "A methodology to predict the instable classes," *Proceedings of the 32nd ACM Symposium on Applied Computing (SAC)*, Morocco, 4th to 6th April 2017
4. **S. Hussain**, J. Keung, and A. A. Khan, and K. E. Bennin, "Detection of fault-Prone classes using logistic regression based object-oriented metrics thresholds" *Proceedings of the IEEE International Conference on Software Quality, Reliability and Security (QRS2016)*, Austria, 2016.
5. **S. Hussain**, J. Keung, and A. A. Khan, and K. E. Bennin, "A methodology for the selection of design patterns," *Proceedings of Computer Software and Applications Conference (COMPSAC)*, 2016.
6. **S. Hussain**, A. A. Khan, and K. E. Bennin, "Empirical investigation of fault predictors in context of class membership probability estimation", *Proceedings of the ACM Symposium on Applied Computing (SAC)*, Pisa Italy, 4-8 April, 2016.

7. **S. Hussain**, “Threshold analysis of design metrics to detect design flaws,” Proceedings of the **ACM Symposium on Applied Computing (SAC)**, 4-8 April, 2016.
8. **S. Hussain**, J. Keung, A. A. Khan, and K. E. Bennin, “Performance evaluation of ensemble Methods: An experiment”, Proceedings of 24th Australasian Software Engineering Conference (**ASWEC**), Oct 2015.

Acknowledgement

It is my pleasure to acknowledge the roles of several individuals who were instrumental for completion of my PhD research.

First and foremost, I would like to express my gratitude to my PhD advisor **Dr. KEUNG, Wai Jacky**, who encouraged me on many occasions to pursue the proposed work and taught me the state of art followed by the software engineering research community. I truly enjoyed the working research environment under supervision of respected Dr. KEUNG, Wai Jacky, that stimulates my original thinking and initiatives. Indeed, Dr. KEUNG, Wai Jacky's skillful guidance, innovative ideas, and stoic patient are greatly appreciated.

I would like to acknowledge the my Ph.D qualifying panel members **Dr. WANG Jianping** and **Dr. LU Zheng**, whose valuable suggestions and advices aid me to complete my research work.

I would like to say a special thanks to my colleagues who worked with me in collaboration and help me to handle the stress situations. Their company made my PhD life as a memorable experience in my future life.

Subsequently, I would also like to say a special thank to my respected teachers, **Prof KWONG, Tak Wu Sam** (HoD), **Dr WONG, Hau San Raymond**, **Dr CHAN, Wing Kwong Ricky**, **Dr LI, Zhenjiang**, **Dr XU, Hong Henry** , **Dr NUTANONG, Sarana** , **Dr WONG, Tsui Fong Helena**, and **Dr YUEN, Chun Hung Joe**, whose company play as a mentor role to groom my teaching and learning experience.

I would also acknowledge the efforts of all my friends who support me in certain dimensions and made my PhD life as a memorable life time.

Finally, I would like to say a very special thanks to my sweet mother, my beloved wife, my pretty sons **Mohammad Zoran** and **Mohammad Azlan Aahil**, my supportive brothers **Zahid Ul Hassan** and **Mohammad Imran Faisal**, and my pretty nephew **Ain Ali**, whose moral support help me to complete my PhD. I have no words to express their meticulous love and support.

Table of Contents

1 List of Publications	I
2 Acknowledgement	III
3 Table of Contents	IV
4 List of Tables	VIII
5 List of Figures	X
6 Acronym and Abbreviation	XIII
7 Dedication	XIV
8 Abstract	XV
CHAPTER 1: INTRODUCTION	
1.1 Introduction	1
1.2 Problem Statement	4
1.3 Illustrations of Problem Statement	5
1.3.1 Illustration-1	5
1.3.2 Illustration-2	6
1.3.3 Illustration-3	6
1.3.4 Illustration-4	7
1.4 Research Objective and Research Questions	12
1.5 Motivation	12
1.6 Introduction of Software Design Patterns	13
1.7 Online Software Design Pattern Repositories	15
1.8 Widely-known Design Pattern Collections	15
1.8.1 Gang-of-Four Design Patterns	16
1.8.2 Douglass Design Patterns	16
1.8.3 Security Design Patterns	16
1.8.4 Dwyer Design Patterns	16
1.9 Text Categorization Approach	17
1.9.1 Preprocessing Activities	17
1.9.1.1 Removal of Stopword	18
1.9.1.2 Words Stemming	18
1.9.1.3 Indexing	18
1.9.1.4 Weighting Methods	18
1.9.1.5 Feature Selection Techniques	20
1.9.2 Text Classification Techniques	23
1.9.2.1 Naïve Bayes	23
1.9.2.2 C4.5 Decision Tree	24

1.9.2.3 Support Vector Machines (SVM)	24
1.9.3 Text Clustering Techniques	25
1.9.3.1 Hierarchical Clustering Algorithms	25
1.9.3.2 K-means	25
1.9.3.3 Self Organizing Map (SOM)	26
1.9.3.4 EM-Algorithm (Model Based Clustering)	26
1.8.3.5 Fuzzy C-means	27
1.10 Performance Measures for the Evaluation of Automated Systems based on Text Categorization approach	28
CHAPTER 2: LITERATURE REVIEW	
2.1 Literature Review	32
2.2 Design Pattern Classification	34
2.2.1 Problem based Classification of Design Patterns	35
2.2.2 Solution based Classification of Design Patterns	35
2.3 Design Pattern Selection	40
2.3.1 UML based Automated Systems	41
2.3.2 Ontology based Automated Systems	44
2.3.3 Text Categorization based Automated Systems	46
2.4 Text Categorization And Applications	47
2.4.1 Tokenization	48
2.4.2 Construction of Feature Space with unique Words	48
2.4.3 Reducing Feature Space	49
CHAPTER 3: PROPOSE APPROACH	
3.1 Overview of the Proposed Approach	51
3.1.1 Preprocessing	52
3.1.1.1 Ensemble-IG (Proposed feature selection method)	57
3.1.2 Unsupervised learning for Design Patterns	58
3.1.3 Identification of Design Pattern Class	59
3.1.4 Design Patterns Suggestion and Selection	60
CHAPTER 4: EVALUATION MODEL	
4.1 Evaluation Model for the Proposed Approach	65
4.1.1 Evaluation of Unsupervised Learning Techniques	66
4.1.1.1 Phase-1	66
4.1.1.2 Phase-2	67
4.1.1.3 Phase-3	68
4.1.2 Evaluation of Selection of Design Pattern(s) Process	69
4.2 Design Pattern Groups	69

4.2.1 Gang-of-Four (GoF) Design Pattern Collection	70
4.2.2 Douglass Design Pattern Collection	71
4.2.3 Security Design Pattern Collection	72
4.2.4 Duyne Design Pattern Collection	75
4.3 Design Problems	79
CHAPTER 5: EXPERIMENT ANALYSIS	
5.1 Experiment Analysis	80
5.2 Pseudocode to describe the implementation procedures	80
5.2.1 Pseudocode for the organization of design patterns	80
5.2.2 Pseudocode for the identification of appropriate pattern class fro the given design problems	82
5.2.3 Pseudocode for the selection of design patterns for the given design problems	83
5.3 Detailed Analysis of implementing tasks described the Pseudocodes	84
5.3.1 Experimental Procedure for the selection of best weighting method (Experiment 1)	84
5.3.2 Experimental Procedure for the Selection of Outperform Unsupervised Learning Technique in Terms of organization of design patterns (Experiment 2)	86
5.3.3 Experimental Procedure to determine the effectiveness of unsupervised learner for pattern organizations with respect to the classification scheme of experts. (Experiment 3)	87
5.3.4 Experimental Procedure for the Selection of best feature selection method (Experiment 4)	90
5.3.5 Experimental Procedure for the Selection of Outperform Unsupervised Learning Technique in terms of identification of pattern class for the sample of design problems (Experiment 5)	90
5.3.6 Experimental Procedure for the Selection of Outperform Unsupervised Learning Technique in Terms of predicting appropriate design pattern for the sample of design problems P (Experiment 6)	90
5.3.7 Experimental Procedure for the Selection of best similarity option in terms of predicting appropriate design pattern for the sample of design problems P (Experiment 7)	93
CHAPTER 6: RESULTS AND DISCUSSION	
6.1 Results and Discussion	94
6.2 First Case Study (Gang-of-Four Design Patterns in the Domain of Object-Oriented Development)	96
6.2.1 Organization of Design Patterns	96
6.2.2 Determination of design pattern class for the sample of given design problems	99
6.2.3 Selection of design pattern for the sample of given design problems	101
6.3 Second Case Study (Schumacher et al Design Patterns in the Domain of Secuirty Systems)	104
6.3.1 Organization of Design Patterns	104
6.3.2 Determination of design pattern class for the sample of given design problems	107
6.3.3 Selection of design pattern for the sample of given design problems	109
6.4 Third Case Study (Douglass Design Patterns in the Domain of Real Time Application)	112

6.4.1 Organization of Design Patterns	112
6.4.2 Determination of design pattern class for the sample of given design problems	115
6.4.3 Selection of design pattern for the sample of given design problems	120
6.5 Fourth Case Study (Duyne Design Patterns in the Human Computer Interaction)	120
6.5.1 Organization of Design Patterns	122
6.5.2 Determination of design pattern class for the sample of given design problems	124
6.5.3 Selection of design pattern for the sample of given design problems	125
CHAPTER 7: IMPLICATION OF DEEP LEARNING IN PATTERN ORGANIZATION	
7.1 Implication of Deep Learning in different domains	128
7.2 Proposed Method	129
7.2.1 Preprocessing	129
7.2.2 Construction of Feature set	130
7.2.3 Text Classification	130
7.3 Experimental Setup	130
7.4 Results and Discussion	132
7.4.1 Organization of GoF Design Patterns	132
7.4.2 Organization of Douglass Design Patterns	134
7.4.3 Organization of Security Design Patterns	135
7.5 Consequences of Case Study	137
CHAPTER 8: IMPLICATIONS AND CONCLUSION	
8.1 Implications and Conclusion	139
APPENDIX	
Appendix A Templates of Design Patterns	144
Appendix B List and Description of Design Problems	151
Appendix C Cosine Similarity and Application of First Similarity Option	170
Appendix D Supervised Learning Techniques	179
REFERENCES	
1 References	180

List of Tables

Table 1.1	Issues in online repository L. Rising-Patterns Almanac 2000	6
Table 1.2	Canonical and Variants for the Design Problem-2 (DP-2)	8
Table 1.3	Generalization, Association, and Absratic Method Call Matrix for solution of pattern and variants of Design Problem 2 (DP-2)	9
Table 1.4	Generated ordered pairs via pattern clustering algorithm for the solution of pattern and variants of Design Problem 2 (DP-2)	11
Table 1.5	Summary of Approaches used for Problem-based Classification of Design Patterns	14
Table 2.1	Summary of Domains and related frequently used design pattern collections	33
Table 2.2	Summary of problem-based the design pattern collections and related information	36
Table 2.3	Summary of problem-based the design pattern collections and related information	42
Table 3.1	A sample of documents	53
Table 3.2	Overview of the Vector Space Model (VSM) for sample of documents	53
Table 3.3	Vector Space Model (VSM) with Binary weighting method for sample of documents	54
Table 3.4	Vector Space Model (VSM) with TF/IDF weighting method for sample of documents	55
Table 3.5	Vector Space Model (VSM) with TFC weighting method for sample of documents	55
Table 3.6	Vector Space Model (VSM) with LTC weighting method for sample of documents	56
Table 3.7	Vector Space Model (VSM) with Entropy weighting method for sample of documents	56
Table 3.8	Constructed features set with M ranked features (M=5) for the sample of the documents	57
Table 3.9	Similarity measure for sparse vectors	60
Table 3.10	Similarity measures with different length of documents	61
Table 3.11	The impact of similarity measures with magnitudes	62
Table 3.12	Sample of patterns and their closeness with a given design problem	64
Table 4.1	Resource information about design problems	79
Table 6.1	Fuzzy c-means (i.e. Outperform Unsupervised learner) recommended class membership values for GoF design problems of Sample 1	101
Table 6.2	Fuzzy c-means (i.e. Outperform Unsupervised learner) recommended class membership values for GoF design problems of Sample 2	102
Table 6.3	Computed ranges for Θ_1 and Θ_2 parameters for design problems of sample-1 in the context of the GoF design pattern collection	103
Table 6.4	Fuzzy c-means (i.e. Outperform Unsupervised learner) recommended class membership values for Douglass design problems of Sample 1	109
Table 6.5	Fuzzy c-means (i.e. Outperform Unsupervised learner) recommended class membership values for Douglass design problems of Sample 2	110
Table 6.6	Computed ranges for Θ_1 and Θ_2 parameters for design problems of sample-1 in the context of the Douglass design pattern collection	111

Table 6.7 Fuzzy c-means (i.e. Outperform Unsupervised learner) recommended class membership values for Security design problems of Sample 1	117
Table 6.8 Fuzzy c-means (i.e. Outperform Unsupervised learner) recommended class membership values for Security design problems of Sample 2	118
Table 6.9 Computed ranges for Θ_1 and Θ_2 parameters for design problems of sample-1 in the context of the Security design pattern collection	121
Table 6.10 Computed ranges for Θ_1 and Θ_2 parameters for design problems of sample-1 in the context of the Duyne design pattern collection	126
Table 7.1 Implication of Deep learning in different domains	128
Table 7.2 Performance evaluation of classifiers of different design pattern catalogs	133
Table 8.1 Performance of unsupervised learner with all weighting methods in terms of organization of design pattern collections under study	140
Table 8.2 Outperform learners for the organization of design patterns of target collections	141
Table 8.3 Effectiveness of Outperform learners to determine the right pattern class for the given samples of design problems	141
Table 8.4 Improvement in the performance of outperform unsupervised learner via the best and proposed feature selection methods (DF=Document Frequency, IG=Information Gain, Corr=Correlation, GR=Gain Ratio)	142
Table 8.5 Improvement in the performance of outperform unsupervised learner via the best and proposed feature selection methods.	143

List of Figures

Figure 1.1 Scenario to describe the employment of pattern for given design problems	2
Figure 1.2 Scenario to describe the employment of pattern for the given design problems	7
Figure 1.3 Set of Preprocessing Activities	17
Figure 2.1 Zimmer’s proposed solution-based technique to organize the design patterns	37
Figure 2.2 Step-1 of Clustering approach (a solution-based technique) to organize the design patterns	38
Figure 2.3 Step-2 of Clustering approach (a solution-based technique) to generate the order pair of design patterns	39
Figure 2.4 Step-3 of Clustering approach (a solution-based technique) to transform the order pair of original and variants of design patterns	39
Figure 3.1 Overview of the proposed Approach	52
Figure 5.1 Flow of activities performed in the Experiment-1	85
Figure 5.2 Flow of activities performed in the Experiment-2	86
Figure 5.3 Flow of activities performed in the Experiment-3	87
Figure 5.4 Flow of activities performed in the Experiment-4	88
Figure 5.5 Flow of activities performed in the Experiment-5	89
Figure 5.6 Flow of activities performed in the Experiment-6	91
Figure 5.7 Flow of activities performed in the Experiment-7	92
Figure 6.1 Working procedures of the proposed approach in presented case studies	95
Figure 6.2 Identification of best weighting method for unsupervised learners weighting methods in the context of the GoF design pattern collection	96
Figure 6.3 Identification of outperform unsupervised learner with best weighting method in the context of GoF design pattern collection	97
Figure 6.4 The performance of unsupervised learners with best weighting methods and K clusters in the context of GoF design pattern collection	98
Figure 6.5 The effect of feature selection methods on the performance of outperform unsupervised learner C-means with a best weighting method in the context of the GoF design pattern collection	99
Figure 6.6 Identification of outperform unsupervised learner with a best weighting method to determine pattern class for the a) Sample-1 and b) Sample-2 design problems in the context of the GoF design pattern collection	100
Figure 6.7 Parameters of similarity options for Sample-1 of 10 design problems (Related to GoF pattern collection) Mean RCD for similarity option using equation 3.6 b)	103

Mean RCD for similarity option using equation 3.7	
Figure 6.8 Identification of best weighting method for unsupervised learners weighting methods in the context of the Douglass design pattern collection	104
Figure 6.9 Identification of outperform unsupervised learner with best weighting method in the context of Douglass design pattern collection	105
Figure 6.10 The performance of unsupervised learners with best weighting methods and K clusters in the context of Douglass design pattern collection	106
Figure 6.11 The effect of feature selection methods on the performance of outperform unsupervised learner C-means with a best weighting method in the context of the Douglass design pattern collection	107
Figure 6.12 Identification of outperform unsupervised learner with a best weighting method to determine pattern class for the a) Sample-1 and b) Sample-2 design problems in the context of the Douglass design pattern collection	108
Figure 6.13 Parameters of similarity options for Sample-1 of 10 design problems (Related to Douglass pattern collection) a) Mean RCD for similarity option using equation 3.6 b) Mean RCD for similarity option using equation 3.7	111
Figure 6.14 Identification of best weighting method for unsupervised learners weighting methods in the context of Security design pattern collection	112
Figure 6.15 Identification of outperform unsupervised learner with best weighting method in the context of Douglass design pattern collection	113
Figure 6.16 The performance of unsupervised learners with best weighting methods and K clusters in the context of Security design pattern collection	114
Figure 6.17 The effect of feature selection methods on the performance of outperform unsupervised learner C-means with the best weighting method in the context of Security design pattern collection	115
Figure 6.18 Identification of outperform unsupervised learner with a best weighting method to determine a pattern class for the a) Sample-1 and b) Sample-2 design problems in the context of Security design pattern	116
Figure 6.19 Parameters of similarity options for Sample-1 of 12 design problems (Related to Security pattern collection) Mean RCD for similarity option using equation 3.6 b) Mean RCD for similarity option using equation 3.7	121
Figure 6.20 Identification of best weighting method for unsupervised learners weighting methods in the context of Duyne design pattern collection	122
Figure 6.21 Identification of outperform unsupervised learner with best weighting method in the context of Duyne design pattern collection	122
Figure 6.22 The performance of unsupervised learners with best weighting methods and K clusters in the context of Duyne design pattern collection	123

Figure 6.23	The effect of feature selection methods on the performance of outperform unsupervised learner PAM-Euclidean with best weighting method in the context of Duyne design pattern collection	124
Figure 6.24	Identification of outperform unsupervised learner with best weighting method to determine a pattern class for the a) Sample-1 and b) Sample-2 design problems in the context of Duyne design pattern collection	125
Figure 6.25	Parameters of similarity options for Sample-1 of 15 design problems (Related to Duyne pattern collection) a) Mean RCD for similarity option using equation 3.6 b) Mean RCD for similarity option using equation 3.	125
Figure 7.1	Overview of the Proposed Method	129
Figure 7.2	Set of preprocessing activities	130
Figure 7.3	Performance evaluation of classifiers for the weighting methods on the GoF Design pattern collection	132
Figure 7.4	Performance evaluation of classifiers with feature sets constructed through feature selection method and proposed approach GoF pattern collection	133
Figure 7.5	Performance evaluation of classifiers for the weighting methods on the Douglass Design pattern collection	134
Figure 7.6	Performance evaluation of classifiers with the feature sets constructed through feature selection method and proposed approach on Douglass pattern collection	135
Figure 7.7	Performance evaluation of classifiers for the weighting methods on the Security Design pattern collection	136
Figure 7.8	Performance evaluation of classifiers with the feature sets constructed through feature selection method and proposed approach on Security pattern collection	137

Acronyms And Abbreviations

GoF	Gang-of-Four
POSA	Pattern Oriented Software Architecture
SOA	Service Oriented Architecture
HCI	Human Computer Interaction
UML	Unified Modeling Language
CBR	Case Base Reasoning
ML	Machine Learning
BOW	Bag of Words
HMM	Hidden Markov Model
YASS	Yet Another Suffix Striper
VSM	Vector Space Model
TF	Term Frequency
TFC	Term Frequency Collection
TF-IDF	Term Frequency and Inverse Document Frequency
LTC	Length Term Collection
DF	Document Frequency
IG	Information Gain
MI	Mutual Information
GI	Gini Index
OR	Odds Ratio
DFS	Distinguishing Feature Selector
LSI	Latent Semantic Indexing
ICA	Independent Component Analysis
PLSI	Probabilistic Latent Semantic Indexing
LDA	Latent Dirichlet Allocation
NB	Naïve Bayes
SVM	Support Vector Machines
FCM	Fuzzy C-Means
MSE	Mean Square Error
RI	Random Index
ARI	Adjusted Random Index
AMI	Adjusted Mutual Information
NMI	Normalized Mutual Information

DEDICATION

In the name of Allah, the Most Gracious and the Most Merciful

This thesis is dedicated to my late father, Allah Wasaya and mother Mumtaz Farah, who taught me that the best kind of knowledge to have is that which is learned for its own sake. It is also dedicated to my beloved wife Hina Gul with my two little champs Mohammad Zoran and Mohammad Azlan Aahil, whose lovely and pretty faces and moral support always encourage me to complete my degree successfully.

Abstract

Software design is considered as a challenging task in agile software development life cycle, where the fundamental structure of software artifacts is highly provoked to its evolution with the passage of time by adding new features or modifying the existing functionality. Consequently, it affects the system level quality attributes such as reusability, maintainability, and understandability. There are two common practical approaches to discuss the declined quality of software systems. The first approach is software refactoring, which is considered as a precise way to improve the design quality of a system by perfecting its internal structure without any change in its external behavior, and the second approach is the employment of design patterns. The use of design patterns in the early life of a designing phase may aid to prevent later refactoring. A design pattern refers to a solution of a commonly occurring problem and is published by an expert in a specific domain, such as in the domain of object-oriented development Gamma et al., published a catalog of 23 design patterns named as Gang-of-Four (GoF) design patterns. The classification scheme and semantic correlation between patterns depends on the experience and knowledge of experts in the corresponding domain. Consequently, a novice developer needs enough knowledge and efforts to understand the classification scheme, the semantic correlation between patterns, and the consequences of each pattern.

Numerous software design patterns have been introduced either as a canonical (a term coined to describe the standard/original solution) or a variant (a term coined to describe an alternative solution) solution to solve a design problem. Books and online repositories/libraries are common sources of cataloged software design patterns. Due to steadily increase in the number of design patterns in the books, literature, and online repositories, it is hard to be aware of the published design patterns and to select the pattern to the real design problem at hand. In this regard, Unified Modeling Based (UML), Ontology, and Text categorization based approach have been introduced to automate the selection of right design pattern for the given design problem. However, the emergence of new patterns, inconsistency in classification schemes, heterogenous pattern description, semi-formal specification, multi-class problem, and an adequate sample size to make precise learning (for individual classifier training) are the main constraints to use the existing automatic techniques to find a candidate design pattern class and suggest more appropriate pattern(s).

To address these issues, we exploit a text categorization based approach via unsupervised learning techniques that targets to present a systematic way to group the similar design patterns and suggest the right design pattern(s) to developers related to the specification of a given design problem (***First Contribution***). The proposed approach is employed through five widely used unsupervised learning techniques, namely Fuzzy c-means, k-means,

Agglomerative, Partition Around Medoids (PAM) with Euclidean and Manhattan distance measures in the context of four design pattern collections used in different domains and 105 real design problems. Subsequently, We also propose an evaluation model to assess the effectiveness of the proposed approach in terms of the 1) organization of design patterns of a target pattern catalog according to the expert's classification scheme, 2) determination of design pattern class for a given design problem, and 3) selection of right design patterns for a given design problem . In the text categorization approach based automated system, the global filter-based feature selection rather than wrapper and embedded methods are used to decrease the undesirable effect of useless features and construct a more representative feature set. The constructed feature set is based on the capacity of global filter-based feature selection method which is biased to their discriminative power. We leverage an Improved Global Feature Section Scheme (IGFSS) to combine the discriminative power of two feature selection methods and propose a new feature selection method named Ensemble-IG (***Second Contribution***). The proposed Ensemble-IG combines the power of Odds Ratio (OR) and Information Gain (IG). Moreover, we also propose a new approach to construct a more representative feature set by leveraging a powerful deep learning algorithm named Deep Belief Network (DBN) (***Third Contribution***).

We observed that Fuzzy c-means and Partition Around Medoids (PAM) make a better results as compared to other unsupervised learning techniques used in the proposed approach. However, we also observed that the learning precision of unsupervised learners remains sensitive to the weighting and feature selection methods. In many cases, we observed that Term Frequency Collection (TFC) and Term Frequency Inverse Document Frequency (TFIDF) remain the best weighting methods for unsupervised learners. Subsequently, we also observed significant improvement in learning precision of the outperformed unsupervised learner of the proposed approach through Ensemble-IG and deep learning based feature selection method.

There are several consequences extracted from the experimental results and described in the presented case studies. We summarize results and concluded that the proposed approach has four advantages over the existing approaches. First, the semi-formal specification of design patterns is not required as a prerequisite because description of problem definition (unstructured form) rather than solutions section of design patterns is used. Second, the ground reality of class label assignment is not mandatory because the proposed approach is employed through unsupervised rather than supervised learning techniques. Third, the lack of classifier's training for each design pattern class. Fourth, an adequate sample size is not required to make precise learning.

CITY UNIVERSITY OF HONG KONG
Qualifying Panel and Examination Panel

Surname: HUSSAIN
First Name: Shahid
Degree: PhD
College/Department: Department of Computer Science

The Qualifying Panel of the above student is composed of:

Supervisor(s)

Dr. KEUNG Wai Jacky	Department of Computer Science City University of Hong Kong
---------------------	--

Qualifying Panel Member(s)

Dr. LU Zheng	Department of Computer Science City University of Hong Kong
--------------	--

Dr. WANG Jianping	Department of Computer Science City University of Hong Kong
-------------------	--

This thesis has been examined and approved by the following examiners:

Dr. YU Yuen Tak	Department of Computer Science City University of Hong Kong
-----------------	--

Prof. ZHANG Qingfu	Department of Computer Science City University of Hong Kong
--------------------	--

Dr. KEUNG Wai Jacky	Department of Computer Science City University of Hong Kong
---------------------	--

Dr. LAI Richard Y	Department of Computer Science and Information Technology La Trobe University
-------------------	---

1.1. Introduction

Software design is considered as a most challenging task in the development life cycle, where a designer plans how a system should meet the customer's requirements and could be more effective. A number of design methods (i.e. Software design process) can be realized during the life cycle of software design. For example, architecture, interface, components, data structure, algorithms, and component design are the common practices of a designer to manage a design issue. The architecture-based design methods are used to identify the sub-systems and manage the structure of the system, control flow between the sub-systems, and decomposition of modules. This group of design methods uses software architectural styles which are represented through components and their relationship and can be considered as coordination tools among the activities of the software development life cycle [1]. The main benefits of using software architectural styles are the reuse, interoperability, understandability and vocabulary of design components. Attribute Driven Design (ADD) is a common architecture-based design method based on an iterative process to apply different architectural strategies and patterns that fulfill the quality attributes. In the working procedure of ADD, a software developer selects an architectural style and chooses the desired quality attributes included in the software system. Subsequently, software developers use the description of design requirements related to selected architectural component and the employed software design patterns [2]. Since the introduction of the concept of patterns [3, 4], based on their experiences, software developers have encapsulated and suggested the proven solutions in the form of design patterns to satisfy the recurring design problems in different domains, such as object-oriented development [5-9], Fault-tolerance in Telecommunication Systems [10], communication software [11], real time systems [12], user interface interaction [13-14], security [15], interdisciplinary [16], and Pattern-Oriented Software Architecture (POSA) series of patterns in different domains [17-21]. Subsequently, Rising - Patterns Almanac 2000¹, euroPLOP², and Portland Pattern Repository³ are good sources of patterns retrieval from the different domain. The rapid increase in the number of patterns has been reported in the form of documented literature and online repositories, which make it hard to be aware of the design patterns and make the proper selection process as a hard problem [22, 23], supported through the following Sommerville's quote [24].

“Only experienced software engineers who have a deep knowledge of patterns can use them effectively. These developers can recognize generic situations where a pattern can be applied. Inexperienced programmers, even if they have read the pattern books, will always find it hard to decide whether they can reuse a pattern or need to develop a special-purpose solution.”

The experienced and inexperienced developers follow the same procedure to select and use the design pattern with respect to the description of design problem [22], which is described in Figure 1.1.

¹<http://www.smallmemory.com/almanac/>

²<http://europlop.net/content/repositories>

³<http://c2.com/ppr/titles.html>

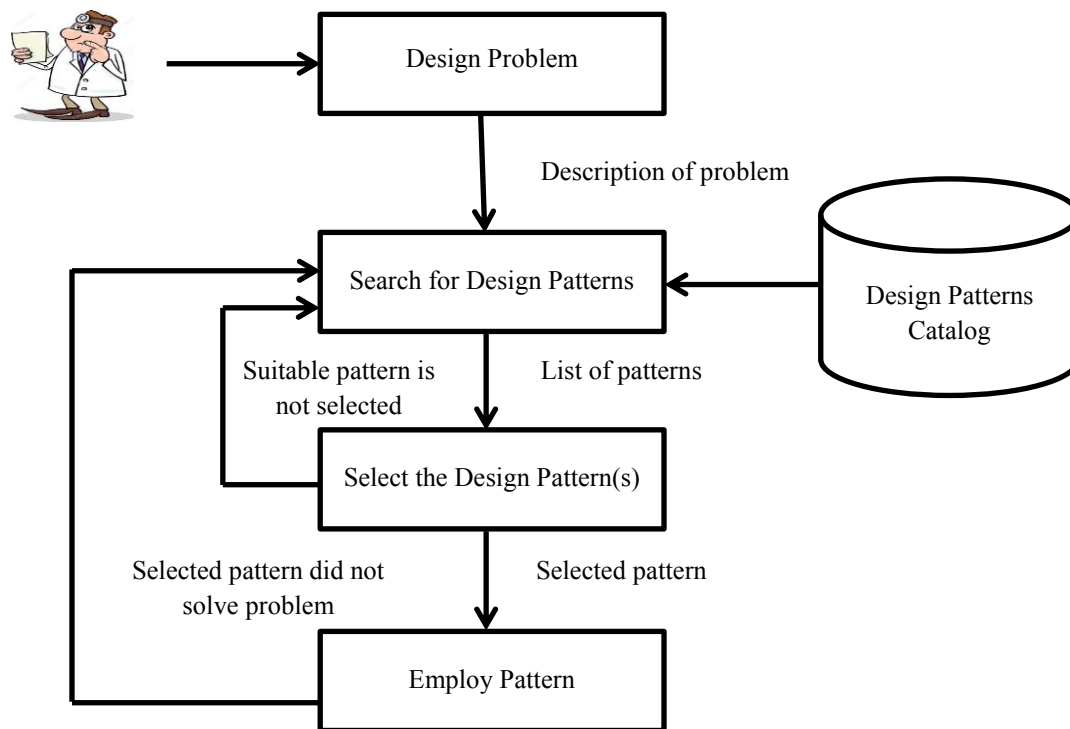


Figure 1.1. Scenario to describe the selection of patterns for given design problems

The searching of an appropriate pattern or group of patterns (i.e. Pattern Class or Pattern Category) is highly related to;

1. A well-defined classification scheme, which is introduced by developer/practitioners' experience, and
2. The documentation of frameworks and patterns which might be a time consuming factor for the developers and yield in the creation of many variants [9].

The existing efforts of developers in the domain of classification of design patterns can be categorized into two groups that are a) Problem-based classification, and b) Solution-based classification schemes. For example, in the domain of object-oriented design problems, Gamma et al. [5], Pree. [6], and Coad et al. [7] classified design patterns into groups with respect to design problem related to certain aspects such as structure, abstract coupling, behavior, aggregation, and interactions of methods and classes. Similarly, in the domain of real-time applications, the design issues with respect to architectural design, memory, safety and reliability, concurrency, resources, and distributions are considered to classify the patterns of real time application [12]. Recently, in an online survey entitled “Trends and Development of Object-Oriented Design Patterns”⁴, 75% of respondents (i.e. Developers) reported the use of problem context (rather than the purpose) when they look for a design pattern. The most common sources of design pattern searching are internet^{1,2,3}, literature [5-21], past experience, and through communication with colleagues or friends. The developer’s interest in the problem context is reported to find a pattern with some limitations of problem-based classification, such as.

⁴<http://www.my3q.com/research/cghmichelle/37053.phtml>

Issue-1. Enough knowledge is required to understand the classification schemes and to find the right patterns. For example, in the case of Gang-of-Four (GoF) (patterns are grouped into 3 categories) and Douglass's (patterns are grouped into 5 categories) pattern collections, a novice developer can easily understand the nomenclature of classification scheme due to less number of pattern classes. However, in case of Booch [25], interdisciplinary [16], and the spoiled pattern [9], it becomes difficult to understand the classification scheme, and long description of documented patterns [22, 23, 26].

Issue-2. The problem domain of design patterns is described in unstructured form related to developers's skills [27].

Issue-3. Some repositories such as Rising¹ have no well-defined pattern classification schemes [28].

Solution based classification schemes are used to organize design patterns on the basis of relationship between pattern and non-pattern classes, and structural dependencies between pattern classes. Usually, the facet and concept analysis are performed on attributes and relationships between objects to cluster the classes into groups. The main limitation of solution-based classification schemes [29, 30] are:

Issue-4. Existing solution-based schemes are limited to analysis of class diagrams which restrict their applicability across the other pattern groups. For example, in the case of SOA⁵ (Service Oriented Architecture) design patterns, solution of patterns is described through notation of data flow/flow chart diagrams.

Issue-5. Time and efforts are required to specify the patterns before classification [26, 31, 32, 33].

The selection of design pattern with respect to a design problem is more understandable than matching the patterns with the existing solutions. However, the sensitivity of pattern selection is related to their context, intent, and relationship with other patterns. For example, the canonical^a solution of Composite GoF (Gang-of-Four) design pattern is the result of formulating the recurring structural and compound design problems related to the composition of objects from its nesting and building tree structures. The researcher's existing efforts for the introduction of tools, recommendation systems, and methods used to automate the selection of design patterns are categorized into three approaches, which are UML-Based [34-36], Ontology-Based [26, 37-39], and Text categorization based approaches [31-33]. The recommendation system, tools, or methods which are designed and implemented by following the UML-based and ontology based approaches have limitations, such as:

Issue-6. The applicability of these UML-based automated systems is limited to those patterns catalog whose solution is described using class diagrams. For example, in the case of Service Oriented Architecture (SOA)⁵ and interface layouts of Human Computer Interaction (HCI) pattern [13], the solution of recurring problems are formulated using data flow/flowchart and visual diagrams (with description), which present the incompatibility of UML-based approach.

Issue-7. The formal specification of design problems is required to present an ontology-based automated system and assure its applicability to all types of design patterns. For example, in the case of knowledge representation and information extraction [26], Case Base Reasoning (CBR) [38], and use of linguistic theories [37] to analyze the design pattern are required the formal specifications to organize the design problems.

Issue-8. Usually, the tool (UML-based or Ontology-based) specific repositories are static and the inclusion of a new pattern in the corresponding repository requires time to analyze and classify the new pattern.

^aThe term canonical is used for the solution part of a design pattern presented by a domain expert. In the proposed study, the term canonical referred to the expert's solution which is provided to solve a design problem.

⁵<http://soapatterns.org/>

Subsequently, computation time and efforts are the two main issues which have been investigated in the [26, 34-39] studies of automated systems for the design pattern selection using UML-based and ontology-based approaches. The text classification approach has been applied in many domains to automate the systems to decrease the cost and computational time. Subsequently, the text categorization approach aid to classify the texts into proper classes with respect to their contents, such as spam e-mail filtering [40], web page classification [41], sentiment analysis [42], author identification [43], and the design pattern selection [31-33]. In their studies [26, 31-33], the authors consider the problem of design pattern selection as an information retrieval problem like other domains [40-43]. In the domain of design pattern selection, text categorization approach has been exploited through supervised learning for the introduction of an automated system [31]. In addition to the promising results, there are few limitations in the implementation of the proposed automated system.

Issue-9. A large sample size is required for precise learning of supervised learners [44, 45].

Issue-10. The individual training of a classifier for each pattern class (Group of patterns already classified by an expert/author/developer). This is feasible in the implementation of a proposed prototype for those pattern catalogs which have a small number of pattern classes. For example, in the case of Gang-of-Four (GoF) and Douglass pattern collection, patterns have been grouped into 3 and 5 pattern classes respectively. However, in the case of using Rising¹, euroPLoP², and SOA⁴ Patterns repositories the computation time overhead in term of the number of supervised learners will be increased [44-46].

Issue-11. The applied feature selection methods are biased to their discriminating power which might affect the precise learning of supervised learners [47, 48] and referred as a multi-class problem

In order to discuss the above mentioned issues related to design pattern classification and selection issues, we proposed an approach, based on unsupervised learning technique, and use the problem definition part of design patterns to automate the classification and selection of design pattern(s) for the given design problems. Subsequently, we propose an evaluation model to assess the effectiveness of the proposed approach.

1.2. Problem Statement

Based on the existence of several classification schemes for the organization of design patterns of same [5-7] or different domains [5, 12, 13, 15] and their documentation using certain resources such as books [5-19], proceedings², and online repositories/libraries⁶⁻¹⁴, the issue of design pattern searching and selection has been raised in [22, 49] studies. Researchers have introduced many automated systems using different approaches for the classification and selection of design pattern for the given design problems. The existence of these systems is limited to.

1. The existence of more than one classification scheme to resolve the design problem of the same domain [23, 49].
2. Lack of well-defined classification schemes for the design patterns hosted on online repositories [22, 49].
3. The differences in the borrowed techniques to present the solutions of recurring design problem [31, 35, 49].
4. The ranking of patterns with respect to the text relevancy of design problems are still main issues which need to be addressed. In this regard, the problem statement is formulated as follows.

⁶<http://wiki.c2.com/?DesignPatterns>

⁷<https://mediatech.aalto.fi/~pnr/Tik-76.278/gof/html/>

⁸http://groups.ischool.berkeley.edu/ui_designpatterns/webpatterns2/webpatterns/home.php

⁹<http://www.welie.com/>

¹⁰<http://www.rdrop.com/~half/Creations/Writings/Web.patterns/index.html>



Problem 1. Design pattern classification schemes are biased to developer's experiences and enough knowledge is required to understand the pattern classes and correlation of patterns in a same pattern class.



Problem 2. The lack of well-structured design pattern classification schemes for the categorization of patterns hosted on online repositories also restrict the organization of patterns into appropriate number of pattern class, which increase the time and effort overhead for better understanding.



Problem 3. Design patterns have been cataloged and formulated according to the developer's interest in the target domain and the context of borrowed techniques such as UML diagrams, Data Flow diagrams, or flowcharts. Existing automated techniques used for the identification and selection of patterns are limited to the context, specified domain, application language, and constraints related to their formal specification, which make their applicability limited to a specific pattern catalog.

1.3. Illustration of Problem Statement

1.3.1. Illustration-1 (More than one classification schemes for patterns of same domain)

Example 1. In the domain of object-oriented development, Gamma et al. [5], Coad et al. [7], and Pree et al. [6], present their own classification schemes to group the design patterns with respect to certain aspects. For example, Gamma et al. considers the instantiation, composition, and communication of class objects, and group the 23 design patterns as solutions for most commonly occurred recurrence design problems. Gamma et al. group the design pattern into three broad categories named Creational, Structural and Behavioral [5]. Coad et al. [7] considers the stereotypical responsibilities and scenario interactions aspect in order to investigate 31 patterns employed by developer's of five well-known industry applications and group them into four categories named Transaction, Aggregate, Plan and Interaction. The names of industrial applications are 1) Connie's Convenience Store, 2) Wally's Recursive Structures and Abstract Coupling between objects to classify the pattern collection [7].

Example 2. In the domain of Human Computer Interaction (HCI), Duyne et al. [13] and Tidwell [14] group and categorize the patterns related to the interactive designing interfaces and implemented for the web sites, mobile devices and desktop applications. Duyne et al. present a classification scheme based on certain design issues and group the 90 patterns into 12 categories. Tidwell [14] describes the 18 design patterns to resolve recurring design problems of user interfaces for the desktop, mobile, and website applications without presenting any classification scheme.

Implications of Illustration-1.

- A developer needs to have enough knowledge to understand the classification scheme and its aspects to resolve a design problem.

¹¹<http://www.educationalpatterns.org/>

¹²<http://designinginterfaces.com/about-the-book/>

¹³<https://developer.yahoo.com/ypatterns/>

¹⁴<http://www.hillside.net/patterns/patterns-catalog>

- For a particular domain and application, numerous patterns have been cataloged to resolve different recurring design problem.

Illustration-2 (lack of well-defined classification schemes for the design patterns hosted on online repositories)

Example 3. Repositories such as Rising - Patterns Almanac 2000¹, EuroPloP², and Portland Pattern Repository³ help to host patterns of different domains and applications. Moreover, there are certain repositories such as Rising - Patterns Almanac 2000¹ which host the pattern catalogs without a well-defined classification scheme [28]. In Table 1.1, some inconsistencies^b and overlapped^c categories related to Almanac 2000¹ are described.

Table 1.1. Issues in online repository Rising-Patterns Almanac 2000

Issue	Pattern Catalogs	Factors for the Categorization
Inconsistency	Transportation and Accounting	Domain
	Design Process and Testing	Methodology
	Creational, Structural and Behavioral	Author's experience
	Java, C++, and Smalltalk Idioms	Programming Language
Overlapping	Fault Tolerant and Event Driven Systems	Application
	Fault Tolerant and Event Driven Systems	Creational pattern class

Implications of Illustration-2.

- A developer needs to have the knowledge, time, and efforts to mine a pattern from the pattern catalogs hosted with inconsistent classification.
- Overlapping of pattern categories also affects the understanding of developer about a pattern catalog.

1.3.2. Illustration 3. (Differences in the borrowed techniques to present the solutions of recurring design problem)

Example 1. In the domain of object-oriented development, we consider a real world design problem described as follows.

Design Problem-1 (DP-1). “The waiter/waitress collects an order from the customers, encapsulate and published the order, and order is responded with a short order cook [50]”.

Due to the problem context similarity, the solution of design problem according to pattern suggested from [5], [7], and [50] catalogs are shown in Figure 1.2-a, Figure 1.2-b, and Figure 1.2-c respectively.

^bThe term inconsistency referred to existence of design patterns of the same domain, but are introduced via different applications, domains and the author's experience. For example, object-oriented design patterns which are recorded via different application languages such as C++, Java and smalltalk.

^cThe term overlapped referred to existence of design patterns of the same domain, but are introduced using different aspects recommended by the domain expert.

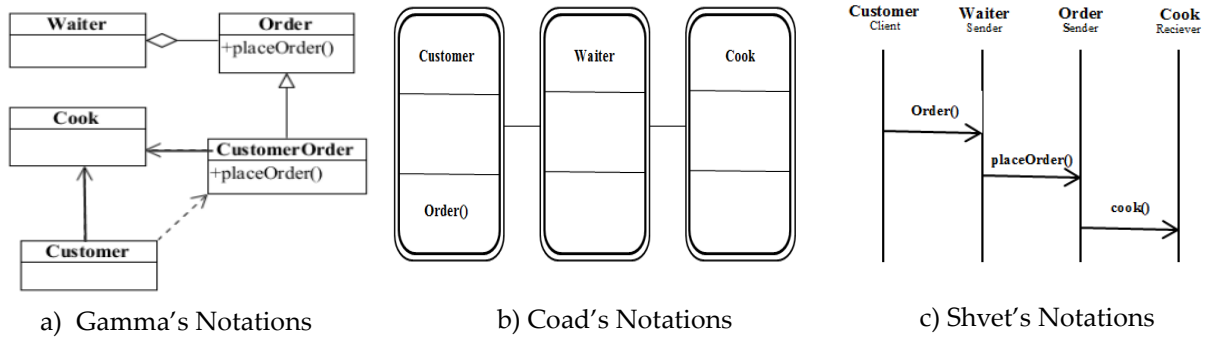


Figure 1.2. Scenario to describe the employment of pattern for the given design problems

Implications of Illustration-3.

- A developer needs to have the knowledge and understanding of notations (formal or author's specific [7]) which are used by skill-full developers to describe and publish the design pattern.

1.3.3. Illustration 4. (Difference in performance of automated systems in the existence of variants^b)

UML-based and ontology-based automated systems for the identification of design patterns are limited to the context (i.e. Specific to the repository [26] and also known as context-aware tools) for which they are designed and implemented. The existence of variants of canonical solutions of design patterns can aid to extend the pattern collection [9]. The inclusion of new patterns in a (tool specific) static repository required time for its analysis and classification [26], such as.

Example 1. We consider three variant solutions used to solve the following design problem (Design Problem 2 (DP-2)). Subsequently, we used the graph similarity score and pattern clustering algorithms and described their procedures.

Design Problem-2 (DP-2). “Design a DVD market place work. The DVD marketplace provides DVD to its clients with three categories: children, normal and new. A DVD is new during some weeks, and after change category. The DVD price depends on its category. It is probable that the system evolves to take into account the horror category [9]”.

The problem context and consequences of state design pattern and the variant solutions for the given design problem are shown in Table 1.2. Consequently, we discuss the applicability of graph similarity score and pattern clustering algorithms as follows.

Graph similarity score algorithm for Detection of Design Patterns:

The graph similarity scoring algorithm is used to find the design patterns [30]. The main steps of the graph similarity algorithm are.

Step-1. Represent the canonical solution of each design pattern into directed graph and mapped into a square matrix.

Step-2. Construct generalizations, associations, abstract class, and abstract method invocation matrices and compute the similarity score between two matrices.

^dThe term variant is used for an alternative solution of a design problem whose canonical solution has been recommended by a domain expert. In a design pattern repository, there may be certain variants which are introduced by developer community who are sharing the repository.

¹⁵<http://ui-patterns.com/>

¹⁶<http://stackoverflow.com/sites>

Table 1.2. Canonical and variants for the Design Problem-2 (DP-2)

Pattern/Variant	Problem Context and Consequences	Solution
Pattern	<p>The state pattern allows an object to change its behavior when its internal state changes. The object must change its behavior at run-time depending on that state. This lets you treat the object's state as an object in its own right that can vary independently from other objects. The state behavior of the object should be decoupled and can be managed during modification. The addition or removal of any new state should occur without code modification.</p>	
Variant-1 for DP-2	<p>Through the use of inheritance links, we are able to distinct the DVD to its states. A new case can be added without implying code modification. The state modification can be managed at running time and DVD class will not be modified whenever a new category is added</p>	
Variant-2 for DP-2	<p>The category of DVD is formulated as a subclass of DVD by imposing the instances destruction. This solution is only valid in using a prototype creational pattern with a category as parameter. Since, each category (i.e. Subclass of DVD) are required as the destruction of instances. The state modification cannot manage at running time, however, the extensibility is possible due to inheritance link.</p>	
Variant-3 for DP-2	<p>The states of DVD class are managed through enumeration, which can be imposed through the use of the switch statement. This thing aids us that changes in category are possible, however, the extensibility remain the main concern. The DVD class will be modified whenever a new category is added.</p>	

We applied these steps to the canonical solution of state pattern and three variants of design problem 2(DP-2) depicted in Table 1.3. Generalizations, associations, and abstract method invocation matrices for these variants including the canonical solution of State pattern are shown in Table 1.3. From the results of Table 1.3, the main implications are.

Table 1.3. Generalization, association, and abstract method call matrix for solution of pattern and variants of Design Problem 2 (DP-2)

Pattern/ Variant	Generalization matrix							Association Matrix							Abstract Method Call Matrix								
State Pattern		Context	State	C-State1	C-State2	C-State3	C-State4		Context	State	C-State1	C-State2	C-State3	C-State4		Context	State	C-State1	C-State2	C-State3	C-State4		
	Context	0	0	0	0	0	0		Context	0	1	0	0	0	0		Context	0	0	0	0	0	0
	State	0	1	0	0	0	0		State	0	0	0	0	0	0		State	0	0	0	0	0	0
	C-State1	0	1	0	0	0	0		C-State1	0	0	0	0	0	0		C-State1	1	0	0	0	0	0
	C-State2	0	1	0	0	0	0		C-State2	0	0	0	0	0	0		C-State2	1	0	0	0	0	0
	C-State3	0	1	0	0	0	0		C-State3	0	0	0	0	0	0		C-State3	1	0	0	0	0	0
	C-State4	0	1	0	0	0	0		C-State4	0	0	0	0	0	0		C-State4	1	0	0	0	0	0
Variant-1		DVD	Category	Children	Normal	Horror	New		DVD	Category	Children	Normal	Horror	New		DVD	Category	Children	Normal	Horror	New		
	DVD	0	0	0	0	0	0		DVD	1	1	1	1	1	1		DVD	0	0	0	0	0	0
	Category	0	1	0	0	0	0		Category	0	0	0	0	0	0		Category	1	0	0	0	0	0
	Children	0	1	0	0	0	0		Children	0	0	0	0	0	0		Children	1	0	0	0	0	0
	Normal	0	1	0	0	0	0		Normal	0	0	0	0	0	0		Normal	1	0	0	0	0	0
	Horror	0	1	0	0	0	0		Horror	0	0	0	0	0	0		Horror	1	0	0	0	0	0
New	0	1	0	0	0	0		New	0	0	0	0	0	0		New	1	0	0	0	0	0	
Variant-2		DVD	New	Children	Normal	Horror			DVD	New	Children	Normal	Horror			DVD	New	Children	Normal	Horror			
	DVD	0	0	0	0	0			DVD	0	0	0	0	0			DVD	0	0	0	0	0	
	New	1	0	0	0	0			New	0	0	0	0	0			New	1	0	0	0	0	
	Children	1	0	0	0	0			Children	0	0	0	0	0			Children	1	0	0	0	0	
	Normal	1	0	0	0	0			Normal	0	0	0	0	0			Normal	1	0	0	0	0	
	Horror	1	0	0	0	0			Horror	0	0	0	0	0			Horror	1	0	0	0	0	
Variant-3		DVD	Category						DVD	Category						DVD	Category						
	DVD	0	0						DVD	0	1					DVD	0	0					
	Category	0	0						Category	0	0					Category	0	0					

- The design problem context, consequences, and solutions of each variant of design problem 2 (DP-2) are formulated and catalog with pros and cons. However, the identification of the most appropriate pattern for design problem 2 depends on the capacity of the tool. For example, in the case of DP-2, if the developer has more concern with extensibility via inheritance, then the solutions of Variant-1 and Variant-2 for DP-2 could be considered. However, the tool capacity (i.e. The function of a Graph Similarity algorithm) is limited to the solution of Variant-1.

- According to tool capacity (i.e. Function of Graph Similarity algorithm), the Generalizations, Associations, and Abstract method invocation matrices of variant-1 are closer to the corresponding matrices of State pattern. Consequently, instantiation of variant-1 could be recommended without considering developer's concern about extensibility.

Pattern Clustering Algorithm for Identification of Design Patterns

The pattern clustering algorithm is used to find the design patterns in the given source code. The main steps of pattern clustering algorithm are.

Step-1. Each pattern structure is transformed into the ordered pair P (Generalized form is shown in equation 1.1) using a set of relations R (shown in equation 1.2).

$$P = \{R_k(i, j) \mid (i, j) \in R, i \in C, j \in C, 1 \leq k \leq n\} \quad (1.1)$$

$$R = \{\text{Association}(A), \text{Generalization}(G), \text{Aggregation}(E), \text{Composition}(C), \text{Dependency}(D), \text{Realization}(R)\} \quad (1.2)$$

Where the terms P and R denote the pattern's order pair and the relationship between two classes. The terms i, j , and k are used to describe the class number and relationship. The following pseudocode is used to transform a pattern structure into ordered pair P .

Algorithm: Transformation of Pattern structure in Order Pairs

First Step: $P_1 = \{R_1(i_1, j_1) \mid a \rightarrow i_1, b \rightarrow j_1\}$
 OtherSteps: $2 \leq k \leq n$
 $P_1 = \{R_k(i_k, j_k) \mid \begin{array}{l} \text{if } i_k = i_1, \text{ then } i_k \rightarrow a, j_k \rightarrow x \\ \text{if } j_k = i_1, \text{ then } i_k \rightarrow x, j_k \rightarrow a \\ \text{if } j_k = j_1, \text{ then } i_k \rightarrow x, j_k \rightarrow b \\ \text{if } i_k = j_1, \text{ then } i_k \rightarrow b, j_k \rightarrow x \end{array}\right\}$

Step-2. The ordered pair of a variant P_v is transformed according to described relations among classes and identify with respect the ordered pair of foundation patterns P_f . The closeness (i.e. Closeness) between two ordered pairs P_v and P_f aid to identify and cluster the similar patterns by satisfying the equation 1.3.

$$P_f \subset P_v \quad (1.3)$$

We applied these steps to the canonical solution of state pattern and three variants of design problem 2 (DP-2) depicted in Table 1.4. The ordered pairs P_v for variant-1 and P_f for the pattern are satisfying the equation 1.3, and it can be clustered with foundation pattern.

Table 1.4. Generated ordered pairs via pattern clustering algorithm for the solution of pattern and variants of Design Problem 2 (DP-2)

Pattern and Variants for Design Problem 2 (DP-2)	Ordered Pair
	$P_f = \{G(2,1), G(3,1), G(4, 1), G(5, 1), E(1, 6), A(6, 1)\}$
	$P_v = \{G(2,1), G(3,1), G(4, 1), G(5, 1), E(1, 6), A(6, 1)\}$
	$P_v = \{G(2,1), G(3,1), G(4, 1), G(5, 1)\}$
	$P_v = \{A(1,2)\}$

1.4. Research Objective and Research Questions

The main objective of this research is to consider design pattern classification and selection [26, 28, 31] as an information retrieval problem and target the relevant issues discussed in Section 1.1. In this regard, we propose an approach to organizing the design patterns (with or without a given design problem) of a catalog by considering the problem description part of each pattern. Subsequently, in the case of design pattern classification, we set the following two objectives to find the precise number of pattern classes (i.e. Clusters).

Research Objective-1. Through the proposed approach, a novice developer will be able to classify the patterns into an appropriate number of groups (i.e. Pattern classes) without gaining enough knowledge to understand the classification schemes (implemented by experts) of the target repository.

Research Objective-2. Through the proposed approach, a novice developer will be able to organize the patterns published on repositories (such as Rising¹), where no classification scheme is implemented or heterogeneity of pattern descriptions exists in the organization of software design patterns.

Subsequently, in the case of design pattern selection, we set the following research objective.

Research Objective-3. Through the proposed approach, a novice developer will be recommended to right design pattern and an appropriate number of the ranked design pattern(s) for the given design problem.

In order to achieve our objectives, we extract the following research questions.

Research Question-1 (RQ-1). Is the use of the proposed approach effective to organize the design patterns with respect to the classification scheme of experts in different domains?

Research Question-2 (RQ-2). Is the use of the proposed approach effective to determine the right number of pattern classes for the given sample of design problems?

Research Question-3 (RQ-3). Is the use of the proposed approach effective to determine the right design pattern for the given design problem?

Research Question-4 (RQ-4). Can the performance of proposed approach be tuned with respect to applied weighting methods and feature selection techniques?

Research Question-5 (RQ-5). Are the existing feature selection methods aiding the proposed approach to overcome the multi-class problem?

1.5. Motivation

There are certain factors which motivate us to propose and implement a new approach and exploit it through the use of unsupervised learning techniques.

1. Classification schemes for the existing design pattern catalogs are biased to researcher's interest and experience. The existence of several classification schemes in the same domain required enough knowledge to understand the classification aspects [23, 26, 49].
2. The lack of well-defined classification schemes or heterogeneity of pattern descriptions to categorize the patterns on the online repositories. Such as Rising - Patterns Almanac 2000¹[28].

3. In case of UML-based and ontology-based automated systems, the time and effort overheads are required for the formal specification of patterns in the context of selection of right design pattern for the given design problem [26, 31-33].
4. The text categorization based automated systems which are exploited through the supervised learning techniques are limited to the sample size for classifier's precise learning. Subsequently, the numbers of classifiers are required for their training with respect to each pattern class to handle the multi-class problem [31].
5. Researchers have started to adopt text categorization approach in certain domains such spam email filtering [40], web page classification [41], sentiment analysis [42], and author identification [43] by considering them as an information retrieval problem.
6. In the domain of design pattern selection, the authors have proposed methods using text categorization approach and exploit through supervised learning, knowledge representation and information extraction techniques [26, 31].

1.6. Introduction of Software Design Patterns

In the domain of software engineering, a pattern is referred as a reusable solution to resolve a common recurring problem within the context of software design. Moreover, a software design pattern is considered as a reflection of the developer's knowledge and experience [3, 4]. The term pattern coined by Alexander in his two famous books entitled "The Timeless Way of Building" [3] and "A Pattern Language: Towns, Building, Constructions" [4]. Alexander has recommended that a pattern should be described in three sections named Context, Description of the problem, and the Solution. However, each section can be described in sub-section depending on the developer's interest and the template which is followed to publish the patterns of a domain. For example, in the domain of object-oriented development, Gamma et al. [5] has introduced 23 design patterns and each pattern classified into two sections named Problem Domain and Solution Domain. The Problem Domain section of each pattern is classified into intent, motivation, and applicability sub-sections, while the Solution Domain section is classified into the structure, participant collaboration, consequences, implementation, and related patterns.

The layouts of Composite, Pool Allocation, Identification and Authentication (I&A), and News Mosaic patterns related to GoF, Douglass, Security, and Duyne design pattern collection are shown in Appendix A. Moreover, in a recent study, the authors have depicted the interest of the software engineering research community for the use of the GoF design patterns [53, 54] and relationship with quality attributes [55]. The rationale between the use design patterns and pattern documentation (i.e. Pattern language) has been empirically investigated and aid to construct a pattern catalog. Usually, the designers follow a procedure (described in Figure 1.1) started from a design problem and ended at the employment of more suitable design pattern. However, there are two main issues with this procedure that are 1) searching of relevant design patterns, and 2) selection of right pattern for any given design problem [26, 31, 56]. The searching of relevant design patterns depends on the well-defined documentation and classification of patterns into groups (i.e. A class of relevant patterns). The list of well-documented software design pattern catalogs with their application domains are shown in Table 1.5. Subsequently, some of these pattern catalogs have been hosted on online repositories/libraries, for example, GoF^{6, 7} and Tidwell¹² design pattern catalogs. However, in case of emerging (due to variant patterns) repositories² of pattern catalogs or inconsistencies in the pattern categorization schemes¹, classification of patterns into appropriate number of classes^{1,2} remain an issue.

Table 1.5. Summary of approaches used for problem-based classification of design patterns

Author	Domain	High Level Categories
Gamma et al. [5]	Object-oriented Design Problems	Creational, Structural and Behavioral
Pree [6]	Object-oriented Design Problems	Pree's classification scheme relies on Recursive Structures and Abstract Coupling
Coad et al. [7]	Object-oriented Design Problems	Transaction, Aggregate, Plan and Interaction
Tichy [8]	Software Design Problems	Decoupling, Variant Management, State Handling, Control, Virtual Machines, Convenience, Compound, Concurrency and Distribution
Rising [11]	Application Type	Accounting, Hypermedia, Trading, C++ Idioms, System Modeling, Air Defense, Interactive, Database, Persistence and Trading
Douglass [12]	Real Time	Architectural Design, Safety and Reliability, Memory, Concurrency Resources and Distribution
Duyne et al [13]	Web Designing	Classify 90 pattern into 12 design pattern group to target different levels of web design.
Tidwell [14]	User Interface designing	The authors used different facets of user interface design and classify the 18 design patterns in the content organization, navigation, page layout and action/command groups.
Schumacher et al. [15]	Security Patterns	The authors group 46 security patterns into 8 categories SACA (System Access and Control Architecture), ACM (Access Control Model), IA (Identification and Authentication), OSAC (Operating System Access Control), SIA (Security Internet Application), FA (Firewall Architecture), ESRM (Enterprise Security and Risk Management) and Accounting
Nilsson [51]	User interface for mobile applications	The author targets the user interface related problems in the domain of mobile application and categorized the patterns with respect to three problem areas named utilizing screen space, interaction mechanisms and design at large.
Lano and Kolahdouz [52]	Model Driven Development	Authors target the design patterns of model transformation languages and categorized them using the meta-modeling approach.

The second issue is selection of right pattern or ranking of pattern in the existence of relevant patterns of a class. On the basis of experience and knowledge, the designers who make their attempts for the selection of design patterns

can be classified into four types: For example, 1) Designers who have both experience and knowledge of patterns of specific domain, 2) Designers who have worked experience in resolving certain design problems of specific domain without knowledge of cataloged patterns of the corresponding domain or not showing any interest in using the cataloged patterns, 3) Designers who have knowledge of the pattern catalogs of a specific domain without any work experience, 4) Designers who are starting their work without knowledge of pattern of a specific domain. For each group of designers, time and effort are required for the adequate selection of design pattern for on hand problems.

Due to the rapid increase in the number of pattern repositories in different domains, the existence of repositories to host patterns with inconsistent classification schemes^{1,2}, the rapid growth in the size of repositories by introducing the new pattern in the same domain^{1,2}, and introduction of spoiled and multi-disciplinary patterns [9], the selection of right design pattern always remain an issue. The researchers have made their efforts to aid designers (of all types) by introducing some sort of automated techniques, methodologies, prototypes, or tools. In the domain of automation, the researcher's efforts can be viewed in many aspects such as recommendation system, pattern detection tools, the inclusion of new ontology and formal tools, and automated systems. Existing efforts can be classified into three groups, that are the use of 1) UML-based approach [30, 34-36], 2) Ontology-based approach [26, 37-39], and 3) Text categorization approach [31]. However, the specification of design patterns and problem, tool specific to particular pattern catalog (i.e. Context-aware tools), inconsistencies in the borrowed technique used to describe the solution of pattern, sample size, and the multi-class problem (in the case of feature selection) are the still issues needs to be resolved.

1.7. Online Software Design Pattern Repositories

For the last two decades, researchers have started to catalog their patterns on online repositories in different domains such as in the domain of object-oriented development^{6,7}, service oriented architecture⁴, human computer interaction^{8,12,15}, web design⁹, pattern languages³, education¹¹, personal website¹⁰ and pattern repositories of hybrid domains¹. Subsequently, several online libraries/groups are available to host certain pattern catalogs of different domains such as EuroPlop², Yahoo¹³, Wiki⁶, and Linda Rising-Patterns Almanac 2000¹. There are some facets which are derived from the emergence of new patterns and regular update of online repositories and libraries, such as.

1. The well-define template of a repository to record a new pattern depends on classification schemes introduced by the authors. For example, in the domain of service oriented architecture⁴, the pattern can be grouped according to a well-defined template which is described for the recording of any new pattern.
2. The repositories which are hosting patterns of different domain depict the inconsistencies in their different classification schemes. For example, Linda Rising-Patterns Almanac 2000¹, patterns are grouped with respect to a domain (Accounting and health care), application (Fault Tolerant and Event Driven systems), methodology (Testing and design process) category or the category which specified according to developer's knowledge and interest (Creational, Structural, and behavioral in case of GoF patterns).
3. The pattern of a same domain hosted on the same library/group with different classification schemes. For example, on hillside¹⁴ pattern library, the interaction pattern to aid designers in their interactive designing.

4. Some repositories welcome to record the variants of canonical solutions. For example, the publishing of new variant of patterns on GoF design pattern⁶ repository.

1.8. Widely-known Design Pattern Collections

To evaluate the effectiveness of the proposed approach, we consider widely-known design patterns collection by considering three parameters, 1) application domain, 2) number of design patterns, and 3) the number of design pattern classes. The brief description of each design pattern collection is as follows.

1.8.1 Gang-of-Four Design Patterns

The aim of these design patterns to describe the simple and elegant solutions to specify the recurring design problems in the context of object-oriented development. These design patterns aid those developers who are struggling for the flexibility and reuse in their object-oriented software. The authors of the GoF design patterns introduce 23 design patterns by considering different aspects related to object and classes, such as abstraction, coupling, and cohesion. Subsequently, these design patterns are grouped into Creational, Structural, and Behavioral categories (a.k.a Design pattern classes).

1.8.2 Douglass Design Patterns

The aim of these design patterns to describe the sophisticated solutions to specify the recurring design problems in the context of designing of real-time applications. These design patterns aid those designers who are struggling to design large-scale, dynamic, complex, and distributed real-time based computing systems. Douglass introduces 34 design patterns by considering different aspects related to memory and resource utilization in the distributed domain. Subsequently, these design patterns are grouped into Memory, Resource, Safety and Reliability, Concurrency, and Distribution categories.

1.8.3 Security Design Patterns

The aim of these design patterns to describe the sophisticated solutions to specify the recurring design problems related to the security issues in software. These design patterns aid those architects who are struggling to address the security requirements which are often expressed vaguely and causes to limit the relevant decisions. The authors of security design patterns introduce 46 design patterns by considering different security aspects in softwares. Subsequently, these design patterns are grouped into Enterprise Security and Risk Management(ESRM), Identification and Authentication (I&A), Access Control Models (ACM), System Access Control Architecture (SACA), Operating System Access Control (OSAC), Firewall Architecture (FA), Secure Internet Applications (SIA), and Accounting categories.

1.8.4 Duyne Design Patterns

The aim of these design patterns is to represent the elegant solutions to specify the recurring design problems in the context of web-based development. These design patterns aid the web design professionals, business managers and client, who are associated with web-based development in certain aspects. Duyne introduce 90 design patterns by considering different aspects by considering different security aspects in web based softwares.

Subsequently, these design patterns are grouped in to Site Genre, Creating a navigation framework, Creating a powerful homepage, Writing and managing content, Building Trust and credibility, Basic E-commerce, Advances E-commerce, Helping customer's complete tasks, Designing effective page layout, Making site search fast and relevant, Making navigation easy, and Speeding up your site categories.

1.9. Text Categorization Approach

In a recent systematic study, the authors [57] reported that the majority of software engineering data is in unstructured form and researchers are applying information retrieval techniques to aid software development procedure. In an information retrieval technique, text categorization is an important component which is used to classify the unstructured or free text documents into predefined categories. Unlike structured data (e.g. Relational database), the organization of unstructured data is not clear, semantically obvious, and easily understandable [57,58]. The use of statistical techniques and Machine Learning (ML) in the domain of information retrieval aid to apply text categorization approach and make sense of unstructured data in research communities of certain domains such as spam e-mail filtering [40], web page classification [41], sentiment analysis [42], author identification [4] and the design pattern selection [31-33]. There are several techniques with their pros and cons used for the text categorization, such as KSONN (Kohonen Self-Organizing Neural Network), CNN (Cellular Neural Network), use of Bayes Classifiers [59], BOW (Bag of Words) model [60], Text Mining [31, 61], and N-Grams [62]. The approach (statistical or machine learning), method (Frequency of keywords, probabilities of word occurrences, construction of a tree, semantic relationship between words, construction of topic models, and preprocessing analysis, which includes the weighting methods and feature extraction), and text category (discrete and continuous test) are varied across the text categorization techniques. The text mining technique for the text categorization is introduced by Feldman and Dagan [63] to analyze the text using machine learning methods (such as supervised and unsupervised learning methods) in the context of an information retrieval, Natural Language Processing (NLP), and information extraction approaches [64]. The core steps of text mining are as follows.

1.9.1. Preprocessing Activities

In text mining, machine learning methods either supervised or unsupervised are used to analyze the text. These methods cannot apply directly to unstructured text. We need to perform some preprocessing activities which transformed unstructured text into a structured form [57, 58, 63, 64]. The precision of supervised and unsupervised learning is highly related to the selection of pre-processing activities used to map the unstructured text into a structured form. The list of preprocessing activities is shown in Figure 1.3.



Figure 1.3. Set of preprocessing activities

The list of pre-processing activities is described briefly in the following sub-sections.

1.9.1.1. Removal of Stopword

The removal of stop words activity is performed to decrease the noise in textual data and reduce the feature space for the machine learning methods to produce the correct results. The term stop word is coined for those words in a natural language which are frequently used and are meaningless in joining of words in a sentence. For example, the pronouns, conjunctions, and prepositions in a sentence which carries no information and considered insignificant for certain natural language processing based applications such as information retrieval, text categorization, text summarization, and clustering. However, the impact of stop words varies between the context of different text corpora. This means, the words which carry no semantic information in one context, might hold some sort of semantic information in another context, which cause the variations in performance of pre-processing tools. Usually, tools and algorithms for removal stop words from documents are designed and implemented using four methods, 1) Classic method (need a precompiled list) [65, 66], 2) Zipf's Law method (need impact analysis of words frequency) [67], 3) Term Based Random Sampling (need the use chunks of randomly selected text and a divergence measure) [68], and 4) Mutual Information Method (need to compute the mutual information between two terms as random variables and their document class) [69].

1.9.1.2. Word Stemming

Word stemming is performed to remove the various suffixes (such as -ing, -ed, -able, -s, -er, -en, -est, -t, and so on) and find the root or stem of the words. For example, the words “present”, “Presented”, “Presenting”, and “presentable” can be stemmed into a word “Present”. Like “Removal of stop words”, this activity is also used to reduce the number of words and reduce the feature and memory space. Based on their functionality and working metaphor, the stemming algorithms can be grouped into three categories such as 1) Truncating Stemmer (Porters [70], Paice [71], Lovins, and Dawson [72] algorithms), 2) Stemmer need Statistical analysis (N-Gram [73, 74], HMM (Hidden Markov Model) [75], YASS (Yet another Suffix Striper) [76]), and hybrid stemmers (Inflectional and Derivational stemmer [77, 78], Corpus Based Stemmer [79], and Context Sensitive [80] Stemmers).

1.9.1.3. Indexing

The indexing activity is performed to present a document as a vector of words (or a point in a space) and documents collection as Vector Space Model (VSM). Subsequently, it is assumed that vectors (i.e. Points) will be semantically similar when they are close together otherwise they will be apart semantically. On the base of matrix type, VSMs can be organized in three forms, 1) term-document, 2) word-context, and 3) pair-pattern to present the similarity between documents, word, and relations [81-88]. Commonly, a document collection is described through a term-document matrix D , where each entry refers to the word's occurrence in the documents.

$$D = (W_{w,d}) \quad (1.4)$$

In the equation 1.4, $W_{w,d}$ is the weight of word w_i in document d . Subsequently, before applying any weighting method, the frequency of a word is considered as their weights.

1.9.1.4. Weighting Methods

The weighting method activity is performed to find the weight ($W_{w,d}$) of a word w in document d . The frequency of the word w_i in document d and collection c of N documents is referred as $f_{w,d}$ and $tf_{w,c}$. There are numerous ways to determine the weight $W_{w,d}$. However, most of the approaches are based on the two empirical observations about the text [89-91].

1. High frequency of a word in a document represents its relevancy to the corresponding document.
2. High frequency of a word across the documents of a collection represents its poor discrimination between the documents either belong to same or different classes.

Subsequently, there are several weighting methods have been introduced. The most commonly used weighting methods are as follows:

A. Boolean

In this weighting method, the weight 1 is assigned to a word if it occurs in the document otherwise weight 0 is assigned to the corresponding word, shown in the equation 1.5.

$$W_{w,d} = \begin{cases} 1 & \text{if } f_{w,d} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (1.5)$$

B. Term Frequency (TF)

In this weighting method, the frequency of a word (i.e. Term) in a document is refers to its weight in the corresponding document, shown in the equation 1.6.

$$W_{w,d} = f_{w,d} \quad (1.6)$$

C. Term Frequency and Inverse Document Frequency (TF-IDF)

In this weighting method, the frequency of a term throughout the documents in the collection is used to compute its weight in the corresponding document. The weight of a word w in a document d of a collection is computed as the proportion of frequency of words in the document and inverse proportion to the number of documents in which word occurs at least one time, shown in the equation 1.7.

$$W_{w,d} = f_{w,d} \times \log\left(\frac{N}{tf_{w,c}}\right) \quad (1.7)$$

D. Term Frequency Collection (TFC)

Like TD-IDF, in this weighting method, the frequency of a term throughout the documents in the collection is also used to compute its weight in the corresponding document. The length of documents is normalized with M words in the collection, shown in the equation 1.8.

$$W_{w_i,d} = \frac{f_{w_i,d} \times \log\left(\frac{N}{tf_{w_i,c}}\right)}{\sqrt{\sum_{j=1}^M \left[f_{w_j,d} \times \log\left(\frac{N}{tf_{w_j,c}}\right) \right]^2}} \quad (1.8)$$

E. Length Term Collection (LTC)

In this weighting method, the weight of a word is computed using the logarithm of its frequency rather than its raw frequency. This weighting is applied to overcome the problem of large differences in the frequencies, shown in the equation 1.9.

$$W_{w_i,d} = \frac{\log(f_{w_i,d} + 1.0) \times \log\left(\frac{N}{tf_{w_i,c}}\right)}{\sqrt{\sum_{j=1}^M \left[\log(f_{w_j,d} + 1.0) \times \log\left(\frac{N}{tf_{w_j,c}}\right) \right]^2}} \quad (1.9)$$

F. Entropy

In this weighting method, the weight $W_{w_i,d}$ of a word is computed as its average uncertainty. The weight is computed as -1 for evenly distributed word, and 0 when the word occurs in only one document, shown in the equation 1.10.

$$W_{w_i,d} = \log(f_{w_i,d} + 1.0) \times \left(1 + \frac{1}{\log(N)} \sum_{j=1}^N \left[\frac{f_{w_j,d}}{tf_{w_i,d}} \log\left(\frac{f_{w_j,d}}{f_{w_i,d}}\right) \right] \right) \quad (1.10)$$

1.9.1.5. Feature Selection Techniques

In text classification based applications, removal of stop words and word indexing aid to reduce the number of unimportant and meaningless words. The high dimensional feature space (computed in terms of $y = f(x)$ and need to mapped in n-dimensional x_n , ($n=1, \dots, N$)) is still a key issue for the performance of classifiers. There are several techniques to reduce the dimensionality of the feature space and can be categorized into 1) feature selection methods, and 2) re-parameterization method.

A. Feature Selection method

In order to decrease the undesirable effect of useless features, feature selection methods are applied. Wrapper, embedded, and filter are three common schemes which are used to formulate the feature selection methods. The wrapper and embedded based feature selection methods required classifier interaction, while filter-based methods do not need such interaction to construct a feature set. Besides, the running time of wrappers and embedded methods may be increased due to the classifiers interaction [92, 47, 48]. The wrappers and embedded methods are specific to the learning model as compared to filter-based methods. Consequently, the use of filter-based methods is

recommended for the automatic systems implemented via text categorization approach. The most common global filter-based method are given as.

i. Document Frequency (DF)

The function of Document Frequency (DF) feature selection method can be performed in two steps. First, the total number of documents for a particular term is counted as its document frequency. Second, those terms of feature space are removed whose document frequency is less than some pre-determined thresholds [92]. The document frequency of a word/term in document d and across the document collection c is computed using the equations 1.11 and 1.12 respectively.

$$DF(w_i, d) = P(w_i d) \quad (1.11)$$

$$DF(w_i, c) = P(w_i c) \quad (1.12)$$

ii. Information Gain (IG)

In the domain of machine learning, the Information Gain (IG) is mostly used for as term(or word) goodness criteria to compute the information in the form of a number of bits and predict the presence or absence of a word in the document. Suppose $\{c_j\}_{j=1}^m$ present the set of classes (i.e. Categories) used to categorize the documents of a collection [83, 87, 92]. In such case, the Information gain (IG) of raw word/term is computed using the equation 1.13.

$$IG(w_i, c_j) = -\sum_{j=1}^m P(c_j) \log P(c_j) + P(w_i) \sum_{j=1}^m P_r(c_j | w_i) \log P_r(c_j | w_i) + P(w_i') \sum_{j=1}^m P_r(c_j | w_i') \log P_r(c_j | w_i') \quad (1.13)$$

iii. Mutual Information (MI)

In the domain of information theory for statistical language modeling, the Mutual information (MI) is used to measure the information content of features and their relationship across all classes. The equation 1.14 is used to measure the mutual information for the term w_i and its dependency with a document class c_j [90, 94]. Where $P(w_i, c_j)$ is referred as the probability of occurrence of the term w_i in the document class c_j .

$$MI(w_i, c_j) = \log \frac{P(w_i, c_j)}{P(w_i)P(c_j)} \quad (1.14)$$

iv. Chi Square

In the domain of mathematical statistics, the Chi square feature selection method is used to measure the independence of two random variables. For example, the term w_i and a document class c_j [95]. The formula of chi square is shown in the equation 1.15.

$$Chi(w_i, c_j) = \frac{N(f_{w_i c_j} f_{w_i' c_j} - f_{w_i c_j'} f_{w_i' c_j})}{(f_{w_i c_j} + f_{w_i c_j'})(f_{w_i c_j} + f_{w_i' c_j})(f_{w_i c_j} + f_{w_i' c_j'})(f_{w_i' c_j} + f_{w_i' c_j'})} \quad (1.15)$$

In equation 1.15, the terms N , $f_{w_i c_j}$, $f_{w_i c'_j}$, and $f_{w_i c_j}$ are referred as a total number of documents, the frequency of w_i for class c_j , the frequency of w_i for all classes except c_j , the frequency with which class c_j occurs without w_i , and frequency of event neither w_i nor c_j occurs respectively.

v. Gini Index (GI)

The original GI (Gini Index) is introduced to split the decision tree induction and later is improved for its use in text categorization with multi-class setting [96]. The formula of GI is shown in the equation 1.16.

$$GI(w_i, c_j) = \sum_{j=1}^M P(w_i | c_j)^2 P(c_j | w_i)^2 \quad (1.16)$$

Where the term $P(w_i | c_j)$ and $P(c_j | w_i)$ referred as the probability of term/word w_i for the class c_j and probability of class c_j for the term/word w_i respectively [96].

vi. Odds Ratio (OR)

The OR (Odds Ratio) feature selection method is used to measure the membership and non-membership of a term/word with a specific class [97, 47]. The formula to compute the Odd Ratio is shown in the equation 1.17.

$$OR(w_i, c_i) = \log \frac{P(w_i | c_i) [1 - P(w_i | c'_i)]}{[1 - P(w_i | c_i)] P(w_i | c'_i)} \quad (1.17)$$

Where the terms $P(w_i | c_i)$ and $P(w_i | c'_i)$ present the probability of w_i for the class c_j and conditional probability w_i for all classes except the class c_j .

vii. Distinguishing Feature Selector (DFS)

The DFS (Distinguishing Feature Selector) is used to eliminate meaningless features according to pre-defined criteria and select the distinctive features [98]. The formula for DFS is shown in equation 1.18.

$$DFS(w_i, c_j) = \sum_{j=1}^M \frac{P(c_j | w_i)}{P(w_i | c_j) + P(w_i | c'_j) + 1} \quad (1.18)$$

Where the terms $P(c_i | w_i)$ and $P(c'_i | w_i)$ presents the conditional probability of c_j for the presence and absence of term w_i respectively, while the $P(w_i | c'_j)$ present the conditional probability of w_i for all classes except c_j .

B. Re-parameterization

The Re-parameterization approach is used to construct the new features either through a combination or transformation of existing parameters. The Latent Semantic Indexing (LSI) [99-103], Independent Component Analysis (ICA) [104, 105], Probabilistic Latent Semantic Indexing (PLSI) [106-108], and Latent Dirichlet Allocation (LDA) [109-112] are common techniques which based on re-parameterization approach.

i. Latent Semantic Indexing (LSI)

The Latent Semantic Indexing (LSI) is the most common method based on the Re-parameterization approach. The assumption behind the working procedure of LSI is that there exists some latent structure formed through the patterns of word usage across all the documents of a collection. Subsequently, some statistical techniques are used to estimate the structure. The Singular Value Decomposition (SVD) technique is applied to compute eigenvector decomposition and factor analysis. Suppose, we have $M \times N$ term-by-document matrix A . Where M and N is referred as the number of terms and documents. The SVD of A is computed through the equation 1.19.

$$A = U \Sigma V^T \quad (1.19)$$

Where U and V are orthonormal columns and computed as $M \times R$ and $R \times N$ respectively, Σ is a diagonal matrix of singular values and computed as $R \times R$, and $R \leq \min(M, N)$ is considered as the rank of A .

ii. Independent Component Analysis (ICA)

The ICA technique is used to decompose any selected random variable into dimensions and each dimension is referred as a statistically independent component. The ICA is similar to LSI to 1) reduce the dimensionality of any term-document matrix, and 2) model documents in K -dimensions.

iii. Probabilistic Latent Semantic Indexing (PLSI)

The PLSI is also known as PLSA (Probabilistic Latent Semantic Analysis) and referred as a generative model which can address the statistical unsoundness of Latent Semantic Indexing (LSI).

1.9.2. Text Classification Techniques

In the machine learning domain, the text classification is exploited through supervised learning techniques. Recently, Delgado et al. [46] have evaluated the performance of 179 classifiers of 17 different families on 121 datasets retrieved from the UCL database to solve the real world classification problem. In the domain of text categorization based automated systems, there are several supervised learners, however, it cannot be decided that which one is better since no classifier can get better results for all problems [46]. Consequently, it is recommended to find the best classifier for a new problem. In the text categorization domain, researchers have used and reported the classification performance of C4.5 Decision Tree, Naïve Bayes, and Support Vector Machine (SVM) supervised learning techniques [31, 47, 64]. The brief description of these classifiers is given as.

1.9.2.1. Naive Bayes (NB)

The Naïve Bayes (NB) is a probabilistic classifier which is based on Bayes theorem. Naïve Bayes works on an assumption that all features are independent of each other and calculate a probability score by multiplying the conditional probabilities with each other. It is supposed that the pattern class $L(p_i)$ of pattern p_i , has some relation to words appears in the description of Problem Definition of design patterns and design problems [113]. The Bayesian formula describes in Equation 1.20 yields the probability of a pattern class given the words used in the description of patterns and design problem.

$$p(L_i | t_1, \dots, t_{m_i}) = \frac{p(t_1, \dots, t_{m_i} | L_k) p(L_k)}{\sum_{l \in L} p(t_1, \dots, t_{m_i} | l) p(l)} \quad (1.20)$$

The prior probability $p(l)$ depicts the probability that design problem belongs to a class label $l \in L$ before its known words. The conditional probability of words (includes in the description of patterns and design problem) given a class L_k is described in Equation 1.21.

$$p(t_1, \dots, t_{m_i} | L_k) = \prod_{j=1}^{m_i} p(t_j | L_k) \quad (1.21)$$

The estimation of the probabilities of words $p(t_j | L_k)$ is required to make the learning of Naive Bayes classifier which depends on their frequencies in documents (i.e. Design patterns) of the training set.

1.9.2.2. C4.5 Decision Tree

The decision tree based algorithms generate a set of rules to make the classification decisions. The C4.5 algorithm used the concept of information entropy and makes the decision from the training set. In order to predict the document's class, the word t_i is selected from a training set T of labeled documents. Subsequently, training set T is partitioned into two subsets $T+$ (documents with word t_i) and $T-$ (documents without the word t_i). The same step is repeated and applied to $T+$ and $T-$. The recursive procedure is stopped when all documents belonging to the same class L_k .

1.9.2.3. Support Vector Machines (SVM)

A Support Vector Machine (SVM) is considered as one of the most effective text categorization algorithms, which looks for a decision surface and determines a margin through its closest data points. Usually, an SVM algorithm can separate documents of a training set into two classes, $y=+1$ for the positive and $y=-1$ for the negative class. Subsequently, the equation 1.22 is used to define a hyper plane at $y=0$ for the set of input vectors. Each input vector for the document d is represented as a count of words as depicted in the equation 1.23.

$$y = f(\vec{t}_d) = b_0 + \sum_{j=1}^M b_j t_{dj} \quad (1.22)$$

$$\vec{t}_d = t_{d1}, \dots, t_{dM} \quad (1.23)$$

The SVM algorithm classifies a new document with $\vec{t}_d$ into a positive class if $f(\vec{t}_d) > 0$ otherwise into negative class [98].

1.9.3. Text Clustering Techniques

In the machine learning domain, the text clustering is exploited through supervised learning techniques. The aim of text clustering is to group the documents with respect to the similarity in their contents and each partition is

referred as a cluster of relevant documents. The two main assumptions for the clustering techniques are 1) Similarity between the documents of a partition, and 2) Dissimilarity between the documents of two or more than two partitions. Subsequently, clustering technique compute a partition on the basis of attribute's data and measures used to assess the similarity and dissimilarity between partitions. The brief description of widely used and empirically investigated clustering algorithms is given as follows.

1.9.3.1. Hierarchical Clustering Algorithms

The hierarchical clustering algorithms work either in bottom-up or top-down fashion. The algorithms which are designed in the bottom-up fashion such as “agglomerative” used the following steps

Step-1. The algorithm takes start by considering each document as a cluster.

Step-2. The clusters of the similar document are merged iteratively.

Step-3. When all clusters are merged into a single cluster, then finish.

Subsequently, the algorithms such as “Divisive” which are designed in the top-down fashion used the following steps.

Step-1. The algorithm takes start by considering all documents as a single cluster.

Step-2. In each iteration, a selected cluster is split into sub-clusters.

Step-3. When all documents are partitioned into an appropriate number of clusters then finish.

There are several distance measures are used to compute the similarity between two documents such as

Suppose the document set D (i.e. Considered as an individual cluster) and d is a document. During their working procedure, “Agglomerative” and “Divisive” takes start with d and D as an individual cluster, respectively. In each iteration, two clusters p and q are selected and a distance measure is applied to compute (i.e. $dist(p, q)$) the distance between two clusters iteratively. In case of “Agglomerative” algorithm, two clusters with minimum distance are merged, while in case of “Divisive” algorithm centroid of two clusters are aid to decide which one cluster should split. The most common distance measures are Ward, Single-link, and complete-link [114-116].

1.9.3.2. K-means

The k-means clustering algorithms have most often used algorithm because it is working procedure based on simple statistics, which can apply datasets of any size. There are certain variants of k-means have been introduced by their common working procedure, which is described through an algorithm as follows. [114,117], Bi-Section-k-means algorithm is one of a common variant of k-means clustering, which claim high quality partitioning the large size textual data.

Algorithm k-means ($D = \{d_1, d_2, \dots, d_N\}$, $P = \{P_1, P_2, \dots, P_k\}$, k)

{ This algorithm work on a partitioning P of the document's set D with k clusters }

Step-1. Selection of k data points from the D and each data point is considered as starting centroids

$$\vec{t}_{p_1}, \vec{t}_{p_2}, \dots, \vec{t}_{p_k}$$

Step-2. Perform Step-3 to Step-4 until cluster centroid $\vec{t}_{p_1}, \vec{t}_{p_2}, \dots, \vec{t}_{p_k}$ become stable.

Step-3. Assign each data point of P to the nearest centroid which is decided on the base of applied distance measure.

Step-4. Compute the new cluster centroids $\vec{t}_{p_1}, \vec{t}_{p_2}, \dots, \vec{t}_{p_k}$ of clusters $P_1, P_2, \dots, P_k$.

Step-5. Return the $P = \{P_1, P_2, \dots, P_k\}$ with stable centroid $\vec{t}_{p_1}, \vec{t}_{p_2}, \dots, \vec{t}_{p_k}$

}

1.9.3.3. Self Organizing Map (SOM)

The Self Organizing Map (SOM) is based on neural networks which can cluster the high dimensional data vectors with respect to use similarity measure. The neural network structure of SOM is described in two layers that are input and output layers, and each layer consists of a set of the neuron (i.e. Clusters). The input layer represents the input dimensionality which is included in the structure of data vector, while the output layer represents the output dimensionality [118]. The neurons (i.e. Clusters) of both layers are connected each other. The working procedure of SOM is described as follows.

Step-1. Each unit of the output layer referred as a cluster center.

Step-2. The unit of output layer become fixed and weighted before learning.

Step-3. The notation for a document (i.e. d), learning rate (i.e. σ), the input vector (i.e. $\vec{w}_i$), and winner neuron (i.e. $\vec{w}_s$) are used to formulate the modified winner neuron, shown in the equations 1.24 and 1.25 respectively.

$$\vec{w}_s' = \vec{w}_s + \sigma \cdot (\vec{w}_s - \vec{w}_i) \quad (1.24)$$

$$\vec{w}_s' = \vec{w}_s + v(i, s) \cdot \sigma \cdot (\vec{w}_s - \vec{w}_i) \quad (1.25)$$

1.9.3.4. EM-Algorithm (Model Based Clustering)

The Expectation Maximization (EM) algorithm helps to view the clustering with respect to the statistical point of view. Suppose, a set of documents $D = \{d_1, d_2, \dots, d_N\}$ is partitioned into k clusters with respect to the probability of a document for a cluster (i.e. q_{ic}). The probability vector $q_i = (q_{i1}, q_{i2}, \dots, q_{ik})$ of document for k clusters is represented as $\sum_{c=1}^k q_{ic} = 1$ [119]. The statistical assumption of the EM-algorithm is described in two steps.

Step-1. For each document, a cluster P_c is from the list of k clusters is picked with probability P_c .

Step-2. The words t of a document r generated according to cluster specific probability $p(t | P_c)$ distribution.

The applicability of these two steps aid to form a mixture model (shown in equation 1.26) and compute the probability of a document described through words $t = t_1, t_2, \dots, t_{n_i}$, where n_i referred to a number of words for document d_i .

$$p(t_1, t_2, \dots, t_{n_i}) = \sum_{c=1}^k q_c p(t_1, t_2, \dots, t_{n_i} | P_c) \quad (1.26)$$

Where P_c and q_c are referred as cluster and probability respectively.

1.9.3.5. Fuzzy c-means

The aim of fuzzy approach is to classify the objects (that is design patterns in our case) into clusters (that is a design pattern class in our case) according to their membership degrees between 0 and 1.

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1N} \\ x_{21} & - & - & - \\ - & - & - & - \\ x_{n1} & - & - & x_{nN} \end{bmatrix} \quad (1.27)$$

Like other clustering techniques, Fuzzy c-means (FCM) uses some notations to describe the functionality of the algorithm. First notation is of dataset shown in equation 1.27. Fuzzy clustering can be applied to numeric as well categorical data or a mixture of both. Equation 1.27 shows, dataset consists of some observations and each observation includes n measure variables which are grouped in form of n-dimensional column vector as shown in equation 1.28.

$$x_k = [x_{1k}, x_{2k}, \dots, x_{nk}]^T, x_k \in \mathbb{R} \quad (1.28)$$

$$X = \{x_k \mid k = 1, 2, \dots, N\} \quad (1.29)$$

Similarly, a dataset with N observations can be represented by equation 1.29. Fuzzy c-means (FCM) algorithm is based on the minimization of its objective function (J) which is shown in the equation 1.30 [120].

$$J(X; U, V) = \sum_{t=1}^c \sum_{k=1}^N (\mu_{tk})^m \|x_k - v_t\|_A^2 \quad (1.30)$$

$$U = [\mu_{tk}] \in M_{fc} \quad (1.31)$$

$$V = [v_1, v_2, \dots, v_c], v_i \in \mathbb{R} \quad (1.32)$$

Where U is the fuzzy partition matrix of dataset X shown in equation 1.31 and V is the vector of cluster prototypes that is centers shown in equation 1.32. Similarly, equation 33 is used to compute the squared inner-product distance norm with respect to equation 1.27 and equation 1.28.

$$D_{tkA}^2 = \|x_k - v_t\|_A^2 = (x_k - v_t)^T A (x_k - v_t) \quad (1.33)$$

$$m \in [1, \infty) \quad (1.34)$$

The fuzziness of the resulted clusters that is cost function can be determined by using equation 1.34 and it can be measured through the total variance of x_k from v_i . Subsequently, weighting component (i.e. m) and weight Matrix (i.e. A) are the two main parameters of the objective function J_m . The aim of m (i.e. $1 \leq m < \infty$) is to control the relative weights which are used for the squared error D_{ikA}^2 (equation 1.33). The increase in the value of m inclines to reduce the membership for the fuzziest state. The best value of m can be derived through an experimental investigation, which is usually in range $1.5 \leq m \leq 3.0$. The second parameter of the objective function J_m of Fuzzy c-means is the weight matrix A , which controls the cluster's shape in $\mathbb{R}$. There are several A -norms such as Euclidean ($A = I$), Diagonal ($A = D_y^{-1}$), and Mahalanobis ($A = C_y^{-1}$) Norms are available and each norm on $\mathbb{R}$ can be computed using the equation 1.33.

1.10. Performance Measures for the Evaluation of Automated Systems based on Text Categorization approach

The evaluation model consists of a set of measures used to assess the effectiveness of supervised and unsupervised learners which are recommended for an automated system. The aim of the supervised learner (i.e. Classifier) is to classify the text documents into their predefined classes based on their learning on the contents of all documents. In case of supervised learning, suppose, we have a training dataset $D = (d_1, d_2, \dots, d_N)$ of N documents which are classified into n labels and described as $L = \{l_1, l_2, \dots, l_n\}$. The supervised learning model is represented by the equation 1.35.

$$f : D \rightarrow L \quad f(d) = l \quad (1.35)$$

The performance of a supervised learning model is evaluated n separate dataset of documents which are not used during the training/learning process. The three widely used performance measure Precision, Recall, and F-measure are shown in the equations 1.36, 1.37, and 1.38 respectively,

$$\text{Precision} = \frac{\#\{\text{Total Related} \cap \text{Total Reterieved}\}}{\#\text{Total Reterieved}} \quad (1.36)$$

$$\text{Recall} = \frac{\#\{\text{Total Related} \cap \text{Total Reterieved}\}}{\#\text{Total Related}} \quad (1.37)$$

$$\text{F-measure} = \frac{2 * (\text{Precision} * \text{Recall})}{\text{Precision} + \text{Recall}} \quad (1.38)$$

Subsequently, the performance evaluation unsupervised learner of is not trivial as compared to precision and recall evaluation of supervised learner. The focus of an evaluation measure is to assess the separation of similar data into some ground truth set of classes or it might need some assumptions. Suppose, we have a partition set

$P = \{p_1, p_2, \dots, p_k\}$, its centroid $\mu_p = \frac{1}{|P|} \sum_{d \in P} \vec{t_d}$, and a distance measure “dist”, and manual labeling set $L = \{l_1, l_2, \dots, l_k\}$. The description of some widely-used performance measure are given as.

A. Mean Square Error (MSE)

The Mean Square Error (MSE) measure the quality of clustering effectively on the base of two assumptions a) The number of dimensions should be constant, and b) the number of clusters is constant. The equation 1.39 and 1.40 describe the MSE for clustering $P = \{p_1, p_2, \dots, p_k\}$ and a cluster p [64].

$$MSE(P) = \sum_{p \in P} MSE(p) \quad (1.39)$$

$$MSE(p) = \sum_{d \in P} dist(d, \mu_p)^2 \quad (1.40)$$

B. Purity Measure

The Purity and InversePurity are comparative measures which based on the two widely used performance measures named precision and recall. The equations 1.41-1.44 describe the Precision, Purity, Recall, and InversePurity respectively.

$$Precision(p, l) = \frac{|p \cap l|}{p} \quad (1.41)$$

$$Purity(P, L) = \sum_{p \in P} \frac{|p|}{|D|} \max_{l \in L} Precision(p, l) \quad (1.42)$$

$$Recall(p, l) = Precision(l, p) \quad (1.43)$$

$$InversePurity(P, L) = \sum_{l \in L} \frac{|l|}{|D|} \max_{p \in P} Recall(p, l) \quad (1.44)$$

C. Adjusted Random Index (ARI)

The Adjusted Random Index (ARI) function is used to evaluate the similarity between true labels (usually manually assigned on the base of ground truth and predicted labels via a clustering algorithm. The equations 1.45 and 1.46 are used to compute the Random Index (RI) and Adjusted Random Index (ARI).

$$RI = \frac{x + y}{CL_2^{N_{samples}}} \quad (1.45)$$

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]} \quad (1.46)$$

Where the term CL and K referred as true and predicted label assigned to documents. Subsequently, the terms x and y are referred as same and different set of pair of elements in CL and K. Finally, the term $CL_2^{N_{samples}}$ is referred as a total number of possible pairs.

D. Adjusted Mutual Information (AMI) and Normalized Mutual Information (NMI)

The Mutual Information (MI) is a function which describes the closeness agreement between true and predicted labeled documents. The Adjusted Mutual Information (AMI) and Normalized Mutual Information (NMI) are the two different normalized variants of MI. Suppose, U and V are the true and predicted labels of N documents. The equations 1.47-1.49 are used to compute the amount of uncertainty (i.e. Entropy) of U and V, and Mutual Information (MI).

$$H(U) = \sum_{i=1}^{|U|} P(i) \log(P(i)) \quad (1.47)$$

$$H(V) = \sum_{j=1}^{|V|} P'(j) \log(P'(j)) \quad (1.48)$$

$$MI(U, V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} P(i, j) \log\left(\frac{P(i, j)}{P(i)P'(j)}\right) \quad (1.49)$$

Subsequently, the equations 1.50 and 1.51 are used to compute the NMI and AMI respectively.

$$NMI(U, V) = \frac{MI(U, V)}{\sqrt{H(U)H(V)}} \quad (1.50)$$

$$AMI = \frac{MI - E(MI)}{\max(H(U), H(V)) - E[MI]} \quad (1.51)$$

The term $E[MI(U, V)]$ is computed through equation 1.52.

$$E[MI(U, V)] = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \sum_{n_{ij}=(x_i+y_j-N)}^{\min(x_i, y_j)} \frac{n_{ij}}{N} \log\left(\frac{N \cdot n_{ij}}{x_i y_j}\right) \frac{x_i! y_j! (N - x_i)! (N - y_j)!}{N! n_{ij}! (x_i - n_{ij})! (y_j - n_{ij})!} \quad (1.52)$$

E. Homogeneity, Completeness and V-measure

The Rosenberg and Hirschberg [121] describe two conditional entropy based performance measures named Homogeneity and Completeness. The Homogeneity and Completeness measures aim that each cluster has members of a single class and all members of any given class are grouped into the same cluster respectively. The equations 1.53 and 1.54 are used to compute the Homogeneity and Completeness of a cluster respectively.

$$h = 1 - \frac{H(C|K)}{H(C)} \quad (1.53)$$

$$c = 1 - \frac{H(K|C)}{H(K)} \quad (1.54)$$

Where the $H(C|K)$ and $H(K|C)$ are referred as the conditional entropy of a class and cluster respectively, while $H(C)$ and $H(K)$ are referred as the entropy of a class and cluster respectively. The entropy and conditional entropy of a class can be computed through the equations 1.55 and 1.56 respectively.

$$H(C) = - \sum_{c=1}^{|C|} \frac{n_c}{n} \log\left(\frac{n_c}{n}\right) \quad (1.55)$$

$$H(C | K) = - \sum_{c=1}^{|C|} \sum_{k=1}^{|K|} \frac{n_{c,k}}{n} \cdot \log\left(\frac{n_{c,k}}{n_k}\right) \quad (1.56)$$

The equation 1.57 is used to compute the harmonic means of h (i.e. Homogeneity) and c (i.e. Completeness) and referred as a V-measure.

$$v = 2 * \frac{h * c}{h + c} \quad (1.57)$$

F. Silhouette Coefficient

The Silhouette Coefficient measure is used to when the objects (i.e. Documents) have no true label assignment (i.e. No ground truth labels). The highest value of the Silhouette Coefficient measure indicates the better partitioning of objects into right group. The Silhouette Coefficient can be computed using the equation 1.58.

$$s = \frac{x - y}{\max(x, y)} \quad (1.58)$$

2.1. Literature Review

Software design patterns are considered as an important way to get the collective wisdom of software engineering (SE) community and aid to give a tool for effective software engineering. However, the organization and documentation of the collective wisdom of SE community are still in a continual process [1, 2]. The software engineering community is making their efforts to introduce and publish new patterns to resolve the recurring design problems of different domains. In the 1970s, the famous mathematician and architect Alexander introduced the concept of patterns and pattern languages [3, 4]. The same concept is adopted by many researchers to introduce the design patterns in different domains such as object-oriented development [5-9], Fault-Tolerant telecommunication [10], communication software [11], web designing [13], education [122], Human Computer Interaction [14], real-time system [12], database management, and network security [15]. The books, verbal communication between developers, online repositories^{1-3,5-15}, question answering systems¹⁶, and literature of published pattern at different venues are the common sources of software design patterns related to different domains. The developer and researcher community of each domain usually target the compiler and other developers of same community when they describe and categorize the patterns with respect to different aspects. For example, in the domain of object-oriented development, Gamma et al describe 23 design patterns and categorized them into three categories such as creational, structural, and behavioral depends on the relationship between classes and objects [5]. Subsequently, Gamma et al describes each pattern into two sections named Problem Definition Domain and Solution Domain, which are further categorized into sub-sections. For example, intent, motivation, and applicability are the sub-section of the Problem Definition Domain of Gamma's design patterns. Unlike a compiler, the developers (i.e. Audience of the corresponding domain) not only need to understand the functionality of any source code, but also required more information to know the robustness and fragility of an implemented design. The well-defined documentation of efforts of developers to resolve the design problems can aid other developers for effective communication. The concept design pattern is adopted and basis on certain objectives which include effective communication between developers, software productivity, and to discuss the declined quality of software. The effective communication between the developers²⁰ depends on the certain features of cataloged pattern such as Clarity, Consistency, Completeness, Conciseness, Concreteness, Courtesy, and Concreteness, which can be affected by several factors such as.

1. The interest and experience of the developers who introduce a new pattern catalog, which might be a cause for the introduction of more than one catalogs in the same domain. For example, in object-oriented development, [5-9] have addressed aspects and used different techniques (new or borrowed) to resolve a recurring design problem.
2. The developers follow a state-of-art to describe the pattern in certain section basis on their interest either in same or different domains. For example, PLoP is the best venue to analyze the state-of-the-art which are followed to catalog the patterns.

Alexander has introduced a template which consists of three sections named Context, Problem description, and a corresponding Solution [3-4]. Subsequently, the building components of each section are varied across the pattern

catalog depend on the developer's experience and knowledge in the response of handling design problem(s). For example, in the domain of object-oriented development Gamma et al. [5] have introduced 23 design patterns and described each pattern with a set of consistent sections named intent, motivation, applicability, structure, participants, collaboration, consequences, implementation, sample code, and related patterns. Moreover, in the domain of Service Oriented Architecture (SOA)⁵, the authors have described each pattern with a set of 6 sections named Problem, Solution, Application, Implementation, Principles, and architecture.

Table 2.1. Summary of domains and related frequently used design pattern collections

Patterns source	Sections	Problem Definition Sections	Solution Sections
Gamma et al [5]	Intent, motivation, applicability, structure, participants, collaboration, consequences, implementation, sample code, and related patterns	Intent, motivation, applicability	Structure, participants, consequences, implementation, sample code, related patterns
Erl ⁵	Problem, Solution, Application, Implementation, Principles, and architecture	Problem, Principles	Solution, Application, Implementation, and architecture
Duyne et al. [13]	Patten title, Problem statement, Background information, forces, solutions, related pattern	Patten title, Problem statement, Background information, forces	Solutions, related pattern
Yahoo patterns ¹³	What Problem does this solve, when to use this pattern, what's the solution, Why use this pattern, accessibility	What Problem does this solve, when to use this pattern, Why use this pattern	What's the solution, accessibility
Welie ⁹	Problem, Use when, how, why, solution, related examples	Problem, Use when, how, why	Solution, related examples
Tidwel ¹²	What, use when, why, how, examples, Related examples	What, use when, why	What, use when, why
Baraki et al [15]	Name, Intent, Motivation, forces and Context, Solution, Consequences, elated Patterns	Name, Intent, Motivation, forces and Context	Solution, Consequences, elated Patterns
Douglass [15]	Abstract, Problem, Pattern Structure, Colloboration Roles, Consequences, Implementing strategies, Related Patterns	Abstract, Problem,	Pattern Structure, Colloboration Roles, Consequences, Implementing Stratetigies, Related Patterns

The description of the pattern structure depends on 1) pattern's domain, 2) developer's interest and knowledge, and 3) developers' awareness about the pattern catalog of the same domain. However, the pattern's structure of any domain can be looked in their roots and described into two broad sections called 1) Problem Definition Domain, and 2) Solution Domain. The Problem Definition Domain is similar to the context and problem description sections, while the solution Domain section is related solution section of Alexandar's given template. The list of domains, they're widely used pattern catalogs, the sections describe the patterns, and their grouping into two broad categories is shown in Table 2.1. The information which is described in Table 2.1 indicate that books and other proceedings are considered as an important source of published design pattern in different domain. Consequently, it becomes difficult for developers to understand all aspects which are addressed by other developers to describe patterns on the basis of their experience. Subsequently, online pattern repositories in different domain, such as object-oriented development^{6,7}, service oriented architecture⁵, human computer interaction^{8,12,15}, web design⁹, pattern languages³, education¹¹, personal website¹⁰ and pattern repositories of hybrid domains¹.

Beside online pattern repositories, there are several online libraries/groups are available to host certain pattern catalogs of different domains such as EuroPlop², Yahoo¹³, Wiki⁶, and Rising-Patterns Almanac 2000¹. The continuous modification due to the emergence of the new pattern in online repositories and libraries, the awareness of patterns for the selection of right one become hard.

Implications:

- The books, proceedings, and online repositories and libraries/groups are the well-known source to publish and find patterns to resolve the recurring design problem.
- The descriptions of design patterns depend on their domain, developer's interest and knowledge, and the procedure which is followed to cataloged the patterns.
- The online repositories/libraries are steadily growing in the number of new patterns or spoiled pattern which is derived from the alternative solutions of the same design problem.

2.2. Design Pattern Classification

There are several venues which can be used by the developer to search right design pattern(s) in order to resolve a design problem such as books, proceedings of conferences/journal, online repositories, and online libraries/groups. From the Table 2.1, it can be described that the structure of patterns (i.e. Into the number of sections) in different domains and the classification scheme to group the similar pattern is highly related to the developer's knowledge, experience, and interest. For example, Gamma et al [5], Cod et al [7], and Pree et al [6]. present their own classification scheme to group the design patterns with respect to certain aspects in the domain of object-oriented development. Gamma et al [5] considers the instantiation, composition, and communication of class objects, and introduce the 23 design patterns as a solution for most commonly occurred recurrence design problem, and group them into three broad categories named Creational, Structural, and Behavioral. Moreover, Coad [7] consider the stereotypical responsibilities and scenario interactions aspects in order to investigate 31 patterns employed by developer's of five well-known industry applications and group them into four categories named Transaction, Aggregate, Plan, and Interaction. The name of industrial application are; 1) Connie's Convenience Store, 2) Wally's

Warehouse, 3) Ilie's Order Center, 4) Dani's Divrters, and 5) Andi's Autopilot. Subsequently, W. Pree focuses on the Recursive Structures and Abstract Coupling between objects to classify the pattern collection. The template used to describe the structure of a pattern is related to the domain, developer's knowledge and experience, and the technique (new or borrowed) to describe the solution. From the Table 2.1, we can also extract information that in the domain of design pattern classification, existing efforts of researchers can be summarized with respect 1) Problem-based classification and 2) solution-based classification.

2.2.1. Problem-based Classification of Design Patterns

In the problem-based classification of design patterns, the authors mostly used the context and intent of design problems and applicability of design patterns to organize the patterns into high-level categories. For example, Gammal et al. [5] introduces 23 design patterns and organized them into 3 high-level categories (Creational, Structural, and behavioral) on the base of the problem description.

The patterns of same categories have relevancy to each other up to a certain extent, however, it cannot be replaced as a right choice with any other pattern of the same category. Consequently, Problem Definition Domain (Table 2.1) is mostly used by authors to classify the pattern into high-level categories. The components of the Problem Definition Domain section are described in the form of narrative text without the constraint of borrowed (or new) technique used to describe the solution of pattern. The problem-based classification schemes, including their domain and high-level categories, are summarized in Table 2.2.

Besides well-defined organization of design pattern with respect to predefined classification schemes (shown in Table 2.2), there are certain domains where patterns have been introduced without any classification schemes such as education, the pattern used by developers during the software development life cycle, cooking, embedded systems, enterprise applications, and automated system. In the same domain, the existence of several pattern catalogs affects the searching of the right patterns to resolve a design problem. The organization of published design pattern (without any classification scheme) either in the form of books and proceeding, or hosted on online catalog/library is still required.

2.2.2. Solution-based Classification of Design Patterns

The Software Engineering (SE) research community has made little effort in the domain of solution-based classification of design patterns into the appropriate number of design pattern classes. However, they have used solution domain of design patterns in designing and implementation of certain tools which can aid to identify or detect the pattern instances in the given source code. The list of design pattern detection tools in different domains

Table 2.2. Summary of problem-based the design pattern collections and related information

Author	Domain	Total Patterns	High-Level Categories
Gamma et al. [5]	Object-oriented Design Problems	23	Creational, Structural and Behavioral
Pree [6]	Object-oriented Design Problems	23	Pree's classification scheme relies on Recursive Structures and Abstract Coupling
Coad et al. [7]	Object-oriented Design Problems	31	Transaction, Aggregate, Plan and Interaction
Tichy [8]	Software Design Problems	100	Decoupling, Variant Management, State Handling, Control, Virtual Machines, Convenience, Compound, Concurrency and Distribution
Rising [11]	Application Type	1200	Accounting, Hypermedia, Trading, C++ Idioms, System Modeling, Air Defense, Interactive, Database, Persistence and Trading
Douglass [12]	Real Time	34	Architectural Design, Safety and Reliability, Memory, Concurrency Resources and Distribution
Schumacher [14]	Security Patterns	46	SACA (System Access and Control Architecture), ACM (Access Control Model), IA (Identification and Authentication), OSAC (Operating System Access Control), SIA (Security Internet Application), FA (Firewall Architecture), ESRM (Enterprise Security and Risk Management) and Accounting
Duyne et al [13]	Web Designing	90	Site Genres, Creating a navigation Framework, Creating a Powerful Homepage, Writing and Managing Content, Building Trust and Credibility, Basic E-Commerce, Advanced E-Commerce, Helping Customers complete Tasks, Designing Effective Page Layouts, Making Site Search Fast and relevant, and speeding Up Yout Site.
Nilsson [51]	User interface for mobile applications	15	Utilizing Screen Space, interaction Mechanism, Design at Large
Lano and Kolahdouz [52]	Model Driven Development	24	Modularization, Optimization, Model-to-Text, Expressiveness
Baraki et al [16]	Interdisciplinary design patterns	8	Classified on the base Socio-technical requirements

of software engineering has been given in section 2.3.1. In the domain of object-oriented development, researchers have made their efforts to organize patterns by investigating the solutions of design patterns. Zimmer [29] proposed an approach to investigate the relationship between 20 Gang-of-Four (GoF) design patterns and organize them accordingly. There are three aims of Zimmer's work; 1) to classify the patterns on the base of relationship exists between them, 2) to introduce a new pattern which can be derived from the generalization process of many design patterns, and 3) to describe the structure of design pattern on several layers. According to Zimmer approach, the relationship between design patterns can be represented as (X, Y) pairs of design patterns. From the Figure 2.1, Zimmer extracts three types of relationships between design patterns such as 1) X used Y in its Solution, 2) X is similar to Y, and 3) X can be combined with Y. Subsequently, Zimmer extracts re-organize the patterns with three architectural layers such as 1) Application domain, 2), Typical software problem, and 3) Basic design pattern and techniques, shown in Figure 2.1.

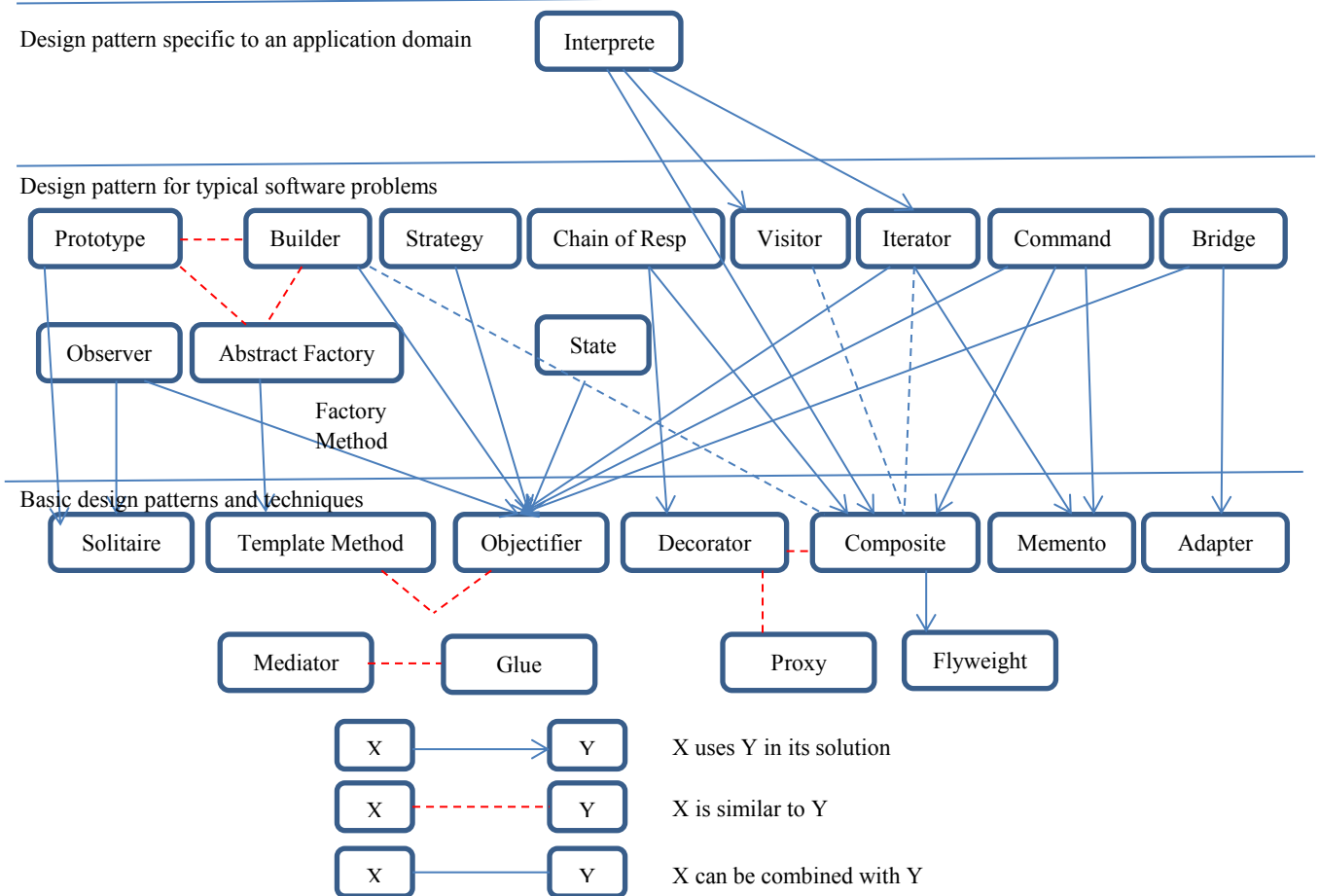


Figure 2.1. Zimmer's proposed solution-based technique to organize the design patterns

Kim and Han [30] target the pattern classification through clustering with high precision rather through the facets [30]. Kim's proposed clustering algorithm work at two-course levels (that are the function and structure) classification of patterns. In their clustering algorithm, the structure of a pattern is transformed into the ordered pair

P (shown in the equation 2.1) by considering the possible relationships (such as Association(A), Generalization(G), Aggregation(E), Composite(C), Dependency(D), and Realization(R)) shown in the equation 2.2) between them.

$$P = \{R_k(i, j) \mid (i, j) \in R, i \in C, j \in C, 1 \leq k \leq n\} \quad (2.1)$$

$$R = \{A, G, E, C, D, R\} \quad (2.2)$$

The ordered pair of a variant P_v is transformed according to described relations among classes and find with respect the ordered pair of foundation patterns P_f . For example, the canonical solution of proxy pattern and order pairs according to clustering algorithm is shown in Figure 2.2.

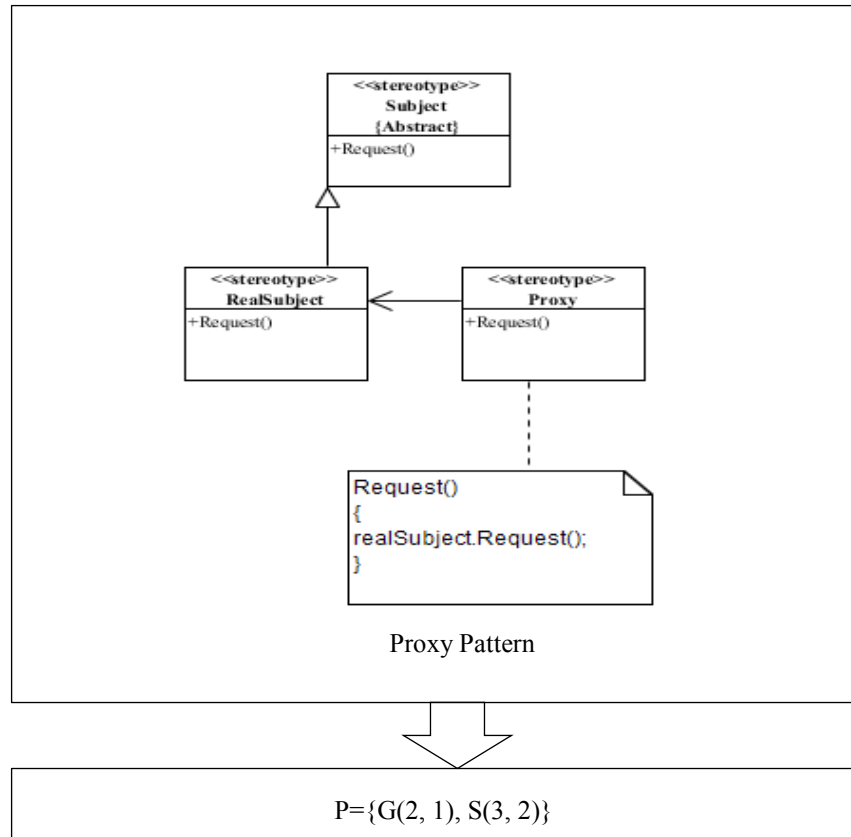


Figure 2.2. Step-1 of Clustering approach (a solution-based technique) to organize the design patterns

Subsequently, a given alternative solution (or a variant) with ordered pair is shown in Figure 2.3.

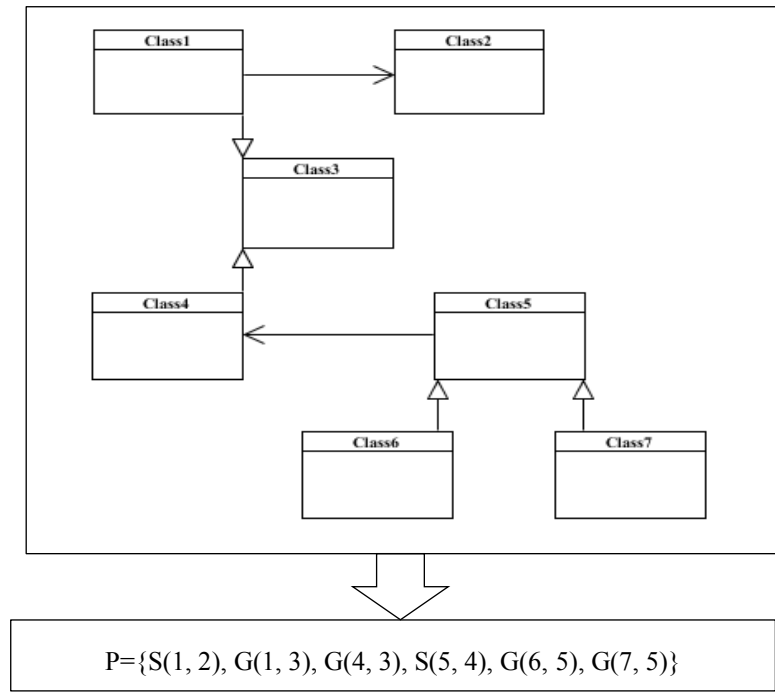


Figure 2.3. Step-2 of Clustering approach (a solution-based technique) to generate the ordered pair of design patterns

The second step of the clustering algorithm is to transform the ordered pair of pattern and any given alternative solution according the relationship which exists between their components. The transformation steps of clustering algorithm are depicted in Figure 2.4.

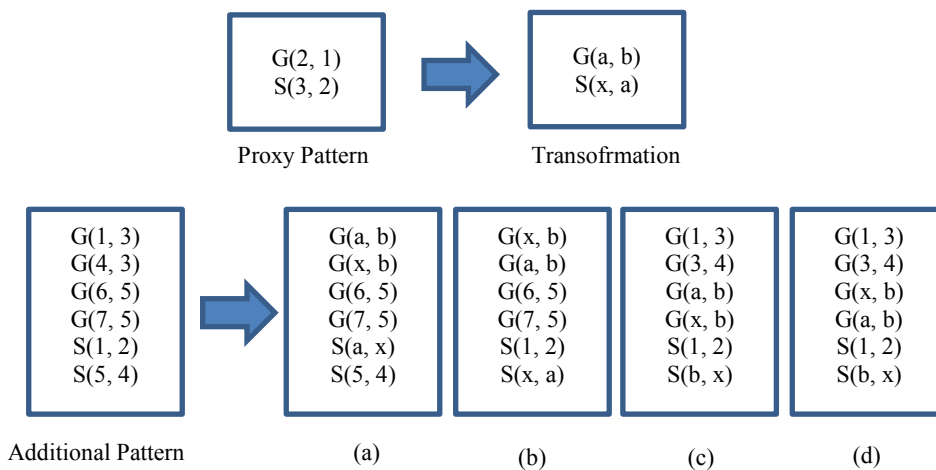


Figure 2.4. Step-3 of Clustering approach (a solution-based technique) to transform the ordered pair of original and variants of design patterns

The closeness between two ordered pairs P_v and P_f aid to identify and cluster the similar patterns by satisfying the equation 2.3.

$$P_f \subset P_v \quad (2.3)$$

Implications:

- In the same domain, there are a number of problem-based classification schemes, which have been introduced on the basis of different aspects to organize the patterns into appropriate number of classes. For example, Gamma et al. [5], Pree [6], Coad et al. [7] in the domain of object-oriented development, and Duyne et al. [13], Baraki et al. [16], Nilsson [51], and Lano and Kolahdouz [52] in the domain of Human Computer Interaction (HCI) have considered different aspects of design problems and introduced classification schemes to organize the patterns accordingly.
- There are several repositories exists which either host design pattern catalogs of similar entries such as PLoP (Pattern Language of Programming) or hosting patterns without a classification scheme such as Rising - Patterns Almanac 2000¹. The researchers have coined the term meta-pattern to discuss commonalities between pattern catalogs. The usage of the meta-pattern is limited to the domain.
- The solution-based classification of design patterns has been introduced to organized the patterns, however, there are few limitations, such as, 1) domain specific, 2) limited to techniques for which approach is applied, and 3) applicability issue across the pattern catalogs of the same domain.

2.3. Design Pattern Selection

A well-defined classification scheme can aid to find the group of relevant patterns and make easy the process adapted to search the patterns for the given design problem. However, the selection or identification of right design pattern from the list of relevant patterns is still an issue which has been addressed by many researchers in different domains. Before starting to investigate the existing efforts of researchers in the domain of design pattern selection, we consider the following two quotes from well-known researchers in the domain of design patterns. The first quote is taken from Sommerville [24], and the second quote is from Gamma et al. [5].

Quote-1. “Only experienced software engineers who have a deep knowledge of patterns can use them effectively. These developers can recognize generic situations where a pattern can be applied. Inexperienced programmers, even if they have read the pattern books, will always find it hard to decide whether they can reuse a pattern or need to develop a special-purpose solution.”

Quote-2. “With more than 20 design patterns in the catalog to choose from, it might be hard to find the one that addresses a particular design problem, especially if the catalog is new and unfamiliar to you.”

From these two quotes, it can be clearly determined that selection of right design pattern is an issue, especially, when an inexperienced developer in the domain where more than 10 design patterns are exist. Later, these issues have been addressed in consecutive studies [22, 26, 31, 34-39]. Besides the support for the pattern selection, creation of a rank list of patterns and recommendation of appropriate patterns is also the important features to resolve the

design pattern selection process. Researchers have made their efforts to introduce several tools, recommendation systems, and prototype to automate the design pattern selection process using different approaches. With respect to applied approaches, we can summarize the existing efforts of researchers into three categories.

2.3.1. UML-based Automated Systems

D-K Kim and C.E. Khawand [34] describe an approach to select the design pattern by specifying its problem domain. They reported that problem domain of design patterns can help to describe the known problems in the context. The authors used Role-Based Modeling Language (RBML) as UML based pattern specification language to describe the Meta-model to formalize the design patterns. The structure of a Meta-Model is formulated through the class and collaboration diagrams to describe the structural and behavioral aspects of software and detect a design pattern. The structural similarity and scalability in terms of a number of design patterns and its variation are the main constraints in implementation of UML-based approaches. Due to the structural similarity between design patterns, a Meta-model cannot specify the problem domain of design patterns. For example, in the case of Gamma et al. design pattern catalog, the Meta-model for Strategy pattern looks closer to Meta-model of the State design pattern, however, their goals are different. Subsequently, there are some design patterns such as Prototype, whose specification through Meta-model cannot be presented [34].

In their study, Hsueh et al. [35] also follow the UML-based approach, describe a case study with a limited number of design patterns, and consider four perspectives such as activity-view, requirement-view, problem-view, and tradeoff-view to find a systematic way to select a design pattern. They use UML notations to describe the design problems by incorporating the non-functional requirements beside functional requirements. The design pattern perspectives include, 1) the Activity-View, to define how a design pattern can help a design activity, 2) Requirement-View, to explore how a design pattern can improve the non-functional requirements, 3) Problem-View, to investigate how a design pattern can help to resolve the design problems and prevent exceptions, and 4) Tradeoff-View, to see how a design pattern can determine the design conflict. The number of limited design patterns in a case study and selection of design pattern through a framework rather than automation are the main constraints [35].

In the domain of object-oriented development, researchers have introduced several tools to detect the pattern instance in the given source rather than to select a pattern for the given design problem. These tools can aid developers to find the usage intensity of design patterns and their association with the internal (e.g. Coupling, cohesion, and complexity) and external quality attributes (e.g. Stability, understandability, flexibility, reusability, and effectiveness). Moreover, through these tools, a developer can investigate the declined quality of software. The common thing among all design pattern detection tools is the usage of UML (i.e. Class diagram notations) to present the structure of patterns. Moreover, the techniques which are used to specify the patterns are different. The list of some commonly used design pattern detection tools is shown in Table 2.3 [36].

Numerous design pattern identification tools have been introduced and implemented in empirical studies. The performance of these tools always remains a debatable issue in software engineering research community due to certain factors, such as,

1. These tools can find the limited number of design patterns, for example DeMIMA [123], MARRPLE [125], and DP-Miner [126] can find 13, 3 and 6 design patterns.
2. The precision and recall of these tools either is unknown or unsatisfactory, such as, the authors of FUJBA [127], Pinot [128], and Ptidej [129] discuss their implementation without their precision and recall.
3. The identification of one or more than one role involved in a design motif [55].

Table 2.3. Summary of problem-based the design pattern collections and related information

Tool	Number of design Patterns	Pattern Specification Approach	URL
DeMIMA [123]	12	DeMIMA (Design Motif Identification Multilayered Approach) is a multi-layered approach used to identify idioms (referred as a relationship among classes and design motif) between design patterns and source code. The first layer named “Source Code Model” is used to describe the source code in the form of a meta-model. The second layer named “Idiom-level Model” is used to describe the certain idioms in the source code.	
Tsantalis et al.’s tool [124]	12	The Tsantalis et al.’s tool is used to 1) presents the graph of variants and canonical solution of design patterns, and 2) to identify the pattern versions (variants) with respect to the canonical solution of the design patterns on the base score similarity between the graph vertices.	https://users.encs.concordia.ca/~nikolaos/pattern_detection.html
MARPLE [125]	3	The MARPLE (Metrics and Architecture Reconstruction Plug-in for Eclipse) which consist of the set of modules to perform individual tasks. The main modules of MARPLE are classified into three categories 1) Data Extraction Modules (used Abstract Syntax Tree (AST), 2) Information Analysis (Design Patterns Detetcion (DPD) and Software Architecture Reconstruction (SAR), and 3) output generation (use of XML files, diagrams, and metrics).	

DP-Miner [126]	6	DP-Miner tool is used to discover patterns from the source code of the system which is represented as a matrix. The value of a cell of a matrix represents the relationship between the classes. The DP-Miner tool investigate the structural, behavioral and semantic aspects of the pattern recovery.	
FUJABA [127]	23	The open source CASE tool named Fujaba(aka Fujaba Tool Suit) is based on model-based software engineering and re-engineering, which aims to identify the patterns in exist in the source code.	http://www.fujaba.de/
Ptidej [128]	19	The Pattern Trace Identification, Detection, and Enhancement in Java (PTIDEJ) project aim to develop a tool suite, which consist of PADL (Pattern and Abstract level Description Language), POM (Primitives, Operators, Metrics), and UI (User Interface) modules. The main objective of the PTIDEJ tool suite is to evaluate the software quality through detection of patterns at language, design and at architectural levels.	http://www.yann-gael.gueheneuc.net/Work/Research/Ptidej/Download/
Pinot [129]	17	The PINOT (Pattern Inference and recovery Tool) is used to discover the design patterns related to the structure or behavior of a system.	http://web.cs.ucdavis.edu/~shini/research/pinot/
WOP [130]	4	The Web of Patterns (WOP) tools is based on OWL (Web Ontology Language) to formulate the design patterns and identify the patterns through a matching ontology mechanism. The WOP tool aid to detect the patterns by accessing the pattern definitions.	
DPJF [131]	23	The DPJF (Detection of Patterns by Joining Forces) tool is used to detect the occurrence of design patterns in the Java programs. The DPJF tools aim at forwarders and Decouplers factors to detect the patterns.	http://sewiki.iai.uni-bonn.de/research/dpd/dpjf/start
Finder [132]	23	The finder tool work in four stages and aid to detect the design patterns. The four stages of finder tool are	https://wiki.eecs.yorku.ca/project/dpd/finder

- 1) Static Fact Extraction, 2) Field Type Probing, 3) Static Facts Refinement and Integration, and 4) Design Pattern Instance Recovery.
-

Implications:

- UML-based automated systems and tools require the formal specification of canonical solutions of patterns and their variants. For example, Tsantalis et al.'s tool [124] needs to transform the canonical and variant solutions into graphs before identification of any design pattern.
- UML-based automated system targets the solution domain of patterns and work in the defined context, which makes their applicability across the pattern catalogs of the same domain and the technique which is followed to formulate the patterns. For example, Tsantalis et al.'s tool [124], Pinot [129], and meta-modeling approach [34] are limited to GoF design pattern catalogs.
- In a recent study Elish et al. [55] reported the issue of the number of identifying patterns through UML-based automated system and tools and conduct an empirical study to investigate the fault density of many design patterns [55].
- Besides the number of target design patterns, the precision and recall of design pattern detection technique always remain an issue to evaluate the software quality. The researchers have empirically investigated the relationship between the design pattern and quality attributes.

2.3.2. Ontology-based Automated Systems

Hasso and Carlson consider the problem definition part of design patterns and perform semantic analysis of sentences and on the base of words meaning to suggest the classification of design patterns. In their study, they perform semantic analysis using linguistic theory and consider the meaning of verbs for classification of design patterns. They claim that their proposed approach helps to select a right design pattern, which becomes in doubt due to lack of its evaluation and automation [28]. In two consecutive studies Khoury et al. And Blomqvist addresses the automation of selection of right design pattern via their proposed ontology based automated systems. For example, In their study, Khoury et al., introduce an ontological interface depicted in Ontology Web Language (OWL) to aid developers to discover an eligible set of security patterns. The proposed ontological interface implements a mapping between security requirements on one side and security bugs, security errors and threats model on the other side of context [39]. Subsequently, the Blomqvist [38] proposed a semi-automatic ontology construction approach named OntoCase, which rely on the use of ontology pattern to rank and select the right design pattern. In this work, the author defines criteria which include, relation matching, class matching, density, centrality and semantic similarity to rank the design patterns. In these studies, evaluation and automation factors are targeted. Subsequently, the computation time and specification in the case of the complex design pattern collections are the still main issues in the evaluation of the proposed approach [25]. In this regard, Birukou [134] proposed a multi-agent system to recommend Implicit Culture patterns (i.e. IC-Patterns). The proposed system which aid to select the patterns using the ICF (Implicit Culture Framework), which makes recommendation of patterns on the base of user's past actions. Subsequently, in order to find relevant patterns, the proposed system also support the CBR method and traditional

Information Retrieval (IR) Process. The IC-Patterns are the combination of client side (web-based user interfaces) and server side (multi-agent platform) components. The user's design problem is considered as query/input of the proposed system which includes the description of the problem (e.g. The use of keywords) and project (e.g. Project size). Subsequently, the Lucene IR library is used for the indexing. However, to achieve form aware search, machine processable description of the pattern is required. The lack of existence of formal description in software design domains and evaluation reports is the common constraints to fulfill the aims of these studies. In a recent study, Elizondo et al. [26] proposed an approach based on knowledge representation and information extraction to analyze the architectural pattern description with respect to the specific quality attributes. In order to facilitate inexperienced architects, the proposed approach is automated using a computable model which can be referred as a prototype tool. The proposed approach claims the applicability to focus on the quality attribute-oriented pattern selection and lack of limitation on the use of a pre-defined and closed pattern repository.

There are numerous recommendation systems which have been introduced by researchers in order to suggest the right pattern(s). For example, Gomes [133] applied Case-Based Reasoning (CBR) approach to automate the selection of design patterns. The aim of the proposed CBR-based framework is to select the patterns which can resolve a design problem and generate the new design. The proposed approach is implemented in the CASE tool REBUILDER, which can provide new CBR-based functionality. The cases of proposed CBR-based framework are described in the form of UML class diagram, which can be retrieved and updated with the support of REBUILDER. The retrieval of cases depends on the similarity between 1) the structure of a pattern and the current design, and 2) semantic distance of class and role names in the pattern. Birukou [134] proposed a multi-agent system to recommend Implicit Culture patterns (i.e. IC-Patterns). The proposed system aids to select the patterns using the ICF (Implicit Culture Framework), which makes the recommendation of patterns on the base of user's past actions. Subsequently, in order to find relevant patterns, the proposed system also support the CBR method and traditional Information Retrieval (IR) Process. The IC-Patterns are a combination of client side (web based user interfaces) and server side (multi-agent platform) components. The user's design problem is considered as query/input of the proposed system which includes the description of the problem (e.g. The use of keywords) and project (e.g. Project size). Subsequently, the Lucene IR library is used for the indexing. However, in order to achieve a form aware search, machine processable description of pattern is required.

Subsequently, in the domain of E-Business project development, the Hang et al. [135] introduce a new cluster analysis method which includes a pattern clustering algorithm (PCKM) and a collaborative filtering algorithm (BARPC). The proposed method is introduced with aim to bridge the Requirements Engineering (RE) patterns and need development of the e-business system. The proposed recommendation system computes the contribution and the similarity degree of patterns, and correlation between project saturation and the applied design pattern.

Implications:

- Ontology-based automated systems either require semi or fully formal specification of design problems and the patterns.

- Unlike UML-based automated system which mostly target the solution domain of patterns and work in the defined context, the research community of ontology-based automated systems show their interest in both problem domain and the solution domain of design pattern. For example, Hasso and Carlson [28] consider the problem definition part of design patterns to perform the semantic analysis of sentences rather than the solution of patterns [28], while Gomes [133] target the solution domain of patterns.
- Like UML-based automated system, ontology-based automated systems also work within the defined context and are not applicable across the catalogs.

2.3.3. Text Categorization based Automated Systems

In the last two decades, the software engineering research community started to show their interest for the use of categorization approach in certain domains where mostly unstructured data are available [57]. For example, spam E-mail filtering [40], web page classification [41], sentiment analysis [42], author identification [43], and the design pattern selection [31-33]. There are two common assumptions which are followed for the use of the text categorization approach in designing and implementing an automated system.

1. Text categorization approach can aid to classify the unstructured text, but, some sort of pre-processing activities are required.
2. Text categorization approach can cluster similar unstructured texts based on text relevancy.

Besides these two assumptions, a set of well-defined measures is also required to evaluate the effectiveness of a proposed automated system. In the domain design pattern selection, Hasheminejad and Galileo [31] follows the text categorization approach and describe a two-step automated method to select design pattern with respect to the degree of similarity between the description of design problems and problem definitions of the retrieved design patterns. The two steps of the proposed method are Learning Design Patterns and Design Pattern Retrieval. In the first step, a classifier is learned for each design pattern class, while in the second step, a candidate design class similar to a design problem is determined based on text categorization. Subsequently, a right design pattern is selected from the design pattern class suggested to the developers. This work has a few shortcomings. First, it needs a classifier for each design pattern class which seems to be inadequate for design pattern catalog, which has been either classified into many categories or obtained as interdisciplinary design patterns [5, 6]. Second, the power of classifiers is highly related to sample size and sparse density, which are not considered in this work. Third, the research community only uses Document Frequency (DF) technique for feature selection and evaluate the increase in performance of the proposed method. The Document Frequency (DF) is a simple technique and it cannot overcome the multi-class problem in feature selection process [47]. In a recent study, Wu et al. [136] proposed a new ensemble method named ForesTexter to overcome the imbalanced text categorization problem and benchmark the results on several widely-used imbalanced datasets. Subsequently, the authors comparatively investigate the proposed ForesTexter method with standard random forest and different variants of SVM algorithms.

Implications:

- Text categorization based automated systems of certain domains are exploited through supervised learning techniques. In a recent study, Delgado et al. [46] have evaluated the performance of 179 classifiers of 17 different families on 121 datasets retrieved from the UCL database to solve the real world classification problem. However, they cannot decide which classifier is better than others. Consequently, the recommendation of classifiers in the design structure of an automated system is the main threat for the applicability of the proposed method.
- In the domain of design pattern selection, the use of a supervised learner in text categorization based automated system have a few limitations.
 - ❖ The training of the supervised learner for an individual design pattern class may increase the overhead in terms of computing effort. For example, in case of Booch [25], Dwyer [3], and some online repositories for design pattern catalogs^{1,2}, where patterns have been classified into several classes, it becomes difficult to train a huge number of supervised learner.
 - ❖ The feature selection methods are recommended as a pre-processing activity, however, the multi-class problem remain an issue to affect the precise learning of a classifier. In a recent study, Uysal et al.[47] and Hussain et al. [33] target the discriminative power of feature selection method and addressed the multi-class problem effect on the performance of classifiers. In their study, Uysal et al., present an improved Global Feature Selection Scheme (IGFSS) and make it functional via four feature selection method. Subsequently, the authors reported the effectiveness of the proposed scheme in the context of classification decision via supervised learners. Hussain et al, leverage the IGFSS scheme and introduce a new feature selection method named Ensemble-IG to target the multi-class problem in the context of unsupervised learning.
- Unlike UML-based and ontology-based automated systems used for the identification of design patterns, the applicability of text categorization based automated system is not limited to the predefined context and domain.

2.4. Text Categorization and Applications

Text mining is the equivalent term of textual analysis or text data mining, which aims to retrieve meaningful and interesting information from unstructured texts. The text mining is an interdisciplinary domain of data mining, information retrieval, statistics, computational linguistics and machine learning. The advancement in correlated fields (such as the generation of topic models through statistical analysis in the information retrieval domain), and availability of unstructured software engineering (SE) data [57] motivate the SE research community to use text mining to solve certain information retrieval problems. The widely used topic models such as LSI (Latent Semantic Indexing) [93, 94] and LDA (latent Dirichlet allocation) [109, 110] are developed for automatic indexing, clustering, searching and organizing large corpora of unlabeled documents. Topic models were originally developed as a means of automatically indexing, searching, clustering, and structuring large corpora of unstructured and unlabeled documents.

The workflow of text mining approach is divided into a) Information Retrieval (IR), b) Natural Language Processing (NLP), 3) Data Mining (DM), and Information Extraction (IE). The key tasks which are performed using text mining approaches are text categorization and clustering, concept extraction, production of taxonomies, document

summarization and sentiment analysis. The use of text categorization to introduce an automated system has been applied in certain domains, such as spam e-mail filtering [40], web page classification [41], sentiment analysis [42], author identification [43] and the design pattern selection [31-33]. In the domain of machine learning and text mining, automatic text categorization is performed either through supervised or unsupervised learning techniques. The former techniques use class labels assigned to documents and latter techniques use attributes of the data and dis(similarity) measures to automate their learning. However, in both cases, some sort of text preprocessing activities which are required to make precise learning [64]. The recommendation of a set of pre-processing activities is related to the interest of researchers and the context of the automated system. There are some common activities related to the effectiveness of an automated system are.

1. Tokenization
2. Construction of feature space with unique words
3. Reducing feature space

The common aim of all these preprocessing activities is to reduce the dimensionality of the feature space to improve the performance of classifiers.

2.4.1. Tokenization

The tokenization activity is considered as some sort of pre-processing activity used to form the basic unit of text or it can be referred as a segmentation process used to divide the text into words or sentences. In natural language processing, a single word (low-level tokenization) or group of words (high-level tokenization) is considered as a token¹⁷. For example, the “Agile Software Developer” and (“Agile”, “Software”, “Developer”) are the examples of high-level and low-level tokenization. Each tokenizer (e.g. OpenNLP) performed some common steps such as segmentation of text into words, and handling of abbreviations and hyphenated words. Subsequently, some extra morphological and lexical information is required to handle the typographical structure of the words such as a) Isolating (e.g. Words of Mandarin Chinese can be divided into smaller units), b) Agglutinative (e.g. The words of Japanese and Tamil are divided into small units), and c) Inflectional (the boundaries of the morphemes of Latine re ambiguous and are not clear in grammatical meaning). The effect of a tokenizer yield to generate a feature space which consists of the number of words/tokens.

2.4.2. Construction of Feature Space with Unique Words

The feature space which is generated after tokenization process have many meaningless words such as “are”, “this”, “and”, “or” etc., make noise in the textual data. The removal of stop words activity is performed to decrease such noise and reduce the feature space for the machine learning methods to produce the correct results. The development of a list of the stop words based on the context of text language and become difficult. Consequently, the precision of tool varied across the application. In the domain of removal of stop words methods, the existing efforts have been classified as a) The classic method [65,66], b) Zipf’s Law [67], c) Term Based Random Sampling

(TBRS) [68], and d) Mutual Information (MI) [69] methods. The word stemming process is performed to convert morphological forms of words (semantically correlated words) into their root words. There are several stemming algorithms designed on two basic assumptions a) mapping of morphological forms (i.e. Same semantics) of words into their root words, and b) words without same semantic are kept separate. The existing efforts of researcher to propose the word stemming algorithms can be categorized into a) Tuncating (Porters [70], Paice/Husk [71], Lovins, Dawson [72]) Stemmers, b) Statistical (HMM (Hidden Markov Model)[75], YASS (Yet Another Suffix Striper) [76], N-gram [73, 74]) Stemmers, and c) Hybrid (Krovetz [70], Xerox [71], Corpus based [72], context sensitive based [73]) Stemmers. In the pre-processing activities of certain automated systems, the truncating algorithms (especially Porter's algorithm) have been adopted by research community due to their simplicity and lack of computational overheads. The truncating algorithms which are related to the removal of suffixes and prefixes (a.k.a affixes) use their lookup tables. The size of the lookup table, complexity, and less error rate are the factors to evaluate and compare the effectiveness of truncating algorithms. The Porter's algorithm is one of the common truncating stemming algorithm which introduced in 1980, however, later on, it is modified. The lookup table of initial Porter's algorithm includes more than 1200 suffixes of as compared to Lovin's stemmer and Paice/Husk Stemmer which includes 294 and 120 suffixes respectively. The number of suffixes target by Porter's and Dawson Stemmers are approximately same, however, Porter's stemmer produces less error rates and complex as compared to Dawson's Stemmer. Consequently, Porter's algorithm is used in the preprocessing phase of the certain automated system.

2.4.3. Reducing Feature Space

In order to decrease the undesirable effect of useless features, feature selection methods are applied. The feature selection methods are grouped into wrapper embedded and Filter categories on the base of their interaction with classifiers to construct a feature set, which might be caused to increase the computation time. The wrappers and embedded methods are specific to the learning model as compared to filter-based methods [47, 92]. Ogura et al., categorized the filter-based feature selection methods and referred as one-sided and two-sided on the base of their characteristics. The one-sided feature selection methods assign a score greater than zero (i.e. $\text{Score} \geq 0$) to a feature for membership class while assigning a score less than zero (i.e. $\text{Score} < 0$) for non-membership classes. For example, the Odd Ratio is referred as the one-sided feature selection method, used to measure the membership and non-membership normalized values via its nominator and denominator [137]. The two-sided feature selection methods assign a score to feature, based on the combination of membership and non-membership in any class. For example, Document Frequency [92], Information Gain [90, 94], and improved Gini Index [96]. Subsequently, Tasci and Gungor classified filter-based methods into two categories and referred as local and global, depend on unique or multi class-based score assigns to a feature [48]. In the text categorization approach, the multi-class problem should be handled to improve the performance of classifiers. Gunal [138] applies the genetic algorithm and combine several filter methods and investigate the classification performance. Subsequently, the authors reported that combine filter methods outperform than the performance of an individual filter method. In a recent study, Uysal introduces an Improved Global Feature Selection Scheme (IGFSS) scheme to improve the classification performance of global

feature selection methods by constructing a more illustrative feature set which includes evenly class-wise distributed features [47]. Subsequently, the authors conduct an experiment on well-known benchmark datasets with various classifiers and used widely-known metrics to evaluate their classification performance.

There are two aims of preprocessing phase of a text categorization based automated system a) to transform unstructured text into a structured form via a vector space model, which can enable the supervised or unsupervised learning process, and b) to reduce the dimensionality of the feature space by removing noise in documents and enhance the precise learning of supervised and unsupervised learners.

Consequently, besides the recommendation of the pre-processing activities, the recommendation of supervised and unsupervised learner is also an important, because in a latest empirical study with 179 classifiers, Delgado et al. [46] concluded that no one classifier is better and it depends on the context of real world problem and working procedure selected classifiers. In the domain of text categorization based automated system, Naïve Bayes, Nearest Neighbor Classifier, Decision Trees, and Support Vector Machines (SVMs) are widely used supervised learners. Subsequently, K-means, k-medoids, Fuzzy c-means, and agglomerative are widely used unsupervised learners, whose effectiveness is related to the selected distance measures such as Euclidean and Manhattan [46, 113-120].

Implication:

- In case of text categorization based automated system, the recommendation of a set of preprocessing activities has more concern before making any precise supervised and unsupervised learning.
- In the preprocessing phase of an automated system, all activities except tokenization are used to reduce the feature space up to some extent.
- In text categorization based automated system, global filter-based feature selection methods are recommended. However, the class-wise discrimination is the main issue which might be a cause of the multi-class problem, which is recently addressed by Uysal et al. and Hussain et al. in their studies.
- The recommendation of term weighting and feature selection methods for the supervised and unsupervised learners based on the context of the conducted empirical studies. Consequently, the selected weighting method and feature method for a learner vary across the domain and context.

3.1. Overview of the Proposed Approach

The employment of design patterns in the software development process is considered as an alternative approach for software refactoring to address the declined quality of software system [5, 14, 55, 139, 140]. In recent studies [53, 54, 141], the authors have reported the developer's interest and user's perception to use the design patterns and their implications with respect to certain aspects. The existence of more than one classification scheme to organize the patterns of the same domain [5, 6, 7], inconsistencies and overlapping of patterns on the online repositories [28], and lack of developer's knowledge motivates us to introduce an approach which can aid to organize the design patterns. The authors of the widely used Gang-of-Four (GoF) design patterns quoted the design pattern selection process as a problem, when a developer is working with a collection of more than 20 design patterns, especially when a developer is unfamiliar (or lack of knowledge) with the target pattern catalog. Hasheminejad and Jalili [31], Hussain et al. [33], and Birukou [56] recommend the scan of intent of design patterns to understand the classification scheme to narrow the search and selection of patterns, which are more relevant to a design problem. The rapid development of software design patterns leads to an increase in the amount of its electronic documentation worldwide. Therefore, there is a need to introduce an approach which can aid those developers who are unfamiliar with the target design pattern catalog, however, they are interested to search and select the more appropriate patterns for a design problem on hand.

Due to increase in the development of the software design pattern in certain domains published either through books or online repositories, we consider design pattern selection as an information retrieval problem [95, 115] and look at the capacity of Text categorization [31-33, 40-43] to exploit the proposed approach. The aim of the proposed approach is to use the unsupervised learning techniques to cluster the design patterns of any target catalog, which are recorded using either the same or different classification schemes. Subsequently, the proposed approach recommends a systematic way to select the right design pattern(s) for a given design problem on the base of text relevancy. There are three objectives of the proposed approach, 1) to organize the design patterns of a target pattern catalog on the base of text relevancy among the description of their problem definition (i.e. Intent of design patterns) via unsupervised learning techniques, 2) to assign a right pattern class (i.e. Cluster of similar patterns) for a given design problem via unsupervised learning techniques, and 3) to recommend the right design pattern(s) for a given real design problem using different similarity options. We applied the text categorization steps on the description of Problem domain part of the design patterns (shown in Appendix A) and description of the design problem (shown in Appendix B). The proposed approach is formulated in four steps, which are described in the Figure 3.1.

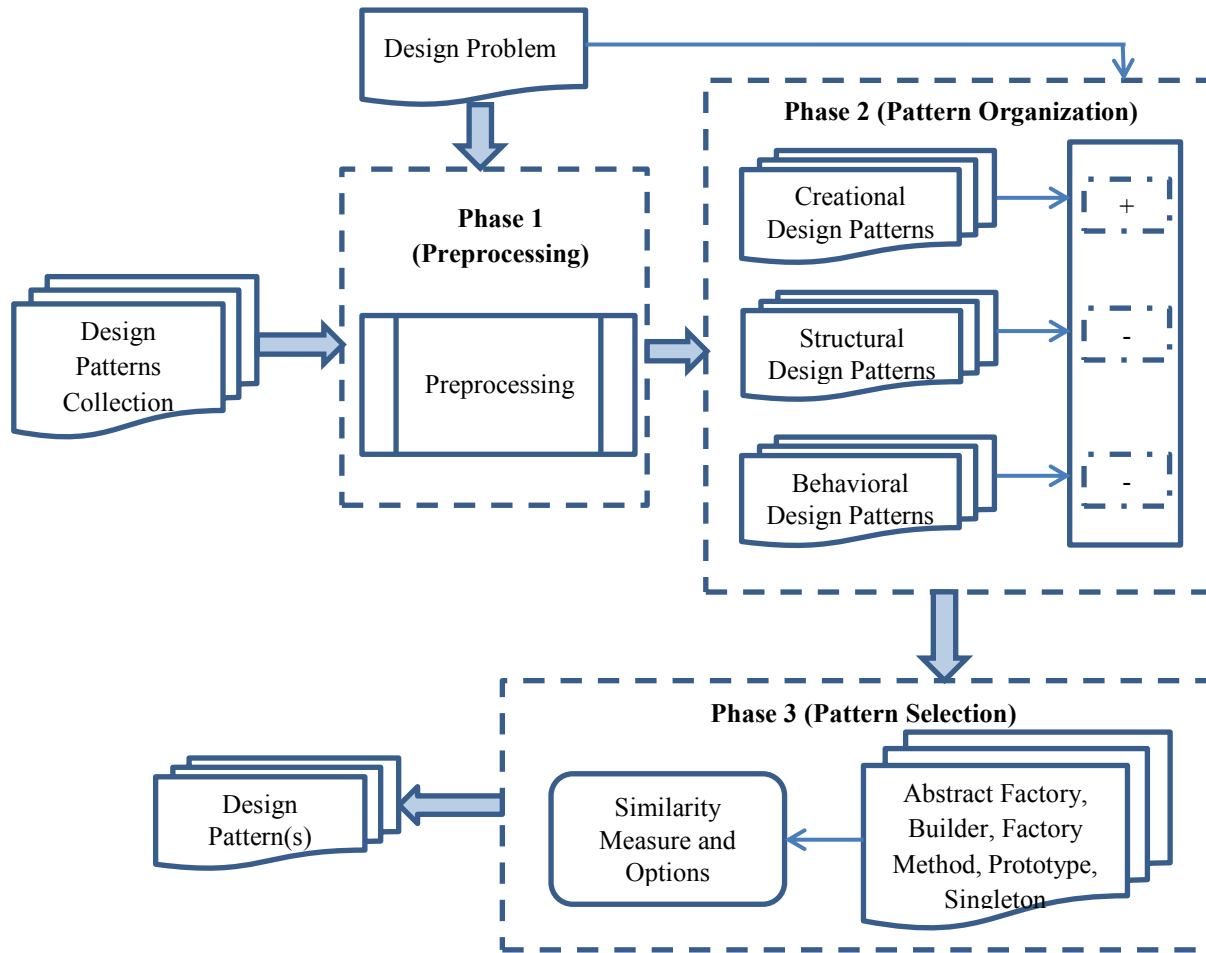


Figure 3.1. Overview of the proposed framework

3.1.1. Preprocessing

The first phase of the proposed approach is preprocessing. In this phase, a set of activities is performed to automate the text categorization of the target text. The detail of each activity is given in section 1.9. The description of design problems and problem definition part (i.e. Intent) of design pattern is the target text (i.e. Inputs) of the proposed approach. The outcome of the first phase is in the form of a Vector Space Model (VSM). The term-document based VSM has been practiced in certain studies [31-33,51, 47]. Consequently, a document collection is described through a term-document matrix D , where each entry refers to the word's occurrence in the documents

$$D = tf_{t,d} \quad (3.1)$$

In the equation 3.1, $tf_{t,d}$ is the weight of term t_i in document d . Subsequently, before apply any weighting method, the frequency of a term is considered as their weights. The working procedure of preprocessing activities is shown in the following example.

Example: We consider a sample collection of N documents (i.e. $N=7$) shown in Table 3.1.

Table 3.1. A sample of documents

Document	Content of Document
Document 1	Design Pattern Class Abstract Structure Client Create Decompose Composite Inherit Behavior Create Composite
Document 2	Design Pattern Class Abstract Structure Client Strawberry Composite Behavior Design Design Class Abstract Decompose
Document 3	Design Pattern Client Create Decompose Behavior Design Client Client
Document 4	Design Pattern Class Abstract Structure Client Create Composite Behavior Design Design Create
Document 5	Design Pattern Class Abstract Structure Client Composite Design Class Abstract Client
Document 6	Design Pattern Class Abstract Structure Client Create Decompose Inherit
Document 7	Pattern Client Create Decompose Composite Behavior Decompose

The frequency of M unique features (i.e. $M=11$) in the documents collection is computed through the equation 3.1. Subsequently, the structure of Vector Space Model (VSM) (equation 3.2) with for N documents and M non-repeated terms (with term frequency) are described in the Table 3.2.

$$\text{VSM} = \sum_{i=1}^N \sum_{j=1}^M w(T_{ij}) \quad (3.2)$$

Table 3.2. List of features with their frequencies for sample of documents

Documents	Design	Pattern	Class	Abstract	Structure	Client	Create	Decompose	Composite	Inherit	Behavior
Document 1	1	1	1	1	1	1	2	1	2	0	1
Document 2	3	1	2	2	1	1	0	2	1	0	1
Document 3	2	1	0	0	0	3	1	1	0	0	1
Document 4	3	1	1	1	1	1	2	0	1	0	1
Document 5	2	1	2	2	1	2	0	0	1	0	0
Document 6	1	1	1	1	1	1	1	1	0	1	0
Document 7	0	1	0	0	0	1	1	2	1	0	1

This weighting methods activity is performed to find the weights $tf_{t,d}$ of a word of term t_i in document d . The frequency of term t_i in document d and collection c of N documents is referred as $tf_{w_i,d}$ and $tf_{w_i,c}$. There are two assumptions which are made on the base of empirical investigation. The first and second assumptions are related to

high frequency of a term in a document and across the documents of a collection. There are several weighing methods are employed to investigate the effectiveness of features. In the proposed approach, Binary, Term Frequency (TF), Term Frequency Inverse Document Frequency (TF/IDF), Term Frequency Collection (TFC), Length Term Collection (LTC), and Entropy are recommended weighting methods [31-33,47, 51, 90].

A. Binary Weighting Method

The aim of the Binary weighting method is to assign the weight 1 to a word if it occurs in the corresponding document, otherwise weight 0 is assigned. The implication of binary weighting method for sample of documents (Table 3.1) is shown in the Table 3.3.

B. Term Frequency (TF)

The aim of the Term Frequency (TF) weighting method is to assign the frequency of a word in the corresponding document as its weight. The implication of TF weighting method for sample of documents (Table 3.1) is shown in the Table 3.2.

Table 3.3. List of features with their Binary weights for sample of documents

Documents	Design	Pattern	Class	Abstract	Structure	Client	Create	Decompose	Composite	Inherit	Behavior
Document 1	1	1	1	1	1	1	1	1	1	0	1
Document 2	1	1	1	1	1	1	0	1	1	0	1
Document 3	1	1	0	0	0	1	1	1	0	0	1
Document 4	1	1	1	1	1	1	1	0	1	0	1
Document 5	1	1	1	1	1	1	0	0	1	0	0
Document 6	1	1	1	1	1	1	1	1	0	1	0
Document 7	0	1	0	0	0	1	1	1	1	0	1

C. Term Frequency Inverse Document Frequency (TF/IDF)

The aim of the Term Frequency Inverse Document Frequency (TF/IDF) weighting method is to assign the proportion of frequency of words in the document and inverse proportion to the number of documents in which word occurs at least one time as its weight. The implication of TF/IDF weighting method for sample of documents (Table 3.1) is shown in the Table 3.4.

Table 3.4. List of features with their TFIDF weights for sample of documents

Documents	Design	Pattern	Class	Abstract	Structure	Client	Create	Decompose	Composite	Inherit	Behavior
Document 1	-0.54	0	0	0	0.334	-0.36	0	0	0.31	1.25	0.34
Document 2	-1.62	0	0	0	0.34	-0.36	0	0	0.15	0	0.34
Document 3	-1.08	0	0	0	0	-1.07	0	0	0	0	0.34
Document 4	-1.62	0	0	0	0.34	-0.36	0	0	0.15	0	0.34
Document 5	-1.08	0	0	0	0.34	-0.71	0	0	0.15	0	0
Document 6	-0.54	0	0	0	0.34	-0.36	0	0	0	1.25	0
Document 7	0	0	0	0	0	-0.36	0	0	0.15	0	0.34

D. Term Frequency Collection (TFC)

The aim of the Term Frequency Collection (TFC) weighting method is to assign the frequency of a term throughout the all documents in the collection (length of documents is normalized with M words) as its weight. The implication of TFC weighting method for sample of documents (Table 3.1) is shown in the Table 3.5.

Table 3.5. List of features with their TFC weights for sample of documents

Documents	Design	Pattern	Class	Abstract	Structure	Client	Create	Decompose	Composite	Inherit	Behavior
Document 1	-0.18	0	0	0	0.11	-0.12	0	0	0.10	0.43	0.11
Document 2	-0.36	0	0	0	0.07	-0.08	0	0	0.03	0	0.07
Document 3	-0.39	0	0	0	0	-0.39	0	0	0	0	0.12
Document 4	-0.33	0	0	0	0.07	-0.07	0	0	0.03	0	0.07
Document 5	-0.19	0	0	0	0.06	-0.13	0	0	0.03	0	0
Document 6	-0.11	0	0	0	0.07	-0.07	0	0	0	0.25	0
Document 7	0	0	0	0	0	-0.07	0	0	0.03	0	0.06

E. Length Term Collection (LTC)

The aim of the Length Term Collection (LTC) weighting method is to assign logarithm of its frequency (rather than raw frequency) as its weight. The implication of LTC weighting method for sample of documents (Table 3.1) is shown in the Table 3.6.

Table 3.6. List of features with their LTC weights for sample of documents

Documents	Design	Pattern	Class	Abstract	Structure	Client	Create	Decompose	Composite	Inherit	Behavior
Document 1	-0.20	0	0	0	0.12	-0.13	0	0	0.09	0.47	0.13
Document 2	-0.29	0	0	0	0.09	-0.09	0	0	0.04	0	0.09
Document 3	-0.41	0	0	0	0	-0.35	0	0	0	0	0.16
Document 4	-0.29	0	0	0	0.09	-0.09	0	0	0.04	0	0.09
Document 5	-0.23	0	0	0	0.09	-0.15	0	0	0.04	0	0
Document 6	-0.17	0	0	0	0.11	-0.11	0	0	0	0.40	0
Document 7	0	0	0	0	0	-0.13	0	0	0.05	0	0.12

F. Entropy

The aim of the Entropy weighting method is to assign the average uncertainty of words as its weight. The implication of Entropy weighting method for sample of documents (Table 3.1) is shown in the Table 3.7.

Table 3.7. List of features with their Entropy weights for sample of documents

Documents	Design	Pattern	Class	Abstract	Structure	Client	Create	Decompose	Composite	Inherit	Behavior
Document 1	-0.51	-0.51	-0.51	-0.51	-0.51	-0.51	-0.81	-0.51	-0.81	-0.51	-0.51
Document 2	-0.79	-0.39	-0.63	-0.63	-0.39	-0.39	0	-0.63	-0.39	0	-0.39
Document 3	-0.06	-0.04	0	0	0	-0.08	-0.04	-0.04	0	0	-0.03
Document 4	-0.74	-0.37	-0.37	-0.37	-0.37	-0.37	-0.59	0	-0.37	0	-0.37
Document 5	-0.34	-0.21	-0.34	-0.34	-0.21	-0.34	0	0	-0.21	0	0
Document 6	-0.31	-0.31	-0.31	-0.31	-0.31	-0.31	-0.31	-0.31	0	-0.31	0
Document 7	0	-0.06	0	0	0	-0.06	-0.06	-0.09	-0.06	0	-0.06

Subsequently, the feature selection activity is performed to reduce the dimensionality of the feature space by removing the unimportant and meaningless terms. Since the wrapper and embedded based feature selection methods require classifier interaction, consequently, the use of filter-based methods are recommended for the automatic systems implemented via text categorization approach [31-33, 64]. The implications of suggested global filter-based feature selection methods

Table 3.8. Constructed features set with M ranked features (M=5) for the sample of documents

Top-M Ranked Features (M=5)	Constructed Feature sets				
	DF	IG	GR	Chi-Sq	GI
1	Design	Create	Create	Create	Pattern
2	Client	Structure	Structure	Structure	Create
3	Pattern	Pattern	Pattern	Pattern	Composite
4	Class	Class	Class	Class	Client
5	Create	Client	Client	Client	Class

(Document Frequency (DF), Information Gain (IG), Gain Ratio (GR), Chi-Square, and Gini Index(GI)) for the sample of documents (with M=5) is shown in the Table 3.8.

3.1.1.1. Ensemble-IG (Proposed feature selection method)

Besides the proposed approach, one of our contributions is to introduce a new feature selection technique named Ensemble-IG by leveraging IGFSS (Improved Global Feature Selection Scheme) [47]. The proposed method, Ensemble-IG, ensemble the one-sided feature selection method OR (Odd Ratio) and global feature selection method Information Gain (IG) about the workflow of leveraged scheme IGFSS. The aim of Ensemble-IG is to overcome the issue of the multi-class problem (in the text categorization) as exist in the classic form of Information Gain (IG). For example, in the classic form of IG, single score of each feature f is calculated using the equation 3.3.

$$IG(f) = -\sum_{k=1}^{NC} P(C_k) \log P(C_k) + P(f) \sum_{k=1}^{NC} P(C_k | f) \log P(C_k | f) + P(\bar{f}) \sum_{k=1}^{NC} P(C_k | \bar{f}) \log P(C_k | \bar{f}) \quad (3.3)$$

In the equation 3.3, the NC , $P(C_k)$, $P(f)$, $P(\bar{f})$, $P(C_k | f)$, and $P(C_k | \bar{f})$ present the number of classes, probability of class C_k , probability for the presence of the feature f , probability for the absence of the feature f , probability of class C_k for the presence of feature f , and the probability of class C_k for the absence of feature f . The workflow of the proposed Ensemble-IG is based on the improved global feature selection scheme by ensemble the classic IG and one sided OR (Odd Ratio) feature selection method, as follows.

Procedure Ensemble-IG (VSM, AL, CL, FL, SFL, FFL, n)**Input:**

VSM = Vector Space Model

AL = {Set of Actual Labels}, For example, in the case of GoF design pattern collection, Structural, Creational and Behavioral class labels are used.

CL = {Set of Created Labels with $NC * 2$ }, where NC is the number of classes (or cardinality of AL)

FL = Actual Feature List

SFL = Sorted Feature List

FFL = Final Feature List

n = The value of n referred to top n features

Begin

Step-1. The first step is used to assign labels to features.

- a. Compute the one-sided local feature selection score (using Odd Ratio) of each feature of FL for each class of AL.
- b. The new label set CL is created with $NC * 2$ labels to describe the membership and non-membership to the classes of AL.
- c. For each feature of FL, compute the local feature selection score, and assign the labels according to highest score.

Step-2. The second step is to apply the classic IG.

- a. Compute the IG feature selection score for each feature.
- b. Create a new sorted (in descending order) feature list SFL

Step-3. The third step is to construct a new feature set.

- a. Traverse each feature of SFL and create a FFS with a size equal to the cardinality of FL and set of negative feature ratio in the range from 0 to 1.
- b. The constructed FFS should be representative of each class of AL

Step-4. The fourth step described the end of the Ensemble-IG workflow.

- a. If the cardinality of FFL is less than the from the cardinality of FL, then add the disregarded features with highest global feature selection score to FFL.

End**Output:**

The creation of new VSM which contain the equal number of features for the each class of AL.

3.1.2. Unsupervised learning for Design Patterns

The proposed approach is based on the Text categorization approach. Subsequently, it is exploited by unsupervised learning techniques. Since due to unfamiliarity (or lack of knowledge) of developers with the target pattern catalog, the unsupervised learning can aid developers to organize the design pattern on the basis of their text relevancy (with or without a given design problem) and narrow down the search for the required pattern.

Consequently, the organization of design patterns can aid to overcome the problems related to the pattern searching [22]. There are several unsupervised learning methods, however, it cannot be said which one is better than others [44, 45]. For example

- A. Hierarchical Clustering techniques are used to cluster the data according to predetermined ordering such as from top to bottom or from bottom to top, such as
 - Agglomerative Algorithm
 - Divisive Algorithm
- B. Partitional Clustering techniques are used to decompose the dataset into a set of disjoint clusters, such as
 - K-means Algorithm
 - Fuzzy c-means
 - Partition Around Medoid
- C. Model-based clustering techniques are used to optimize the fit between the given data and some mathematical models, such as.
 - Decision Trees
 - Neural Networks
- D. Density-based clustering techniques are used to draw a cluster from a specific probability distribution, such as.
 - DBSCAN (Density Based Spatial Clustering of Applications with Noise)
 - AUTOCLASS

In order to fulfill our contribution, several unsupervised learning techniques are assessed in the domain of our problem (that is the classification and selection of right design pattern(s) for a given design problem) and recommend the best one which is more compatible with our problem domain. We consider five common unsupervised learning techniques in the text mining literature [64, 143], which are Fuzzy c-means, K-means, Agglomerative, and Partitioning Around Medoids (a.k.a. K-Medoids with distance measures Euclidean and Manhattan). The detail of unsupervised learning technique which are used in the proposed approach is given in section 1.9.3.

3.1.3. Identification of Design Pattern Class

In the proposed approach, a design pattern class refers as a group of patterns suggested via the unsupervised learner of the proposed approach for a given design problem. The number of design pattern classes (according to expert's opinion) depends on the decision of an unsupervised learning technique used in the proposed method. For example, in the Figure 3.1, we choose Gang-of-Four design pattern collection to illustrate the design pattern classes for a given design problem. Subsequently, each block inside the rectangle (labeled as a Phase 2) refer as a design pattern class, while + and – symbols denote the decision of the corresponding learning technique (e.g. Fuzzy c-means) for the given design problem. The + symbol describes the closeness of a design problem with patterns (i.e. Members) of corresponding candidate design pattern class (i.e. Cluster).

3.1.4. Design Patterns Suggestion and Selection

The unsupervised learning techniques (suggested in the proposed approach) are used to narrow down the pattern search via clustering the patterns (Section 4.2) on the base of their similarity and dissimilarity employed through distance measures such as Minkowski, Euclidean, and Manhattan. Through clustering the pattern search become narrow. Subsequently, the selection of right pattern(s) from the cluster (i.e. Pattern class for the given design problem) is still required. The aim of this phase of our proposed approach is to suggest the design pattern(s) using similarity measures and options. Similarity measure(s) is/are suggested to compute the degree of closeness between given design problem and patterns of suggested pattern class (i.e. Cluster of the patterns which is suggested through the proposed approach (Section 3.1)). Similarly, the similarity options are suggested to select the right pattern(s) by tuning certain thresholds parameters. We evaluate numerous similarity measure (with different aspects) such as Cosine, Pearson Correlation, Extended Jaccard, Euclidean distance, Manhattan Distance, Minkowski (a generalized form of Euclidean and Manhattan distance), and Dice Coefficient [144]. In the proposed approach, we recommend the Cosine Similarity measure to compute the degree of closeness between design problem and patterns of suggested pattern class. The main factors behind the recommendation of Cosine Similarity measures are.

- i. The Cosine similarity measure is very efficient to assess the sparse vectors (i.e. Vectors which have mostly zero values) as compared to other measures. In this regard, we consider two documents and described them in the vector form by depicting the frequency of their independent terms.

<u>Document 1</u>	<u>Document 2</u>
Design Structure Pattern Design Client Inherit	Design Composite Client Abstract Class Inherit

The sparse vector form of both documents in terms of the frequency of words and their closeness using similarity measures is shown in the Table 3.9.

Table 3.9. Similarity measure for sparse vectors

Unique Terms	<Design, Structure, Pattern, Composite, Client, Abstract, Class, Inherit>
Vector for Document 1	< 2, 1, 1, 0, 1, 0, 0, 1 >
Vector for Document 2	< 1, 0, 0, 1, 1, 1, 1, 1>
Cosine Similarity	0.58
Jaccard	0.37
Pearson Correlation	0.50
Dice	0.54
Euclidean	0.28
Manhattan	0.14

ii. The heterogenous description of published patterns [26] varies in the length, which motivate us to suggest the use of Cosine Similarity (CS) measure because this measure is independent of document length [144]. In this regard, we consider two identical copies of a document d1 and get a new document d2.

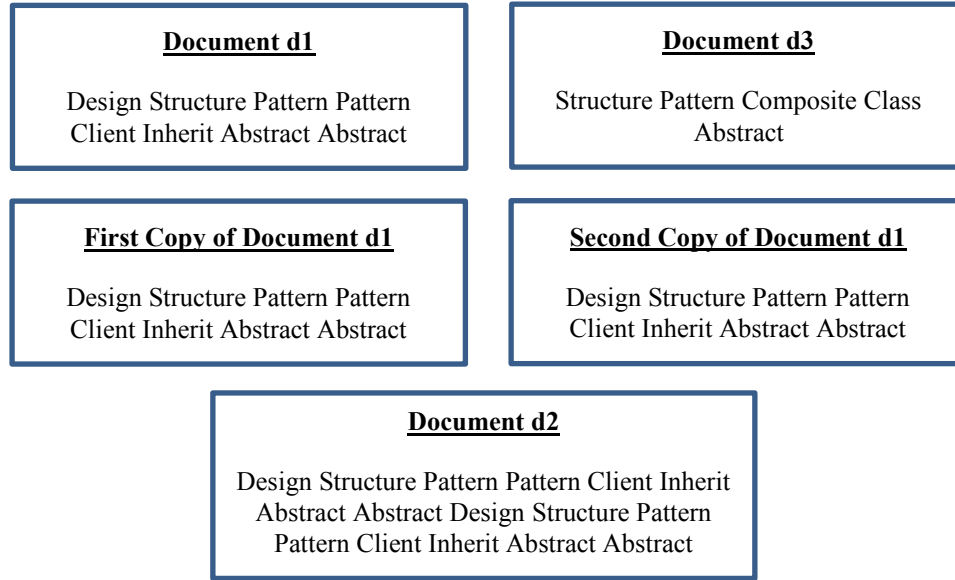


Table 3.10. Similarity measures with different length of documents

Unique Terms	<Design, Structure, Pattern, Composite, Client, Abstract, Class, Inherit>
Vector for Document d1	< 1, 1, 2, 0, 0, 1, 1, 2 >
First Copy (C1) of d1	< 1, 1, 2, 0, 0, 1, 1, 2 >
Second (C2) Copy of d1	< 1, 1, 2, 0, 0, 1, 1, 2 >
Vector for Document d2	< 2, 2, 4, 0, 0, 2, 2, 4>
Vector for Document d3	< 0, 1, 1, 1, 1, 0, 0, 1>
Cosine Similarity (CS)	CS(d1, d2) = 1 CS(d1, d3) = 0.64 CS(d2, d3) = 0.64
Euclidean Distance (ED)	ED(d1, d2) = 0.22 ED(d1, d3) = 0.27 ED(d2, d3) = 0.15

The vector form of three documents d1, d2 (by combining two copies of d1), and d3 with cosine similarity values are shown in Table 3.10. The results of Table 3.10 depict that the patterns with the same composition and different to indicate that documents d1 and d2 are identical.

iii. Since the proposed approach is introduced by considering the design pattern classification and selection as information retrieval problem, the Cosine similarity measure is suggested for the text categorization based applications [31-33, 51,100, 144].

iv. The cosine similarity measure relies on the orientation of feature vector rather than magnitude and normalized the inner product of vectors. In this regard, we consider four documents

<p><u>Document d2</u></p> <p>Design Inherit Abstract Composite Class</p>	<p><u>Document d1</u></p> <p>Design Structure Pattern Client</p>
<p><u>Document d3</u></p> <p>Design Design Design Design Design Design Design Design Design Design Structure Pattern Client</p>	<p><u>Document d4</u></p> <p>Design Structure Structure Structure Structure Structure Structure Structure Structure Structure Structure Inherit Abstract Composite Class</p>

Table 3.11. The impact of similarity measures with magnitudes

Unique Terms	<Design, Structure, Pattern, Client, Inherit, Abstract, Composite, Class>
Vector for Document d1	< 1, 1, 1, 1, 0, 0, 0, 0 >
Vector for Document d2	< 0, 0, 0, 0, 1, 1, 1, 1 >
Vector for Document d3	< 10, 1, 1, 1, 0, 0, 0, 0 >
Vector for Document d4	< 1, 10, 0, 0, 1, 1, 1, 1 >
Cosine Similarity (CS)	CS(d1, d2) = 0.22 CS(d3, d4) = 0.19
Euclidean Distance (ED)	ED(d1, d2) = 0.27 ED(d3, d4) = 0.21
Jaccard Similarity (JS)	JS(d1, d2) = 0.12 JS(d3, d4) = 0.25

The results of Table 3.11 depict that the cosine similarity measure is less prone the magnitude of the terms as compared to Jaccard and Euclidean distance measure. The aim of Cosine Similarity measure in the third phase of the proposed approach is to find the degree of closeness between given design problem and the N design patterns of suggested pattern class (in the second phase of proposed approach). Firstly, the term weighted vectors for Pattern_i ($i \in N$) and Problem are described through the equation 3.4.

$$W(d) = (w(d, t_1), w(d, t_2), \dots, w(d, t_n)) \quad (3.4)$$

Secondly, the correlation between design problem and design patterns of the suggested pattern class is calculated using the equation 3.5.

$$CS_i = \sum_{j=1}^N w(\text{Pattern}_i, t_j) \times w(\text{Problem}, t_j) \quad (3.5)$$

Subsequently, after computing the degree of closeness between given design problem and the N design patterns of suggested pattern class, we need to recommend similarity options, which can aid for suggestion of design pattern(s) to a developer.

A. First Similarity Option

The first similarity option is used to find the right design pattern on the base of highest cosine value among N values determined through the equation 3.5. In mathematics and statistics domain, there are several techniques are used to find a target value such as.

- 1) Mode (return a value which is often appears in a data set),
- 2) Extrema (collectively refer to maxima (largest value),
- 3) Minima (smallest) value), and
- 4) Mathematical optimization (used for selection of best element from a list, but need to satisfy some criterion).

In the proposed framework, we used Argument of the Maxima (Arg Max) function to find the largest output. Consequently, we used Arg Max function as first similarity option which aid to recommend the right design pattern (with highest cosine value) from the list of N patterns. Equation 3.6 can aid to find the k^{th} design pattern (as right design pattern) to a developer.

$$k = \arg \max_i CS_i \quad (3.6)$$

For example, in the Table 3.12, we present a list of patterns (i.e. N=7) and their cosine value to describe the closeness with a given design problem. The first similarity option (equation 3.6) returns the P2 with highest cosine value as a right design pattern.

B. Second Similarity Option

Due to the closeness between structural similarity of patterns (such as in case of the GoF design patterns, Adapter, Decorator, Proxy, and Façade use the composition and delegation to solve the design problems), a developer can often use the same pattern to solve certain problems without caring their intent [5, 9]. In such scenario, a pattern catalog is updated with certain alternative solutions. The aim of second similarity option is to

suggest the zero, one, or more than one design pattern to a developer on the base of the value of a threshold parameter Θ_1 . The value of a threshold parameter Θ_1 can be determined through the target catalog and sample of design problems (discussed in the section 4.3), however, a developer can also tune the threshold parameter Θ_1 . The equation 3.7 can be used to recommend zero, one, or more than one design pattern depends on the value of Θ_1 .

$$|CS_i| > \Theta_1 \quad (3.7)$$

C. Thrid Similarity Option

The aim of third similarity option (equation 3.8) is to suggest design patterns contiguous to the best design pattern (with $CS_{\max}$) to the developer on the base of the value of a threshold parameter Θ_2 . Like a second similar option, the value of a threshold parameter Θ_2 can be determined through the target catalog and sample of design problems (discussed in the section 4.3), however, a developer can also tune the threshold parameter Θ_2 .

$$|CS_{\max} - CS_i| \leq \Theta_2 \quad (3.8)$$

Table 3.12. Sample of patterns and their closeness with a given design problem

i	Patterns of suggested pattern class	Cosine Similarity (CS_i)
1	P1	0.33
2	P2	0.51
3	P3	0.31
4	P4	0.20
5	P5	0.34
6	P6	0.20
7	P7	0.26

4.1. Evaluation Model for the Proposed Approach

The proposed approach is employed through the use of unsupervised learning technique, which aided to cluster (i.e. Organize) the design patterns according to expert's opinion. The evaluation of unsupervised learning technique is a controversial issue (i.e. Which means neither good nor bad). The first time Bonner [145] contended that there is no universal definition of a good clustering, however, the evaluation remains in the eye of the domain specific beholders. There are several measures have been introduced to assess the clustering quality. These measures can be grouped into two categories (that are internal and quality criteria) on the basis of certain aspects. The measures which are related to the internal quality of clustering techniques aid to assess the inter-cluster homogeneity and separability or their combination. These measures not use any external information and the data itself. The main criteria to assess the internal clustering quality include the least variance partition (Sum of Squared Error (SSE)) [146], Condorcet's Criteria [147], C-Criteria (extension of Condorcet's Criteria) [148], and Category Utility metric [149]. The measures (i.e. Aka clustering comparison measures) which are related to the external quality of clustering techniques aid to find the closeness between the structure of the clusters and predefined classification of the data instances. Since the objective of the proposed approach is to organize and select the design patterns according to expert opinion, consequently, we propose an evaluation model based on the external (rather than internal) quality measures. The clustering community has extensively used the external quality criteria when the ground truth reality of clustering is available [150, 151]. The cluster partitions similarity measures such as RI (Rand Index) [152] and Jaccard Coefficient [153], and Information Theoretic (IT) [154, 155] are widely used pair-counting measures. The recommendation of best measure for a particular application becomes a challenge in the existence of different measures [156]. In order to improve the interpretability of the clustering results, adjusted for chance measures are widely used as external quality criteria such as ARI (Adjusted Rand Index) [157] and AMI (Adjusted Mutual Information) [158]. The ARI validation criteria are used when the reference clustering (on ground truth) has large and nearly equal size clusters and no assumption should be made on the cluster structure such as k-means which usually assumes the isotropic blob shapes. Subsequently, the ARI is also useful in a purely unsupervised setting to generate the consensus index and enable the use of clustering model. In the proposed approach, we also used the ARI to investigate the effectiveness of unsupervised learning techniques.

In order to describe the effectiveness of proposed approach, we formulate an evaluation model which can be employed using a set of suggested measures within the context of four design pattern collections and 105 real design problems. The set of measures is suggested to investigate the effectiveness of the proposed approach in different aspects, such as.

1. To find the best weighting method for an unsupervised learning technique (used in the proposed approach) within the context of giving corpus.
2. To find the best unsupervised learning technique (used in the proposed approach) within the context of giving corpus.
3. To find the right design pattern(s) for the given design problems.

The evaluation model for the proposed approach (Figure 3.1) is performed in four steps. The first step formulates the evaluation of unsupervised learning techniques used in the proposed method to find the candidate design pattern class for a particular design problem. Subsequently, the second step formulate the evaluation of proposed options describe in Section 4.1.2 to suggest the right design pattern(s) from the candidate design pattern class. Finally, the third step formulates the use of resources include four design pattern collections and 105 real design problems to evaluate the proposed approach.

4.1.1. Evaluation of Unsupervised Learning Techniques

The evaluation performance of unsupervised learning technique is not as trivial as we consider precision and recall of a supervised learning technique. However, some ground truth set of classes or some assumptions which satisfied that, members belong to the same class are more similar using some sort of similarity metrics. For example, the Adjusted Rand Index (ARI), V-measure and Mutual Information (MI) metrics need ground truth set of classes to evaluate unsupervised learning techniques [62]. However, Silhouette Coefficient metric did not need ground truth labels and learning technique itself perform evaluation [63]. The unsupervised learner with the highest value of any one of these similarity matrices is considered as the best classifier.

Therefore, in order to meet the main goals of the proposed approach, the classification effectiveness of unsupervised learning techniques is measured in three phases.

4.1.1.1. Phase-1

In the first phase, the performance measures Recall (R), Precision (P), and F-measure aid to select the best weighting method for each respective unsupervised learner of the proposed approach. In order to select the best weighting method for each learner, the effect of each method such as Binary, TFIDF, TF, TFC, and Entropy on learning technique is computed in term of F-measure.

$$P = \frac{\sum_{c=1}^N TP_c}{\sum_{c=1}^N (TP_c + FP_c)}, \quad \text{Micro-Averaging} \quad (4.1)$$

$$R = \frac{\sum_{c=1}^N TP_c}{\sum_{c=1}^N (TP_c + FN_c)}, \quad \text{Micro-Averaging} \quad (4.2)$$

$$F = \frac{2 \times P \times R}{(P + R)} \quad (4.3)$$

In the equations 4.1-4.3, N is the number of design pattern classes decided to evaluate the performance of learning techniques. For example, in the case of GoF design pattern collection N is 3 (Section 4.2.1). The term TP is the number of design problems which are correctly identified for each design pattern class, FP is the number of design problems which are incorrectly identified for each design pattern class, and FN is the number of design problems

which are missing from each corresponding design pattern class. The values of P, R, and F are computed using equations 4.1-4.3 respectively. The evaluation procedure to select the best weighting method for an unsupervised learning technique is shown as follows.

Step-1. Select the pattern collection (i.e. Pattern corpus), such as Gang-of-Four in the domain of object-oriented development.

Step-2. Apply the text preprocessing activities (Section 3.1.1) such as removal of stop words, word stemming, and indexing to construct a Vector Space Model (VSM).

Step-3. Select a weighting method (Binary, TFIDF, TF, TFC, and Entropy) to assign weight.

Step-4. Select an unsupervised learning technique (used in the proposed approach) and apply on the VSM to organize the patterns according to the predefine number of classes, For example, in the case of GoF, the unsupervised learning technique is evaluated with $k=3$.

Step-5. The equations 4.1-4.3 are computed to find the performance of the selected unsupervised learner on VSM of the selected weighting method.

Step-6. Repeat step-3 to Step-5 until the unsupervised learning technique with all weighting methods used in the proposed approach have been evaluated.

Step-7. For each unsupervised learner, the weighting method with highest F-measure value is considered as its best weighting method.

4.1.1.2. Phase-2

Subsequently, in the second phase, Adjusted Rand Index (ARI) as a pair counting algorithm aid to select the best learning technique of the proposed approach. The value of ARI is computed using equations 4.4-4.5 and an unsupervised learner is considered as best with the highest ARI value.

$$ARI = \frac{\text{RandomIndex} - E[\text{RandomIndex}]}{\max(\text{RandomIndex}) - E[\text{RandomIndex}]} \quad (4.4)$$

$$\text{RandomIndex} = \frac{P_a + P_b}{CI_2^{n_{\text{samples}}}} \quad (4.5)$$

In the equations 4.4-4.5, CI is a ground truth class label and K is the assumed number of classes decided by the unsupervised learner. Suppose we have two partitions P_1 (with n_1 elements) and P_2 (with n_2 elements) for n elements (in case of GoF pattern group, the value of n will 23) with the same number of classes N (i.e. In case of GoF pattern group, the value of N will 3). Subsequently, we suppose that N_1 and N_2 are the two disjunctive tables and T as a contingency table with n_{ij} elements. In order to compute the Random Index, each partition P is characterized by $n \times n$ paired comparison table to evaluate the total pair of agreements. For example.

- a) Pairs belong to the same classes of both P_1 and P_2 .
- b) Pairs belong to different classes of both P_1 and P_2 .
- c) Pairs belong to the same class of P_1 but to different classes of P_2 .
- d) Pairs belong to the same class of P_2 but to different classes of P_1 .

The first two cases (a & b) referred as total number of agreements while the last two cases (c & d) referred as the total of discordances. The term (P_a in equation 4.5) is the number of pairs of elements that are identified in the same set of Cl and K , while P_b is the number of pairs of elements that are identified in the different set of Cl and K [151, 154, 155]. The evaluation procedure to select the best unsupervised learning technique is shown as follows.

Step-1. Select the pattern collection (i.e. Pattern corpus) with ground truth class labeling (i.e. Cl in the equation 4.5), such as Gang-of-Four design pattern collection, each pattern is associated with Creational, Structural, and Behavioral class labels.

Step-2. Select an unsupervised learner.

Step-3. Select the best weighting method for the selected unsupervised learner using the steps described in the phase 1.

Step-4. Apply the selected unsupervised learner with its best weighting method to predict the new label for each pattern (i.e. K).

Step-5. The equations 4.4-4.5 are computed to find the value of ARI with ground truth (i.e. Cl) and predicted (i.e. K) class labels.

Step-6. Repeat step-2 to Step-5 for each unsupervised learning technique.

Step-7. The unsupervised learner of the proposed approach (in the context of a pattern corpus) with highest ARI value is recommended a better unsupervised learner.

4.1.1.3. Phase-3

Finally, in the third phase, the effect of Document Frequency (DF), Correlation, Information Gain (IG), Chi-Square, Gain Ratio (GR), and Ensemble-IG (proposed technique) feature selection methods on the performance of unsupervised learner is evaluated, and best feature selection is selected. Like phase 1, in this phase the performance measures Recall (R), Precision (P), and F-measure aid to select the best feature selection method for each respective learner of the proposed approach. The evaluation procedure to select the best feature selection method for each unsupervised learning technique is shown as follows.

Step-1. Select the pattern collection (i.e. Pattern corpus), such as Gang-of-Four in the domain of object-oriented development.

Step-2. Apply the text preprocessing activities (Section 3.1.1) such as removal of stop words, word stemming, index, and weighting method to construct a Vector Space Model (VSM).

Step-3. Select a global filter-based feature selection method (Document Frequency (DF), Information Gain (IG), Mutual Information (MI), Correlation Coefficient (CC) and Chi-Square (CHI)) to select the top N features.

Step-4. Select an unsupervised learning technique (used in the proposed approach) and apply on the VSM to organize the patterns according to the predefine number of classes, For example, in the case of GoF, the unsupervised learning technique is evaluated with $k=3$.

Step-5. The equations 4.1-4.3 are computed to find the performance of the selected unsupervised learner on VSM with top N features.

Step-6. Repeat step-3 to Step-5 until the unsupervised learning technique with all feature selection methods used in the proposed approach have been evaluated.

Step-7. For each unsupervised learner, the feature selection method with highest F-measure value is considered as its best feature selection method.

The procedures which are described to evaluate the effectiveness of unsupervised learners of the proposed approach for the organization of design patterns. Subsequently, the same procedures are used to investigate the effectiveness of unsupervised learners to find the right design pattern for the given design problems.

4.1.2. Evaluation of Selection of Design Pattern(s) Process

After identification of a candidate design pattern class, the next step is to investigate the effectiveness of the proposed approach to select more appropriate design pattern(s) for a given design problem. The cosine similarity measure is used to find the degree of closeness between patterns of suggested pattern class and a given design problem. We suggest the similarity options (3.4) are used to suggest the right design patterns. In order to choose the best similarity option, the ratio of correctly suggested design patterns and total suggested design pattern is computed using the equation 4.6 of RCD (Ratio of Corrected Design patterns).

$$RCD = \frac{\text{Number of Right Suggested Design Pattern}}{\text{Total Suggested Design Patterns}} \quad (4.6)$$

The evaluation procedure to select the best similarity option is shown as follows.

Step-1. Select the patterns (suggested patterns) of the pattern class suggested through an unsupervised learner for a given design problem.

Step-2. Compute the cosine similarity using the equation 3.5 given in the section 3.1.4.

Step-3. Apply the similarity options (equations 3.6-3.8 in the Section 3.1.4) to recommend the appropriate (or right) design patterns for developers.

Step-4. Compute the equation 4.6 to find the RCD value for each similarity option.

Step-5. Repeat from Step-1 to Step-4 until the patterns for all problems of the sample are suggested.

In this step of evaluation, similarity equations described in Section 3.1.4 are applied and more appropriate design pattern(s) are suggested to the developer. The suggested design pattern(s) are retrieved from the list of candidate design pattern class. The ratio of correctly suggested design patterns and total suggested design pattern, aid in choosing the best similarity equation. The similarity equation with the highest value of RCD (Ratio of Corrected Design patterns), computed by the equation 4.6, is selected.

4.2. Design Pattern Groups

In software engineering, some groups of experts from different domains have categorized the design patterns on the base their intent, motivation, and applicability. The domain of patterns, corresponding community interest, the number of design patterns and pattern classes are the main factors behind the rationale to select a design pattern collection. We consider four often used design pattern collection in our proposed evaluation model.

4.2.1. Gang-of-Four (GoF) Design Pattern Collection

The GoF design Patterns are described to target the certain issue occurs in the context of object-oriented software development. The GoF collection includes 23 object-oriented design patterns which are divided into three groups named Creational, Structural, and Behavioral. The GoF patterns of each pattern class are described with respect to class and object level scopes [5].

A. Creational Pattern Class

The Creational patterns are defined at class and object level and use some part of objects or subclasses for creation of new objects. The list of design patterns is as follows.

- i. Factory Method (Class-level)
- ii. Abstract Factory (Object-level)
- iii. Builder (Object-level)
- iv. Prototype (Object-level)
- v. Singleton (Object-level)

B. Structural Pattern Class

The Structural patterns at class level use the inheritance to compose the classes, while at object level, different objects are assembled. The list of design patterns is as follows.

- i. Adapter (Class-level and Object-level)
- ii. Bridge (Object-level)
- iii. Composite (Object-level)
- iv. Decorator (Object-level)
- v. Facade (Object-level)
- vi. Flyweight (Object-level)
- vii. Proxy (Object-level)

C. Behavioral Pattern Class

The Behavioral patterns at class level use the inheritance relevant algorithms to describe the flow of control, while at object level use the cooperation of objects to perform a common task. The list of design patterns is as follows.

- i. Adapter (Class-level)
- ii. Template Method (Class-level)
- iii. Chain of Responsibility (Method-level)
- iv. Command (Method-level)
- v. Iterator (Method-level)
- vi. Mediator (Method-level)
- vii. Memento (Method-level)
- viii. Observer (Method-level)
- ix. State (Method-level)
- x. Strategy (Method-level)

xi. Visitor (Method-level)

This case study includes 1465 non-repeated words of all 23 design patterns after removing the stop words and words stemming [5].

4.2.2. Douglass Design Pattern Collection

The Douglass design Patterns are described to target the certain issues occurs in the context of the development of real time applications. The Douglass collection includes 34 real time systems relevant design patterns which are divided into five categories named Concurrency, Safety, and Reliability, Distribution, Memory, and Resource. The list of pattern classes with corresponding design patterns is as follows [12].

A. Memory Pattern Class

The aim of Memory design patterns is to focus on the efficient management of memory by considering it as a shared resource and make sure the schedulability of the system. The list of design patterns related to Memory pattern class is as follows.

- i. Static Allocation Pattern
- ii. Pool Allocation Pattern
- iii. Fixed Size Buffer Pattern
- iv. Smart Pointer Pattern
- v. Garbage Collection Pattern
- vi. Garbage Compactor Pattern

B. Resource Pattern Class

The aim of Resource design patterns is to organize and manage the shared resources used in real-time and embedded systems. The list of design patterns related to Resource pattern class is as follows.

- i. Critical Section Pattern
- ii. Priority Inheritance Pattern
- iii. Highest Locker Pattern
- iv. Priority Ceiling Pattern
- v. Simultaneous Locking Pattern
- vi. Ordered Locking Pattern

C. Concurrency Pattern Class

The aim of concurrency design patterns is to control and schedule the architectural elements. Subsequently, these patterns aid to manage the resources and protect them from simultaneous access. The list of design patterns related to Concurrency pattern class is as follows.

- i. Message Queuing Pattern
- ii. Interrupt Pattern
- iii. Guarded Call Pattern
- iv. Rendezvous Pattern
- v. Cyclic Executive Pattern

- vi. Round Robin Pattern
- vii. Static Priority Pattern
- viii. Dynamic Priority Pattern

D. Distribution Pattern Class

The aim of Distribution design patterns is to describe the policies, structure and procedures to manage the symmetric and asymmetric distribution as an important aspect of system architecture. The list of design patterns related to Distribution pattern class is as follows.

- i. Shared Memory Pattern
- ii. Remote Method Call Pattern
- iii. Observer Pattern
- iv. Data Bus Pattern
- v. Proxy Pattern
- vi. Broker

E. Safety and Reliability

The aim of Distribution design patterns is to ensure the safety and reliability of a system in terms of addressing the consequences of failure. The list of design patterns related to Safety and Reliability pattern class is as follows.

- i. Protected Single Channel Pattern
- ii. Homogenous Redundancy Pattern
- iii. Heterogenous Redundancy Pattern
- iv. Monitor-Actuator Pattern
- v. Sanity Check Pattern
- vi. Watchdog Pattern
- vii. Safety Executive Pattern

This case study includes 1271 non-repeated words of all 34 design patterns after removing the stop words and words stemming [12].

4.2.3. Security Design Pattern Collection

The Security design Patterns are described to target the certain issues occurs in the context of the development of security based applications. The Security pattern collection includes 46 security systems relevant design patterns which are divided into eight categories named SACA (System Access and Control Architecture), ACM (Access Control Model), IA (Identification and Authentication), OSAC (Operating System Access Control), SIA (Security Internet Application), FA (Firewall Architecture), ESRM (Enterprise Security and Risk Management) and Accounting [15]. The list of pattern classes with corresponding design patterns is as follows [15].

A. Enterprise Security and Risk Management

The aim of Enterprise Security and Risk Management design patterns is to address enterprise-wide security issues. The list of design patterns related to Enterprise Security and Risk Management pattern class is as follows.

- i. Security Needs Identification of Enterprise Assets
- ii. Asset Valuation
- iii. Threat Assessment
- iv. Vulnerability Assessment
- v. Risk Determination
- vi. Enterprise Security Approaches
- vii. Enterprise Security Services
- viii. Enterprise Partner Communication

B. Identification and Authentication (I&A)

The aim of Identification and Authentication (I&A) design patterns is to address the certain aspects of I&A service. The list of design patterns related to Identification and Authentication (I&A) pattern class is as follows

- i. I&A Requirements
- ii. Automated I&A Design Alternatives
- iii. Password Design and Use
- iv. Biometrics Design Alternatives

C. Access Control Models

The aim of Access Control Models design patterns is to address the security Requirements besides functional specifications at all stages of system design. The list of design patterns related to Access Control Models pattern class is as follows

- i. Authorization
- ii. Role-Based Access Control
- iii. Multilevel Security
- iv. Reference Monitor
- v. Role Rights Definition

D. System Access Control Architecture

The aim of System Access Control Architecture design patterns is to check the access control security service of the system. The list of design patterns related to System Access Control Architecture pattern class is as follows

- i. Access Control Requirements
- ii. Single Access Point
- iii. Check Point
- iv. Security Session
- v. Full Access with Errors

- vi. Limited Access

- E. Operating System Access Control

The aim of Operating System Access Control design patterns is to support the execution of applications under the operating system constraints. The list of design patterns related to Operating System Access Control pattern class is as follows

- i. Authenticator
- ii. Controlled Process Creator
- iii. Controlled Object Factory
- iv. Controlled Object Monitor
- v. Controlled Virtual Address Space
- vi. Execution Domain
- vii. Controlled Execution Environment
- viii. File Authorization

- F. Accounting

The aim of Accounting design patterns is to manage the security events violation which occurs during the working activities. The list of design patterns related to Accounting pattern class is as follows

- i. Security Accounting Requirements
- ii. Audit Requirements Firewall Architectures
- iii. Audit Trails and Logging Requirements
- iv. Intrusion Detection Requirements
- v. Non-Repudiation Requirements

- G. Firewall Architectures

The aim of Firewall Architectures design patterns is to manage the security events violation which occurs during the working activities. The list of design patterns related to Firewall Architectures pattern class is as follows

- i. Packet Filter Firewall
- ii. Proxy-Based Firewall
- iii. Stateful Firewall

- H. Secure Internet Applications

The aim of Secure Internet Applications design patterns is to manage the internet applications. The list of design patterns related to Secure Internet Applications pattern class is as follows

- i. Information Obscurity
- ii. Secure Channels
- iii. Known Partners
- iv. Demilitarized Zone
- v. Protection Reverse Proxy
- vi. Integration Reverse Proxy

vii. Front Door

This case study includes 1230 non-repeated words of all 34 design patterns after removing the stop words and words stemming.

4.2.4. Duyne's HCI Design Pattern Collection

The Duyne's design Patterns are described to target the certain issues occurs in the context of the development of web based applications. The Duyne's HCI collection includes 90 patterns which are divided into 12 groups named Site Genres, Navigation Framework, Powerful Homepage, Writing and Managing Content, Building Trust and Credibility, Basic e-Commerce, Advance e-commerce, Helping Customers Complete Tasks, Effective Page layout, Fast Site Research, Making Easy Navigation and Speeding up site [13]. The list of pattern classes with corresponding design patterns is as follows.

A. Site Genere

The aim of Site Genere design patterns is to give a framework to construct different types of web sites and enable you how to communicate with your customers. The list of design patterns related to Site Genere pattern class is as follows.

- i. Personal E-Commerce
- ii. News Mosaics
- iii. Community Conference
- iv. Self Service Government
- v. Non Profits as Networks of Help
- vi. Grassroots Information Sites
- vii. Valuable Company Sites
- viii. Educational Forums
- ix. Stimulating Arts and Entertainment
- x. Web Apps That Work
- xi. Enabling Intranets

B. Creating a Navigation Framework

The aim of Creating a Navigation Framework design pattern is to maximize the flexibility of websites to accommodate the navigation, searching, and browsing habits of website customers. The list of design patterns related to Creating a Navigation Framework pattern class is as follows.

- i. Multiple ways to navigate
- ii. Browseable Content
- iii. Hierarchical Organization
- iv. Task Based Organization
- v. Alphabetical Organization

- vi. Chronological Organization
- vii. Popularity Based Organization
- viii. Category pages
- ix. Site Accessibility

C. Creating a Powerful homepage

The aim of Creating a Powerful homepage design pattern is to define that how homepage of a website is designed to fit the requirements of website customers. The list of design patterns related to Creating a Powerful homepage pattern class is as follows.

- i. Home Page Portal
- ii. Up-Front Value Proposition

D. Writing and Managing Content

The aim of Writing and Managing Content design patterns is to present an overview to help us in managing the content of a website effectively. The list of design patterns related to Writing and Managing Content pattern class is as follows.

- i. Page Templates
- ii. Content Modules
- iii. Headline and Blurbs
- iv. Personalized Content
- v. Message Boards
- vi. Writing for Search Engines
- vii. Inverse Pyramid Writing Style
- viii. Printable Pages
- ix. Distinctive TML Titles
- x. Internationalized and Localized Content

E. Building trust and Credibility

The aim of Building trust and Credibility design patterns is to give an overview of certain issues related to customer's trust and credibility about the use of a website. The list of design patterns related to Building trust and Credibility pattern class is as follows.

- i. Site Branding
- ii. E-mail Subscriptions
- iii. Fair Information Practices
- iv. Privacy Policy
- v. About Us
- vi. Secure Connection

F. Basic E-Commerce

The aim of Basic E-Commerce design patterns is to give a way how to create the customer's experience effect on e-commerce web sites. The list of design patterns related to Basic E-Commerce pattern class is as follows.

- i. Quick Flow Checkpoint
- ii. Clean Product Details
- iii. Shopping Cart
- iv. Quick Address Selection
- v. Quick Shipping Method Selection
- vi. Payment Method
- vii. Order Summary
- viii. Order Confirmation and Thanks you
- ix. Easy Returns

G. Advances E-Commerce

The aim of Advances E-Commerce design patterns is to discuss the advanced and optional features which can be included in a website for sending gifts, order tracking or promoting products. The list of design patterns related to Advances E-Commerce pattern class is as follows.

- i. Feature Products
- ii. Cross Selling and Up Selling
- iii. Personalized Recommendations
- iv. Recommendation Community
- v. Multiple Destinations
- vi. Gift Giving
- vii. Order Tracking and History

H. Helping Customers Complete Tasks

The aim of Helping Customers Complete Tasks design patterns is to present the ways how a website can be organized to decrease the problems and improve the rate about the task completion. The list of design patterns related to Helping Customers Complete Task pattern class is as follows.

- i. Process Funnel
- ii. Sign In/New Account
- iii. Guest Account
- iv. Account Management
- v. Persistent Customer Sessions
- vi. Pop-up Windows

vii. Frequency Asked Questions

viii. Context Sensitive Help

I. Designing Effective Page Layouts

The aim of Designing Effective Page Layouts design patterns is to describe the ways that how a clear, predictable, and easy to understand layout of web pages can be created. The list of design patterns related to Designing Effective Page Layouts pattern class is as follows.

- i. Grid Layout
- ii. Above the Fold
- iii. Clear First Reads
- iv. Expanding Width Screen Size
- v. Fixed Width Screen Size
- vi. Consistent Sidebars of Related Content

J. Making Site Search Fast and Relevant

The aims of Making Site Search Fast and Relevant design patterns is to give different way which can aid to improve the designing of interaction to make customer's search more effective. The list of design patterns related to Making Site Search Fast and Relevant pattern class is as follows.

- i. Search Action Module
- ii. Straightforward Search Forms
- iii. Organized Search Result

K. Making Navigation Easy

The aim of Making Navigation Easy design patterns is to give a framework to present different ways for organizing and displaying the navigational elements of websites. The list of design patterns related to Creating a Making Navigation Easy pattern class is as follows.

- i. Unified Browsing Hierarchy
- ii. Navigation Bar
- iii. Tab Rows
- iv. Action Buttons
- v. High Visibility Action Buttons
- vi. Location Bread Crumbs
- vii. Embedded Links
- viii. External Links
- ix. Descriptive Longer Link names
- x. Obvious Links
- xi. Familiar Language

- xii. Preventing Errors
- xiii. Meaningful Error Messages
- xiv. Page Not Found

L. Speeding Up your site

The aim of Speeding Up your site design patterns is to enhance the look and feel of a website. The list of design patterns related to Speeding Up your site pattern class is as follows.

- i. Low Numbers of Files
- ii. Fast Downloading Images
- iii. Separate Tables
- iv. HTML Power
- v. Reusable Images

This case study includes 1575 non-repeated words of all 90 design patterns after removing the stop words and words stemming [13].

4.3. Real Design Problems

In order to evaluate the effectiveness of our proposed framework, we retrieved 100 real design problems from different sources [9, 156, 157, 158, 159]. We retrieved the design problems which have been addressed and mapped according to expert opinions, which can aid us to investigate the authenticity of the proposed framework. The reference of resources, authors who address and mapped the problems, and the involvement of the community is shown in Table 4.1. We have provided the description of design problems in Appendix B to evaluate the effectiveness of the proposed method.

Table 4.1. Resource information about design problems

Reference of Resource	Authors	Involved Participants	Related Pattern Collection
Bouhour et al. [9]	Bouhour, Leblance, and Percebois	126	Gang-of-Four patterns
Silberschatz et al. [156]	Silberschatz, Galvin, and Gagne	3	Security Patterns
Tanenbaum et al. [157]	Tanenbaum and H. Bos	2	Real Time Design Patterns
Shalloway et al. [158]	Shalloway and Trott	3	Gang-of-Four patterns
Shvets [159]	Shvets	1	Gang-of-Four patterns

5.1. Experiment Analysis

The aim of this chapter is to describe the implementation procedure of the proposed approach and provide guidelines for readers to replicate the implementation procedures in other domains. Firstly, three pseudocodes are described to provide the guidelines to readers that how the proposed approach is employed in order to achieve the target objectives which are mentioned in section 1.4. These pseudocodes aid to provide guidelines for readers to replicate the proposed study. Subsequently, the flow of each pseudocode consists of certain tasks, such as in pseudocodes (Section 5.1.1 and Section 5.1.2), the selection of best weighting and feature selection methods. Consequently, secondly, numerous experiments are performed to target these tasks and evaluate the effectiveness of the proposed approach by benchmarking the experimental results. These experiments are grouped according to the targeted tasks such as 1) to determine the best weighting method for each unsupervised learning technique for the given pattern collection, 2) to recommend the outperform unsupervised learning technique for the organization of design patterns according to actual classification schemes presented through the expert opinions, 3) to recommend the outperform unsupervised learning technique in order to determine the appropriate pattern class for the given sample of design problems, 4) to select the best feature selection method for the unsupervised learning technique, and 5) to recommend the right design patterns for the given sample of design problems.

5.2. Pseudocodes to Describe the Implementation Procedures

In this section, three pseudocodes are described to provide the guidelines to readers that how the proposed approach is employed in order to achieve the target objectives. The first pseudocode is used to organize the design pattern of any target design pattern collection. Subsequently, the second pseudocode is used to find a right design pattern class for a given design problem. Finally, the third pseudocode is used to select a right design pattern from the list of a candidate pattern class.

5.2.1. Pseudocodes for the Organization of Design Patterns

The aim of first pseudocode (i.e. Pseudocode-1) is to describe the flow of activities which are performed to achieve the target objective that is the implementation of the proposed approach for the organization of design patterns. The pseudocode-1 describes the procedure for the organization of design patterns of a target collection as follows.

Pseudocode-1 Pattern_Organization (PC, USC, k, N, M)

Input:

PC = {Set of problem domain description of design patterns in the given Pattern Collection}

USL = {Set of UnSupervised Learners under study, such as Fuzzy c-means, K-means, agglomerative, Partitioning Around Medoids (a.k.a. K-Medoids with distance measures Euclidean and Manhattan)}

k = The number of clusters to organize the pattern catalog, which are recommended through an expert opinion, For example, in the case of GoF (k=3) and Douglass (k=5).

N = The number of design patterns in a collection, For example, in case of GoF and Douglass pattern collection the value of N is 23 and 34 respectively.

M = The number of unique words, For example, in case of GoF and Douglass pattern collection the value of M is 1465 and 1271 respectively.

Procedure Begin:

1. Perform first three activities of preprocessing phase of the proposed framework to construct a Support Vector Model (SVM) with N design patterns and M Unique words.
2. Apply a weighting method (such as Binary, TF, TFIDF, TFC, LTC, and Entropy) to constructed SVM.
3. Apply an unsupervised learning technique to organize the N design patterns.
4. Apply the evaluation criteria (F-measure in the section 4.1.1) to assess the performance of the unsupervised learner with corresponding weighting method and k clusters.
5. Repeat Steps 2, 3 and 4 until the all possible assessments have done.
6. For each unsupervised learner select the best weighting method, and classify the patterns with different number of clusters (For example, the value of k varies between 1 to 10)
7. Apply the evaluation criteria (ARI in the section 4.1.1) to assess the performance of each unsupervised learner with k.
8. Select the outperform unsupervised learner.
9. In order to evaluate the improvement in the effectiveness of outperform unsupervised learner, apply a feature selection method (Such as Document Frequency (DF), Information Gain (IG), Mutual Information (MI), Correlation Coefficient (CC), Chi-Square (CHI), and Ensemble-IG) to rank the top n features.
10. The unsupervised learner with k cluster is applied on the top n features and evaluate the performance in terms of F-measure.

Procedure End**Output:**

The outperform unsupervised learner with best weighting and feature selection method for the organization of design patterns catalog (i.e. PC).

5.2.2. Pseudocodes for the Identification of appropriate Pattern Class for the given design problems

The aim of second pseudocode (i.e. Pseudocode-2) is to describe the flow of activities which are performed to achieve the target objective that is the implementation of the proposed approach for determination of appropriate pattern classes for the sample of given design problems. The identification of appropriate pattern class for a given design problem via the proposed approach is described through pseudocode-2 as follows.

Pseudocode-2 Determing_Pattern_Class_For_Problem (PC, DP, USC, k, N, M)

Input:

PC = {Set of problem domain description of design patterns in the given design pattern Collection}

DP = {Set of descriptions of the given design problems described in the context of target pattern collection}

USL = {Set of Unsupervised Learners under study, such as Fuzzy c-means, K-means, agglomerative, Partitioning Around Medoids (a.k.a. K-Medoids with distance measures Euclidean and Manhattan)}

k = The number of clusters to organize the pattern catalog, which are recommended through an expert opinion, For example, in the case of GoF (k=3) and Douglass (k=5).

N = The number of design patterns in a collection and a design problem, For example, in case of GoF N will equal to 24 (i.e. 23 design patterns and 1 design problem)

M = The number of unique words

Procedure Begin

1. Perform first three activities of preprocessing phase of the proposed framework to construct a Support Vector Model (SVM) with N+1 (i.e. N design patterns and 1 target design problem) feature vectors and M Unique words.
2. Perform the Step-2 to Step-6 of pseudocode-1 to select the best weighting method of each unsupervised learner.
3. Apply the evaluation criteria (ARI in the section 4.1.1) to assess the performance of each unsupervised learner with k (e.g. In the case of GoF pattern collection, the value of k is 3)
4. Select the outperform unsupervised learner in order to determine the appropriate pattern class for the given design problem.

Procedure End

Output:

The outperform unsupervised learner with the best weighting method to identify the right pattern class (i.e. Category) for the given sample of design problems (which are already mapped into appropriate categories according to expert opinions). For example, in the case of Design problem 1(DP-1), the Behavioral pattern class is the right choice according to expert opinion.

5.2.3. Pseudocodes for the Selection of Design Patterns for the given design problems

The aim of third pseudocode (i.e. Pseudocode-3) is to describe the flow of activities which are performed to achieve the target objective that is the selection of right design patterns the given sample of the given design problems. The implementation procedure for the selection of right design pattern(s) for a given design problem via the proposed approach is described the pseudocode-3 as follows.

Pseudocode-3 Pattern_Selection_For_Problem (Pattern_Vectors, Problem_Vector)

Input:

Pattern_Vectors = {Set of the feature vectors for design patterns of a pattern class recommended for the target design problem}

Problem_Vector = Describe the feature vector for target design problem

Procedure Begin

1. The equation 3.5 (for cosine similarity measure in the section 3.1.4) can be used to measure the closeness of patterns with the given design problems.
2. The similarity options are assessed to select the right design pattern(s).
3. The equation 3.6 described the first similarity option (Section 3.1.4) which aids to recommend the right design pattern for a given design problem on the base of cosine similarity value. For example, in the case of Design Problem 1 (Table C-1), the Chain of Responsibility is a recommended pattern with highest cosine similarity value (i.e. 0.54) as compared to other patterns of the same class.
4. The equation 3.7 described the second similarity option (Section 3.1.4) which aids to recommend zero, one or more than design patterns for a given design problem based on the suggested threshold (Θ_1) value, which might be an average of the cosine similarity values. For example, in the case of Design Problem 1 (Part A of Appendix B), the suggested threshold value is 0.31 to 0.53 (Table 6.3) to recommend the zero, one or more than one design pattern(s). The number of recommended patterns depends on the selected value of the suggested threshold range. For example, in case of Θ_1 (less than 0.31) more than one right design pattern can be recommended. Subsequently, in the case of Θ_1 greater than 0.53, no pattern is recommended.
5. The equation 3.8 described the third similarity option (Section 3.1.4) which aids to recommend design patterns contiguous to the best design pattern (with CS_{max}) based on the suggested threshold (Θ_2) for cosine similarity values. For example, in the case of Design Problem 1 (Part A of Appendix B), the suggested threshold value is 0.0 to 0.22 (Table 6.3). The number of suggested patterns depends on the selected value from the suggested range of threshold (Θ_2). For example, in case of $\Theta_2 = 0.22$ Mediator and Chain of Responsibility are the recommended patterns. Subsequently, in case of $\Theta_1 = 0.0$, Chain of Responsibility pattern is recommended (i.e. Same case as of Step-1).
6. Finally, the equation 4.6 is used to select the best similarity option (Section 3.1.4) by considering the ratio of the number of right suggested and total suggested design patterns. For example, on the sample of ten

design problems (Part A of Appendix B), the first similarity option (from equation 3.6 in the Section 3.1.4) with highest RCD (Ratio of Corrected Design patterns) is recommended.

Procedure End

Output:

The recommendation of right design pattern(s) from the patterns list (Pattern_Vectors) of determined pattern class with respect to content similarity for the given design problem (i.e. Problem_Vector) and best similarity option is suggested.

5.3. Detail Analysis of Implementing Tasks Described in Pseudocodes

The pseudocodes described in the section 5.2 have some common tasks which are performed during the experiments, such as the selection of best weighting and feature selection methods, and selection of outperform unsupervised learning technique. The detail of experiments to perform these tasks is as follows.

5.3.1. Experimental Procedure for the Selection of Best weighting method (Experiment 1)

The performance of supervised and unsupervised learning techniques are highly correlated with the characteristics of target datasets besides their types. Consequently, the best weighting method for an unsupervised learning technique is decided based on their effectiveness with the corresponding weighting methods. In this study, we evaluate the effectiveness of each unsupervised learning technique (employed via the proposed approach) with five weighting methods named Binary, Term Frequency (TF), Term Frequency Inverse Document Frequency (TF/IDF), Term Frequency Collection (TFC), Length Term Collection (LTC), and Entropy. The weighting method with which an unsupervised learner obtains the highest F-measure is recommended as a best weighting method. However, its depend on the target corpus (i.e. Design pattern collection). The experimental procedure for the selection of best weighting method for each unsupervised learning technique is shown in the Figure 5.1.

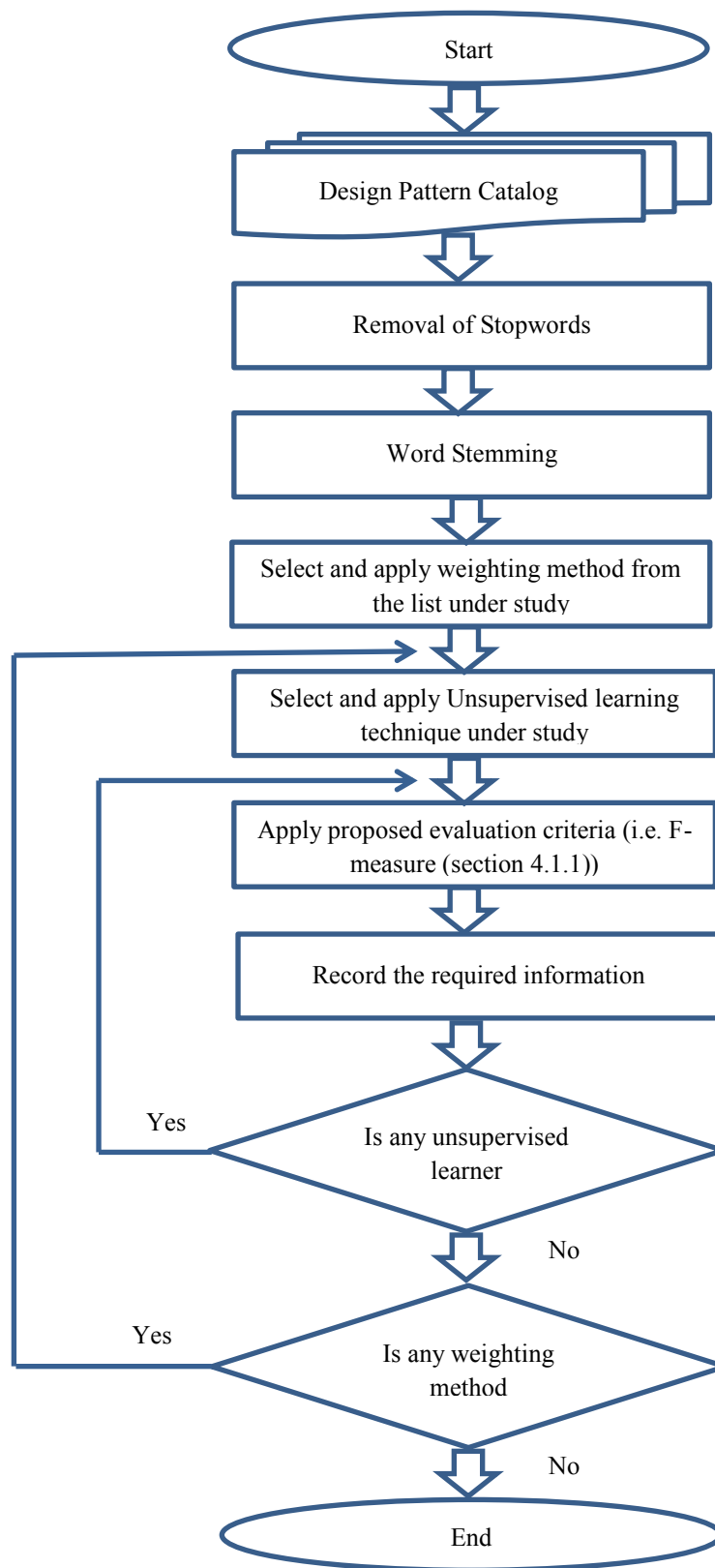


Figure 5.1. Flow of activities performed in the Experiment-1

5.3.2. Experimental Procedure for the Selection of Outperform Unsupervised Learning Technique in Terms of organization of design patterns (Experiment 2)

The aim of this experiment is to determine the outperformed unsupervised learning technique in terms of organization of design patterns of a target collection. Firstly, a design pattern collection is selected. Secondly, the best weighting method is selected for each unsupervised learning technique (Using Experiment 1). Third, the performance of each unsupervised learner is evaluated in terms of the Adjusted Rand Index (ARI) (Section 4.1.1). Finally, an unsupervised learner with highest ARI is selected as an outperform unsupervised learner for the target design pattern collection. The procedure of the experiment is shown in the Figure 5.2.

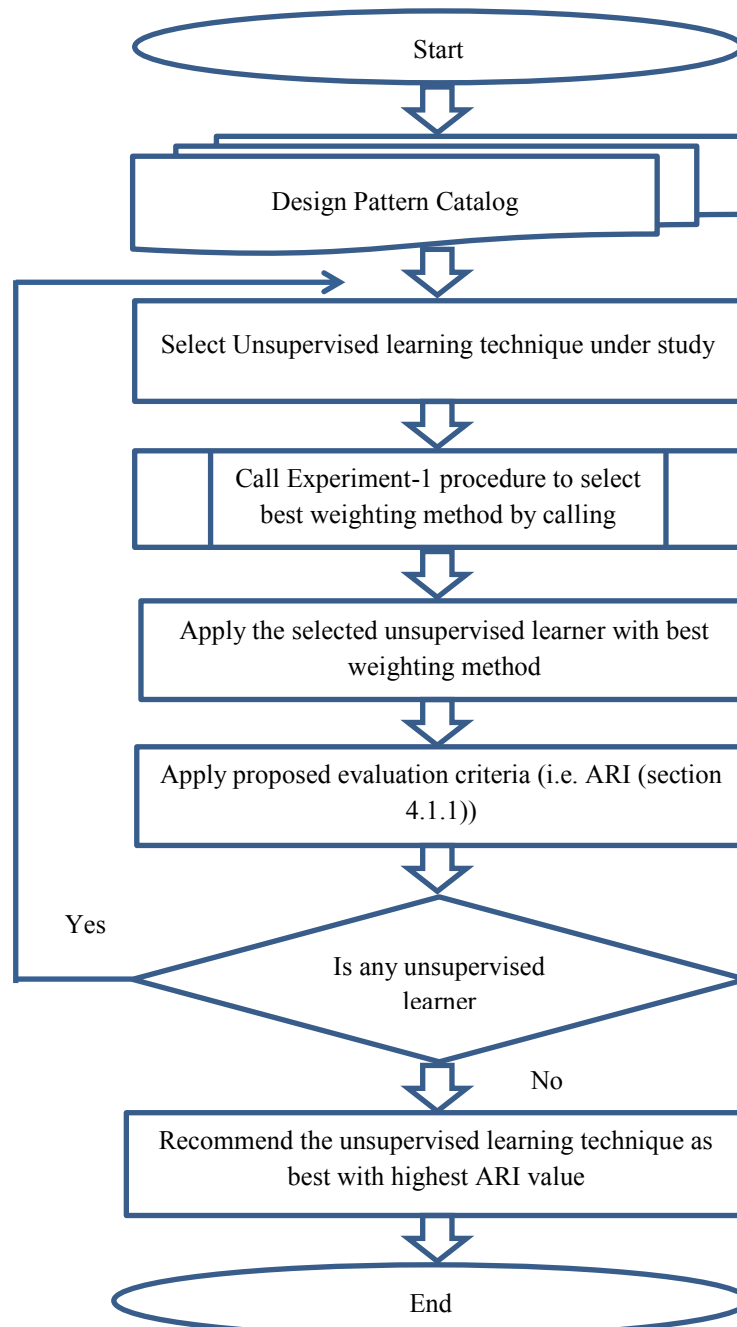


Figure 5.2. Flow of activities performed in the Experiment-2

5.3.3. Experimental Procedure to determine the effectiveness of unsupervised learner for pattern organizations with respect to the classification scheme of experts. (Experiment 3)

The aim of this experiment is to determine the effectiveness of the proposed approach for the organization of

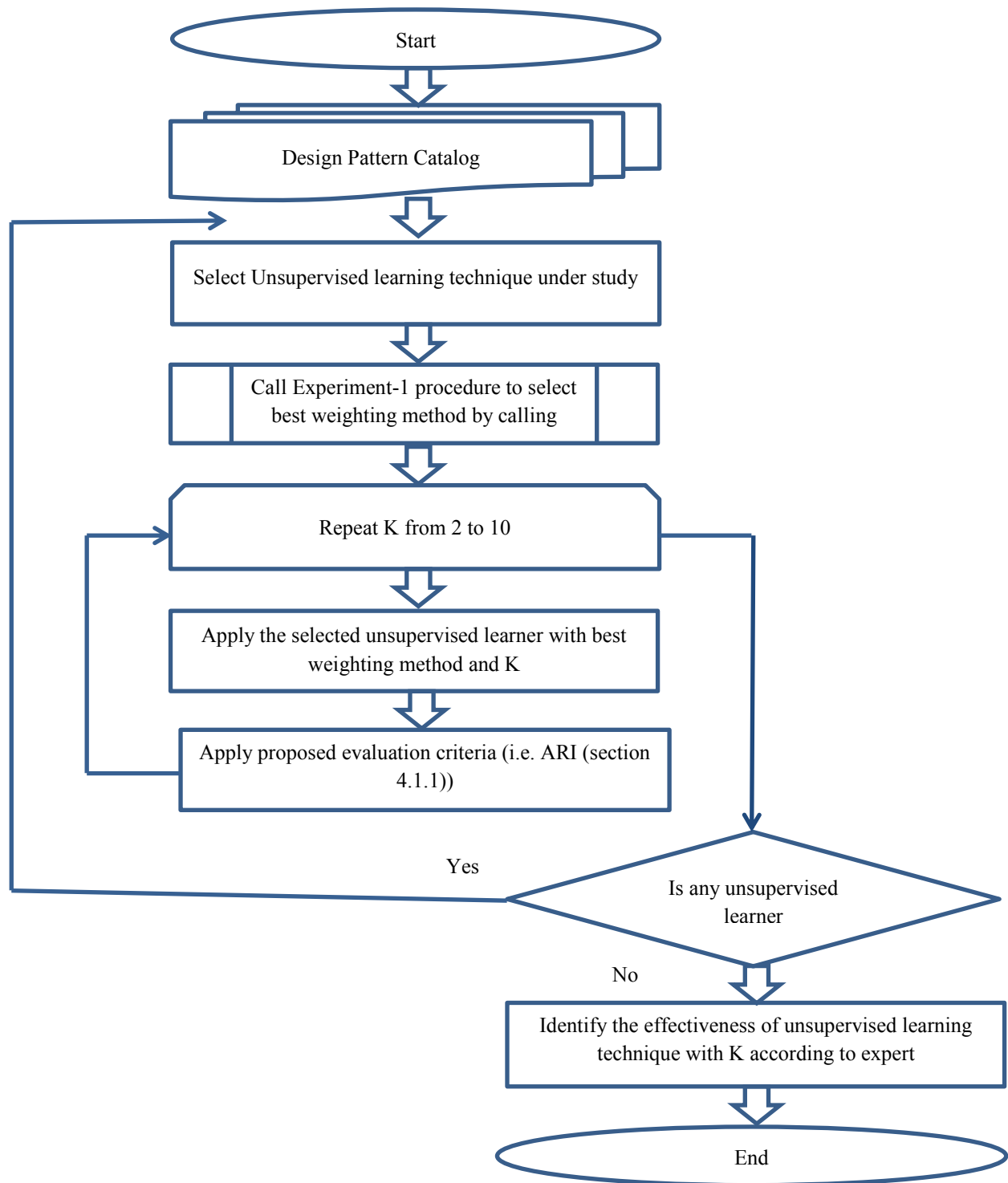
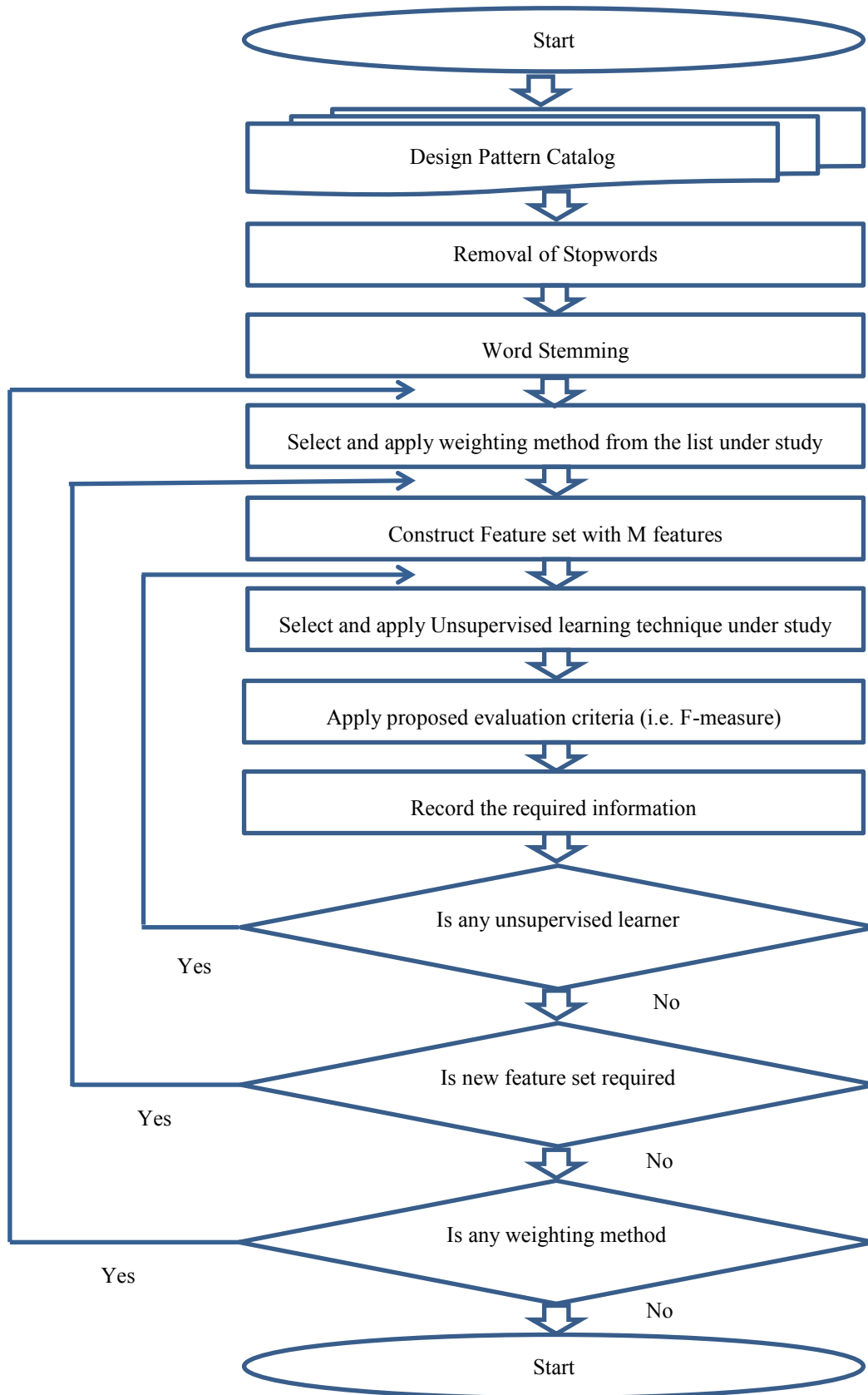


Figure 5.3. Flow of activities performed in the Experiment-3

**Figure 5.4.** Flow of activities performed in the Experiment-4

design pattern according the classification scheme of experts of the target pattern collection. For example, in the case of GoF design pattern collection, the authors have organized the 23 design patterns into three categories named Creational, Structural, and Behavioral. The effectiveness of each unsupervised learner of the proposed approach depends on the appropriate grouping of all design patterns into 3 categories (i.e. $k=3$ for each unsupervised learner). The procedure of the experiment is shown in the Figure 5.3.

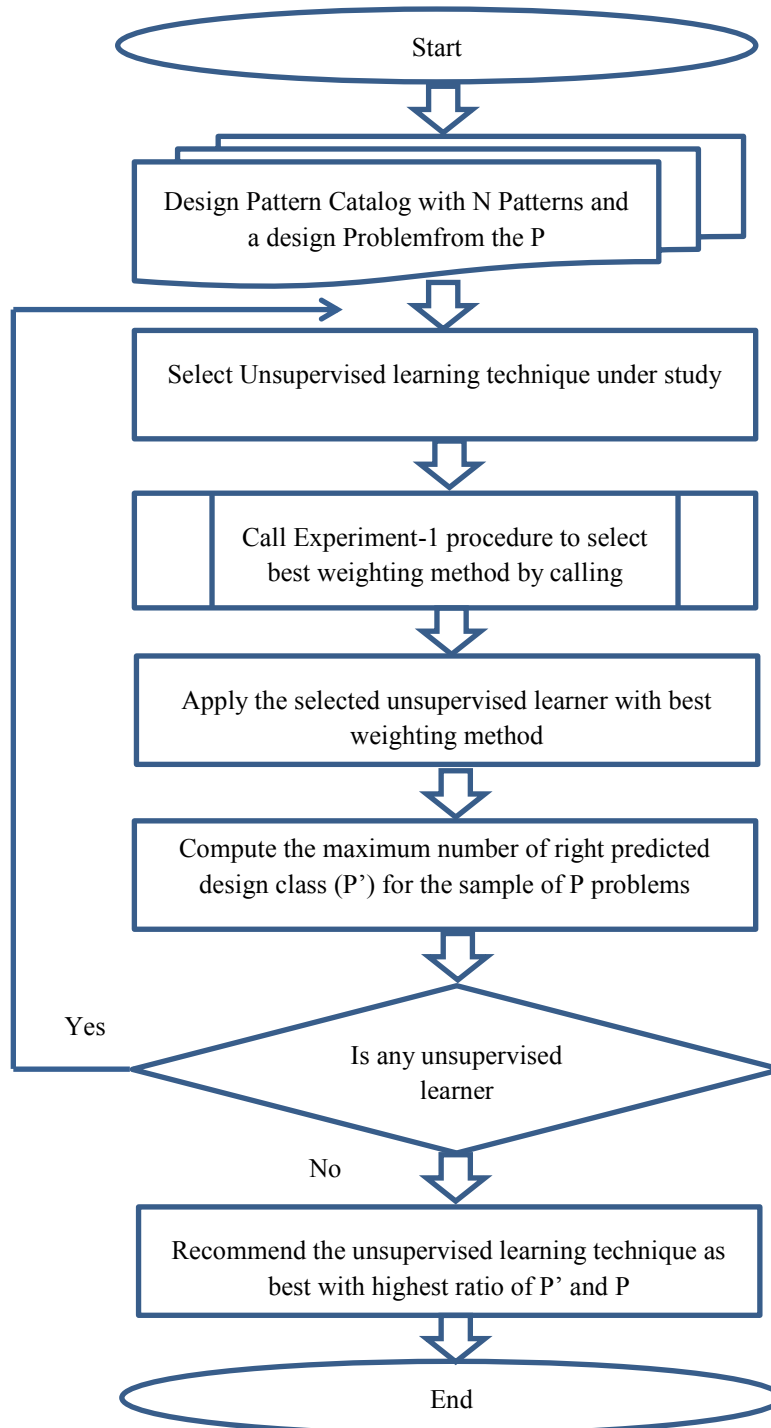


Figure 5.5. Flow of activities performed in the Experiment-5

5.3.4. Experimental Procedure for the Selection of best feature selection method (Experiment 4)

The undesirable effect of useless features also affects the performance of either supervised or unsupervised learning techniques. The aim of this experiment is to investigate the effect of feature selection on the performance of unsupervised learning technique under study and recommend the best feature selection method on the target design pattern collection. Firstly, a design pattern collection is selected. Secondly, the best weighting method is selected to weight the terms of the target corpus. Third, a feature selection method is selected to construct a feature set with M features. Fourth, the performance of each unsupervised learner is evaluated in terms of F-measure (Section 4.1.1). The same process is repeated for each unsupervised learner with all weighting methods and feature selection methods. The procedure of the experiment is shown in the Figure 5.4.

5.3.5. Experimental Procedure for the Selection of Outperform Unsupervised Learning Technique in terms of identification of pattern class for the sample of design problems (Experiment 5)

The aim of this experiment is to determine the outperform unsupervised learning technique in terms of identification of appropriate design pattern class for the given design problem. Subsequently, this experiment is repeated with certain sample of design problems related to the target design pattern collection. Firstly, the a design pattern collection is with N design patterns and a sample of P related design problems are selected. Secondly, the best weighting method is selected for each unsupervised learning technique (Using Experiment 1). The procedure of the experiment is shown in the Figure 5.5. Third, the performance of each unsupervised learner is evaluated in terms of determining the maximum number of right design pattern class (i.e. P') for the given sample of design problem. Finally, an unsupervised learner with highest ratio of P' and P (i.e. P'/P) is selected as an outperform unsupervised learner for the target design pattern collection. The results of this experiment depend on the sample of design problems and the target design pattern collection.

5.3.6. Experimental Procedure for the Selection of Outperform Unsupervised Learning Technique in Terms of predicting appropriate design pattern for the sample of design problems P (Experiment 6)

The aim of this experiment is to determine the outperform unsupervised learning technique in terms of identification of appropriate design patterns for the given design problem. Subsequently, this experiment is repeated with certain sample of design problems related to the target design pattern collection. Firstly, a design pattern collection is with N design patterns and a sample of P related design problems are selected. Secondly, the best weighting method is selected for each unsupervised learning technique (Using Experiment 1). Third, the performance of each unsupervised learner is evaluated in terms of determining the maximum number of right design patterns (i.e. P') for the given sample of design problem. Finally, an unsupervised learner with highest ratio of P' and P (i.e. P'/P) is selected as an outperform unsupervised learner for the target design pattern collection. The results of this experiment depend on the sample of design problems and the target design pattern collection. The procedure of the experiment is shown in the Figure 5.6.

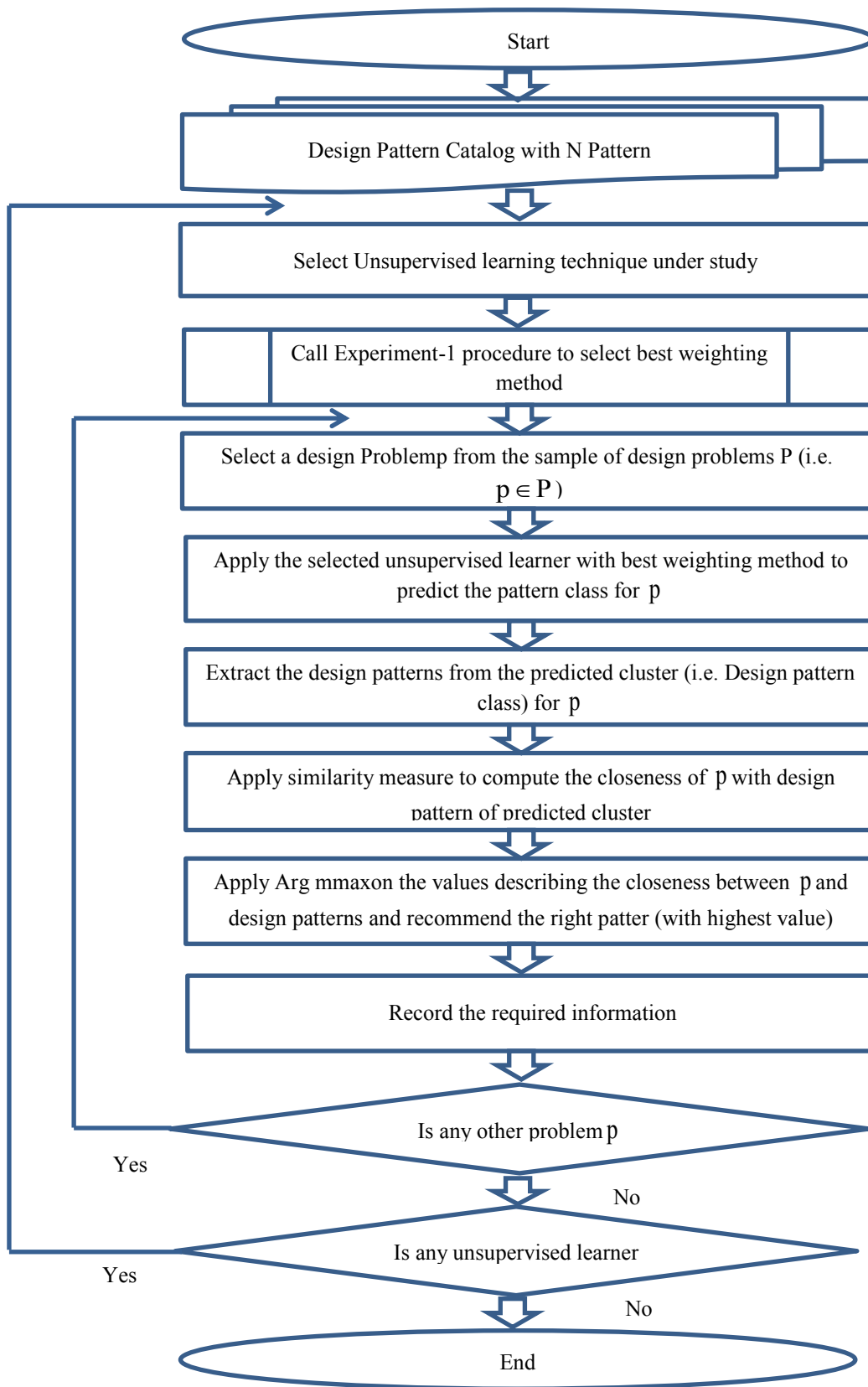


Figure 5.6. Flow of activities performed in the Experiment-6

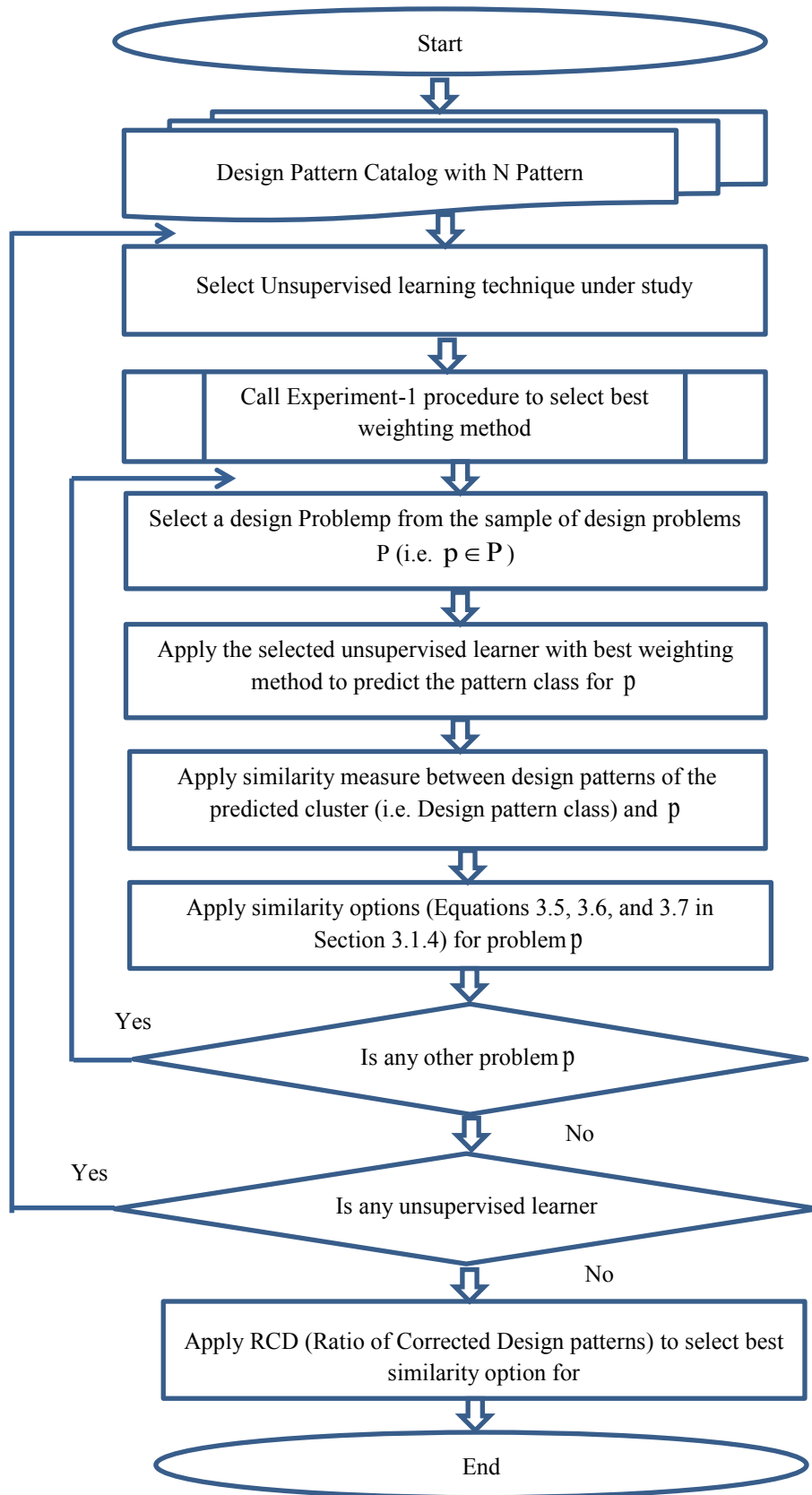


Figure 5.7. Flow of activities performed in the Experiment-7

5.3.7. Experimental Procedure for the Selection of best similarity option in terms of predicting appropriate design pattern for the sample of design problems P (Experiment 7)

In the proposed approach (Section 3.1.1), three similarity options are suggested to select the appropriate design pattern(s). The aim of this experiment is used to determine the best similarity option in terms of predicting the appropriate design patterns for the given design problems. Firstly, a design pattern collection is with N design patterns and a sample of P related design problems are selected. Secondly, the best weighting method is selected for each unsupervised learning technique (Using Experiment 1). Third, the unsupervised learners are applied to predict the appropriate design pattern class for each design problem. Fourth, similarity measure is applied to compute the closeness of each design problem with design patterns of suggested pattern class. Fifth. Similarity options are used to select the appropriate design patterns. Finally, the RCD (Ratio of Corrected Design patterns) measure is applied to select the best similarity option (Section 3.1.4). The procedure of the experiment is shown in the Figure 5.7.

6.1. Results and Discussion

The proposed approach is employed in the context of four widely used design pattern collections in different domains and many design problems which have been mapped and resolved by the research/developer community. The detail of design pattern collections and related design problems is given in the sections 4.2 and 4.3 respectively. In this regard, we present four case studies and each case study includes a design pattern catalog and a set of related problems. The Figure 6.1 describes the working procedure of the proposed approach.

In each case study, many experiments (Detail of experiments is given in chapter 5) are performed to describe the effectiveness of the proposed approach. The analysis of each case study is structured into three sections based on the target objectives of the proposed approach, which are.

1. Effectiveness of the proposed approach in terms of organization of software design pattern collection.

In this section, the effectiveness of the proposed approach is reported in terms of determining the

- a) Best weighting method for each unsupervised learning technique of the proposed approach.
- b) Outperform unsupervised learning technique in terms of organization of design patterns of target pattern collection.
- c) The effectiveness of outperform unsupervised learning technique in terms of organization of design patterns with respect to several categories suggested through the expert's classification scheme.
- d) The best feature set to improve the effectiveness of outperforms unsupervised learning technique in the context of the target design pattern collection.

2. Effectiveness of the proposed approach to determine the right pattern class for the sample of given design problems.

In this section, the effectiveness of the proposed approach is reported in terms of determining the outperform unsupervised learning technique in terms of organization of design patterns and identification of right design pattern class with respect to the sample of given design problems.

3. Effectiveness of the proposed approach in order to determine the right design patterns for the sample of given design problems.

In this section, the effectiveness of the proposed approach is reported in terms of determining the right design patterns for the sample of given design problems.

Subsequently, the results are discussed within the context of the target design pattern collection and sample of related design problems.

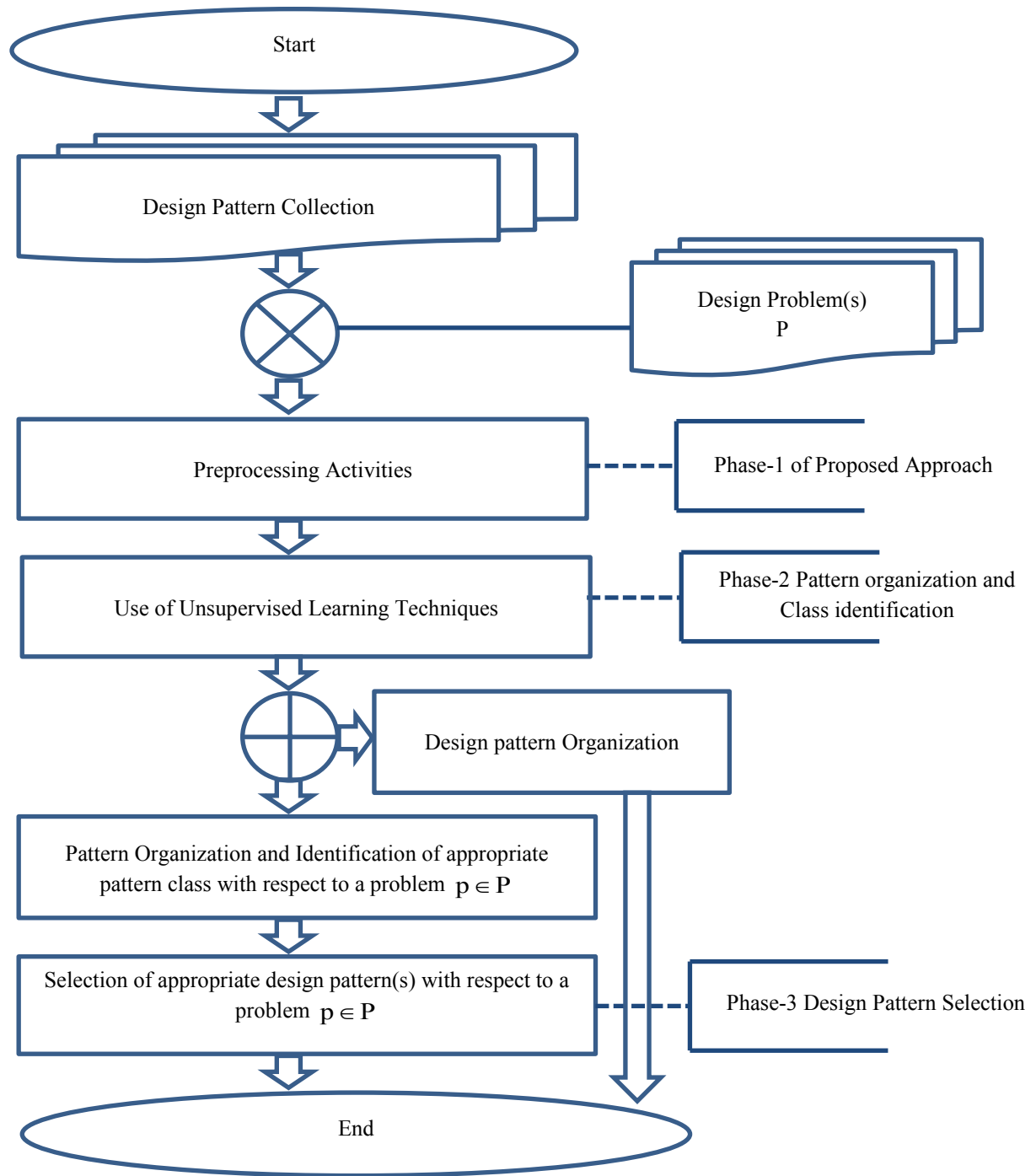


Figure 6.1. Working procedures of proposed approach in presented case studies

6.2. First Case Study (Gang-of-Four Design Patterns in the Domain of Object-Oriented Development)

The aim of first case study is to use the proposed approach in the context of software design patterns related to designing issues of object-oriented development. The Gang-of-Four (GoF) design patterns are widely used by object-oriented software development community (Detail is given in the section 4.2.1).

6.2.1. Organization of Design Patterns

Firstly, to determine the best weighting method for each unsupervised learning technique, the experiment 1 is performed according to given guidance in the section 5.3.1. The results are shown in the Figure 6.2.

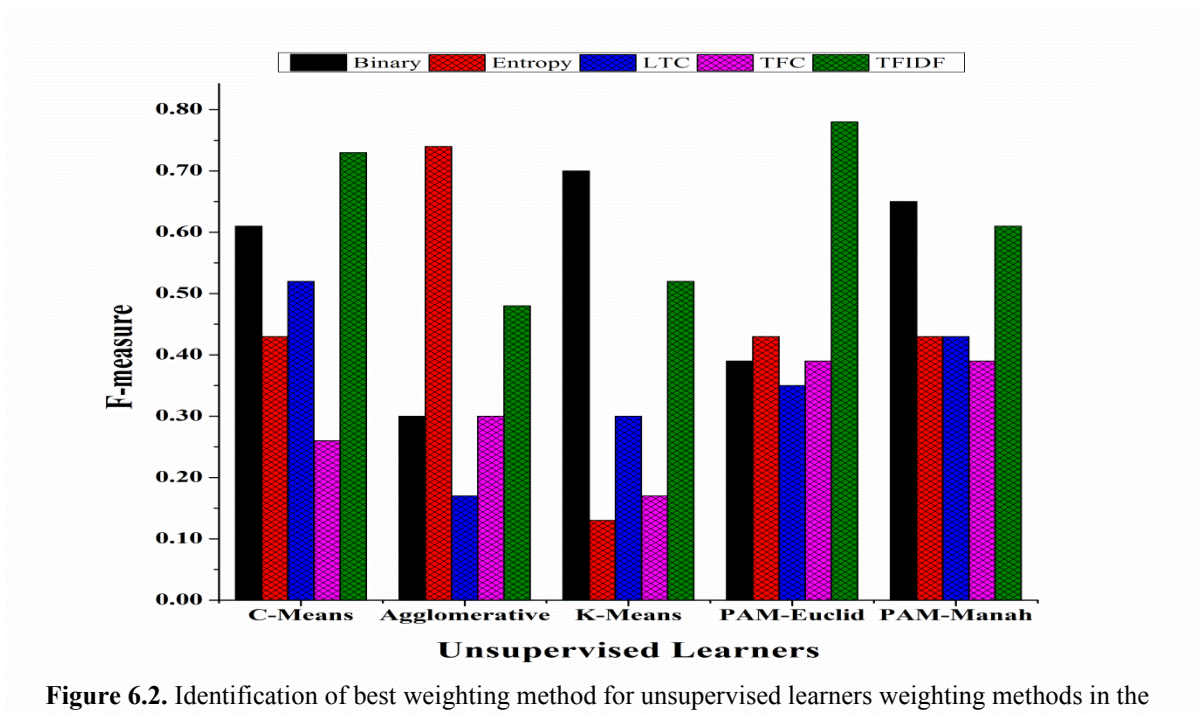


Figure 6.2. Identification of best weighting method for unsupervised learners weighting methods in the context of GoF design pattern collection

The weighting method with highest F-measure value of unsupervised learner is recommended as a best weighting method for the corresponding learner. For example, the TFIDF is recommended as a best weighting method with highest F-measure values 0.73 and 0.78 for Fuzzy C-means and PAM-Euclidean respectively. Subsequently, the Binary is recommended as a best weighting method with highest F-measure values 0.70 and 0.65 for K-means and PAM-Manhattan respectively (Response to RQ-4).

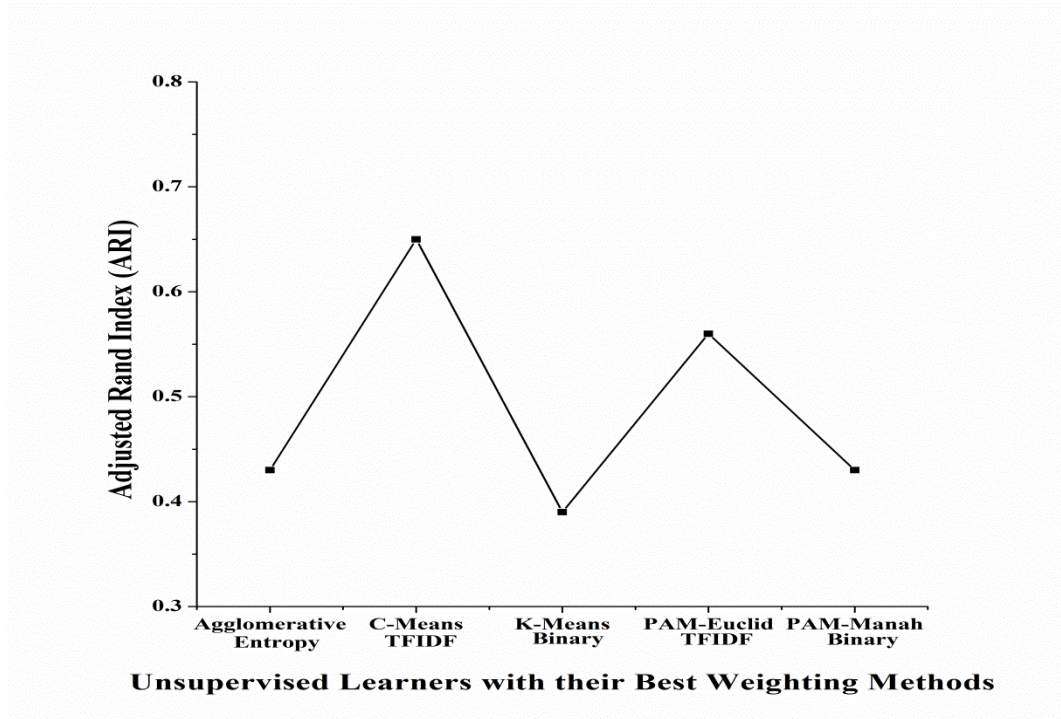


Figure 6.3. Identification of outperform unsupervised learner with best weighting method in the context of GoF design pattern collection

Secondly, in order to find the outperformed unsupervised learner, the experiment 2 is performed according to given guidance in the section 5.3.2. The performance of each unsupervised learner is analyzed with best weighting methods. The results are shown in the Figure 6.3. The result of Figure 6.3 indicates that Fuzzy c-means with highest ARI value (i.e. $ARI=0.65$) is recommended as an outperform unsupervised learner as compared to Agglomerative ($ARI=0.43$), K-means (0.39), PAM-Euclidean ($ARI=0.56$), and PAM-Manhattan ($ARI=0.43$) learners of the proposed approach.

Third, to find the effectiveness of the proposed approach in terms of organization of design patterns with respect to the classification scheme of experts, the experiment 3 is performed according to given guidance in the section 5.3.3. The performance of each unsupervised learner is analyzed with best weighting methods. The results are shown in the Figure 6.4. The highest ARI value of each unsupervised learner at $K=3$ (according to expert opinion) indicates the effectiveness of the proposed approach (Response to RQ-1).

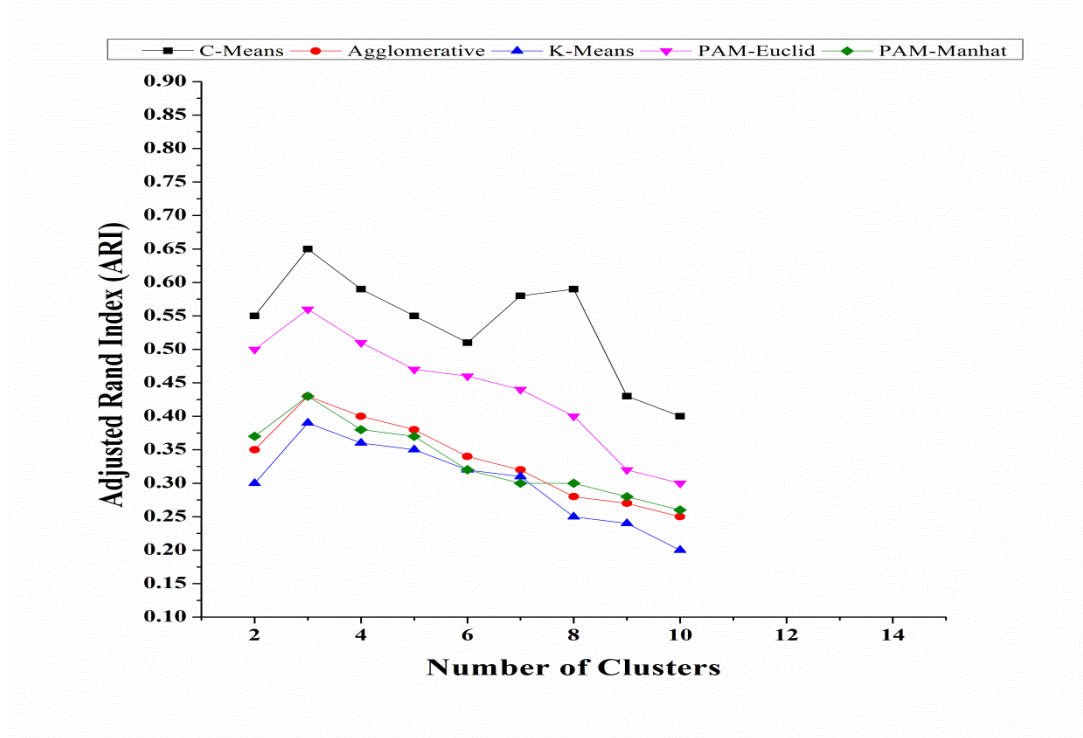


Figure 6.4. The performance of unsupervised learners with best weighting methods and K clusters in the context of GoF design pattern collection

Fourth, in order to investigate the effectiveness of feature selection methods on the performance of unsupervised learners of the proposed approach, the experiment 4 is performed according to given guidelines in the section 5.3.4. Therefore, we applied six feature selection methods Chi-Square, Document Frequency (DF), Correlation, Information Gain, Gain Ratio including our proposed method named Ensemble-IG (to overcome the multi-class problem) and evaluate their impact on the performance of Outperform unsupervised learner Fuzzy c-means. The evaluation results are shown in the Figure 6.5. The x-axis of Figure 6.5 (labeled with Number of Rank Features) refers to top n features rank by each feature selection method. According to F-measure criterion, the feature selection method with the highest increase in the performance of Fuzzy c-means as compared to its actual performance $F\text{-measure}=0.73$ before feature selection (illustrated in the Figure 6.2) is suggested. For example, with top 10 features (depicted on the x-axis of Figure 6.5) rank by each feature selection method, there is an increase from 6.84% to 15.06% (in term of F-measure) in the performance of Fuzzy c-means. The proposed feature selection method Ensemble-IG with the highest increase in the performance (i.e. 15.06%) is suggested for the fuzzy c-means. We have replicated this procedure with Top n (where n ranges from 50 to 350) features and illustrated the evaluation results in the Figure 6.5 (Response to RQ-4 and RQ-5).

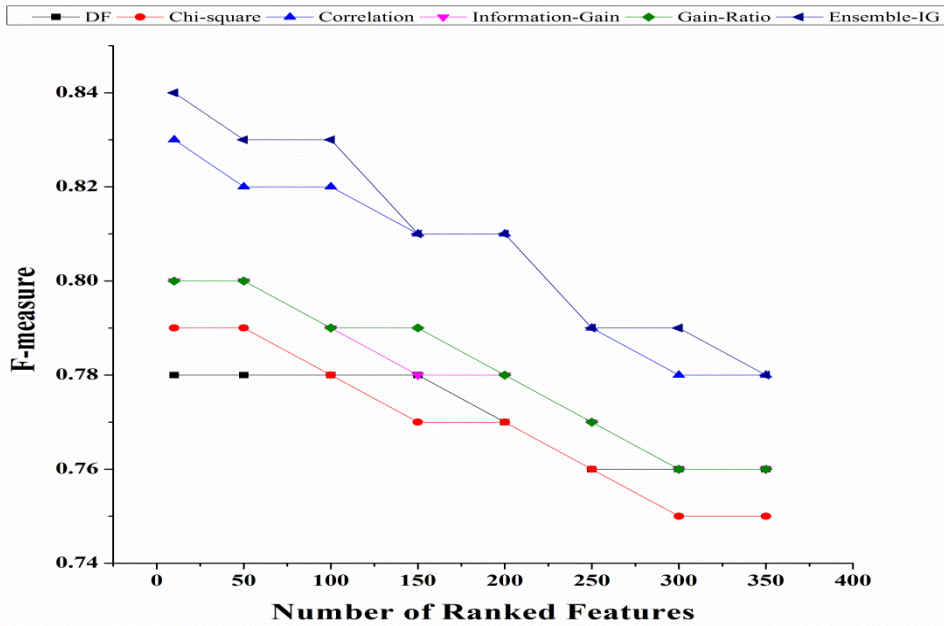


Figure 6.5. The effect of feature selection methods on the performance of outperform unsupervised learner C-means with best weighting method in the context of GoF design pattern collection

6.2.2. Determination of design pattern class for the sample of given design problems

The second objective of the proposed approach is to find the right design pattern class for the sample of given design problems. In this regard, the experiment 5 is performed. In study 53 design problems are considered. The experiment 5 is performed with two samples (i.e. P) of randomly selected design problems to evaluate the effectiveness of the proposed approach. In the case of sample 1, 10 design problems are considered. Subsequently, in the case of sample 2, 30 (including 10 design problems of sample 1) are considered for analysis (Appendix B). Moreover, it is assured that each sample should include at least one design pattern from each design pattern class (such as 3 pattern classes in GoF pattern collection). The performance of each unsupervised learner with their best weighting method is evaluated in terms of determining the effectiveness of the proposed approach. For each design problem p of sample P (i.e. $p \in P$), the ARI value for each unsupervised learner is computed. Subsequently, average ARI value for each learner is computed for the first and second sample of design problems. The performance of each unsupervised learner is analyzed with best weighting methods. The results for the first and second sample of design problem are shown in the Figure 6.6 (a-b). The main consequences from the result of Figure 6.6 (a-b) are (Response to RQ-2).

- Performance of C-means remains better as compared to other unsupervised learners in terms of ARI with both samples of selected design problems.
- There is no significant decrease in the average performance of each unsupervised learner when the numbers of design problems are increased in the sample.

- In the comparative evaluation with sample-1 and sample-2 design problems, performance of PAM-Euclidean decreases 11.53%, the highest decrease as compared to other unsupervised learners.

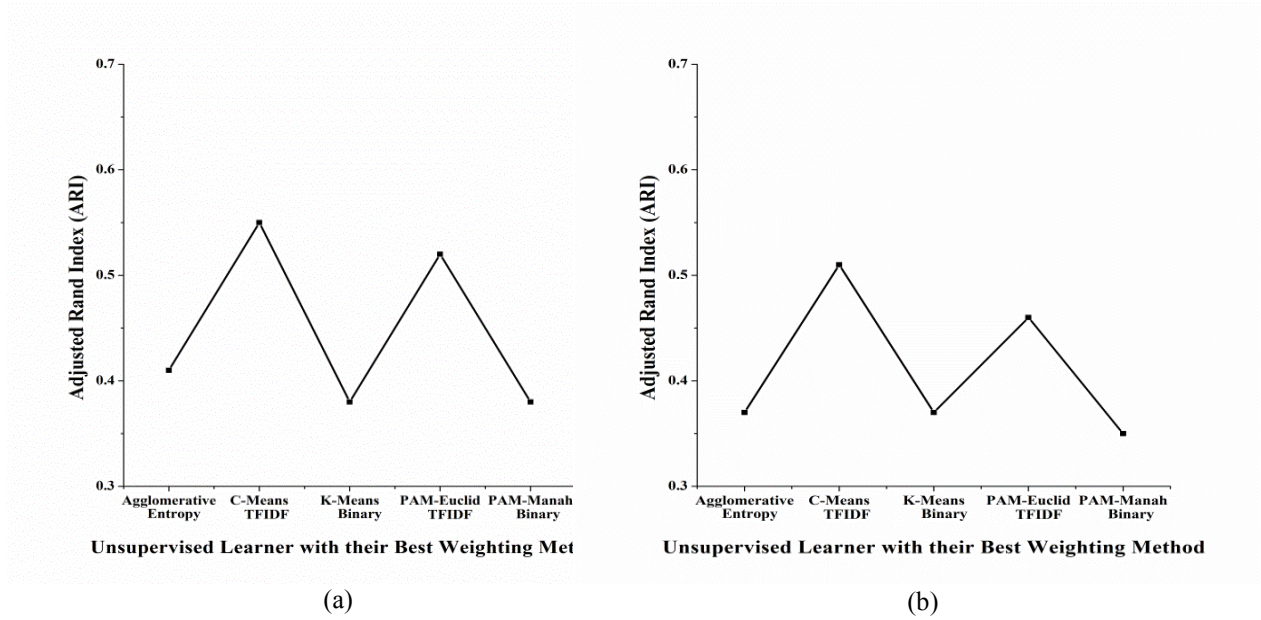


Figure 6.6. Identification of outperform unsupervised learner with best weighting method to determine pattern class for the a) Sample-1 and b) Sample-2 design problems in the context of GoF design pattern collection

Since the Fuzzy c-means is the outperformed unsupervised learner of the proposed approach in the context of the GoF design pattern collection. Consequently, we compute the class membership degree values to describe the closeness between the given design problems and design pattern classes. The highest class membership value of c-means aids us to recommend the right design pattern class for any given design problem of the sample-1 and sample-2. The class member value is not computed for the rest of unsupervised learners. The class membership values for Sample-1 and Sample-2 of GoF design problems with real and predicted design pattern classes are depicted in Table 6.1 and Table 6.2 respectively. For each design problem shown in Table 6.1 and Table 6.2, the highest class membership value aid to suggest the design pattern class. For example, in Table 6.1, for Design Problem 1, the Behavioral design pattern class with highest class membership value (i.e. 0.52) is recommended by Fuzzy c-means. Subsequently, same recommendations are given for rest of design problems in Table 6.1 and Table 6.2 respectively.

Table 6.1. Fuzzy c-means (i.e. Outperform unsupervised learner) recommended class membership values for GoF design problems of Sample 1.

Design Problems	Actual Pattern Class	Predicted Pattern Class	Class Membership Degree Value Recommended By Fuzzy C-means		
			Creational	Structural	Behavioral
Design Problem 1	Behavioral	Behavioral	0.28	0.20	<u>0.52</u>
Design Problem 2	Behavioral	Structural	0.16	<u>0.54</u>	0.30
Design Problem 3	Behavioral	Behavioral	0.10	0.18	<u>0.72</u>
Design Problem 4	Structural	Structural	0.15	<u>0.55</u>	0.30
Design Problem 5	Behavioral	Behavioral	0.14	0.20	<u>0.66</u>
Design Problem 6	Creational	Creational	<u>0.68</u>	0.12	0.20
Design Problem 7	Structural	Structural	<u>0.70</u>	0.14	0.16
Design Problem 8	Structural	Behavioral	0.20	0.14	<u>0.66</u>
Design Problem 9	Creational	Structural	<u>0.63</u>	0.19	0.18
Design Problem 10	Structural	Behavioral	0.09	0.19	<u>0.72</u>

6.2.3. Selection of design pattern for the sample of given design problems

The third objective of the proposed approach is to find the right design pattern for the sample of given design problems using the cosine similarity measure and suggested similarity options (Section 3.1.4). In this regard, the experiment 5 and 6 (Sections 5.3.6 and 5.3.7) are performed to find the closeness between design problems and design patterns of suggested pattern class and best similarity option. For example, in Table 6.1, for the Design Problem 1, Behavioral design pattern class is suggested via the fuzzy c-means. Firstly, for each design problem of target sample (Sample-1 of 10 Design Problems), experiment 5 is performed to determine its closeness with design patterns of suggested pattern class. The results are shown in Appendix C. Secondly; the experiment 6 is performed to select the best similarity option in terms of determining the right design patterns for the target sample of design problems (Response to RQ-3).

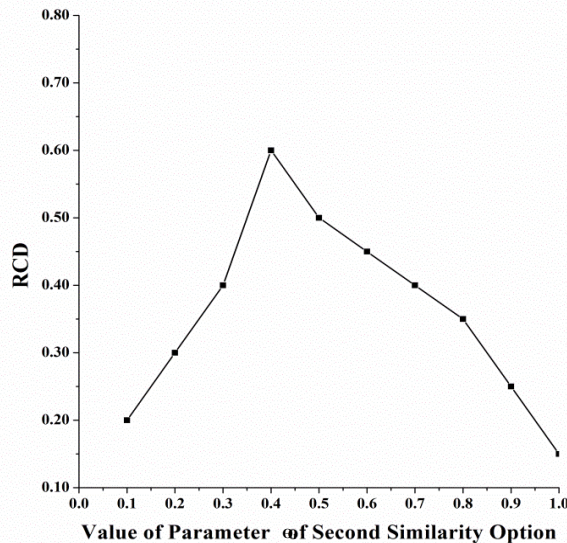
Table 6.2. Fuzzy c-means (i.e. Outperform unsupervised learner) recommended class membership values for GoF design problems of Sample 2.

Design Problems	Actual Pattern Class	Predicted Pattern Class	Class Membership Degree Value Recommended By Fuzzy C-means		
			Creational	Structural	Behavioral
Design Problem 1	Behavioral	Behavioral	0.28	0.20	<u>0.52</u>
Design Problem 2	Behavioral	Structural	0.16	<u>0.54</u>	0.30
Design Problem 3	Behavioral	Behavioral	0.10	0.18	<u>0.72</u>
Design Problem 4	Structural	Structural	0.15	<u>0.55</u>	0.30
Design Problem 5	Behavioral	Behavioral	0.14	0.20	<u>0.66</u>
Design Problem 6	Creational	Creational	<u>0.68</u>	0.12	0.20
Design Problem 7	Structural	Structural	<u>0.70</u>	0.14	0.16
Design Problem 8	Structural	Behavioral	0.20	0.14	<u>0.66</u>
Design Problem 9	Creational	Structural	<u>0.63</u>	0.19	0.18
Design Problem 10	Structural	Behavioral	0.09	0.19	<u>0.72</u>
Design Problem 11	Creational	Structural	0.15	<u>0.65</u>	0.20
Design Problem 12	Creational	Structural	0.05	<u>0.60</u>	0.35
Design Problem 13	Structural	Behavioral	0.10	0.19	<u>0.71</u>
Design Problem 14	Behavioral	Behavioral	0.11	0.30	<u>0.59</u>
Design Problem 15	Behavioral	Behavioral	0.20	0.17	<u>0.63</u>
Design Problem 16	Behavioral	Structural	0.20	<u>0.55</u>	0.25
Design Problem 17	Structural	Structural	0.13	<u>0.62</u>	0.25
Design Problem 18	Structural	Creational	<u>0.64</u>	0.16	0.20
Design Problem 19	Behavioral	Behavioral	0.12	0.30	<u>0.58</u>
Design Problem 20	Behavioral	Structural	0.15	<u>0.70</u>	0.15
Design Problem 21	Behavioral	Behavioral	0.16	0.28	<u>0.56</u>
Design Problem 22	Creational	Creational	<u>0.62</u>	0.18	0.20
Design Problem 23	Structural	Behavioral	0.10	0.17	<u>0.73</u>
Design Problem 24	Behavioral	Behavioral	0.16	0.20	<u>0.64</u>
Design Problem 25	Behavioral	Structural	0.30	<u>0.53</u>	0.17
Design Problem 26	Structural	Structural	0.11	<u>0.58</u>	0.31
Design Problem 27	Behavioral	Creational	<u>0.61</u>	0.19	0.20
Design Problem 28	Behavioral	Behavioral	0.14	0.22	<u>0.64</u>
Design Problem 29	Behavioral	Behavioral	0.20	0.11	<u>0.69</u>
Design Problem 30	Structural	Behavioral	0.25	0.10	<u>0.65</u>

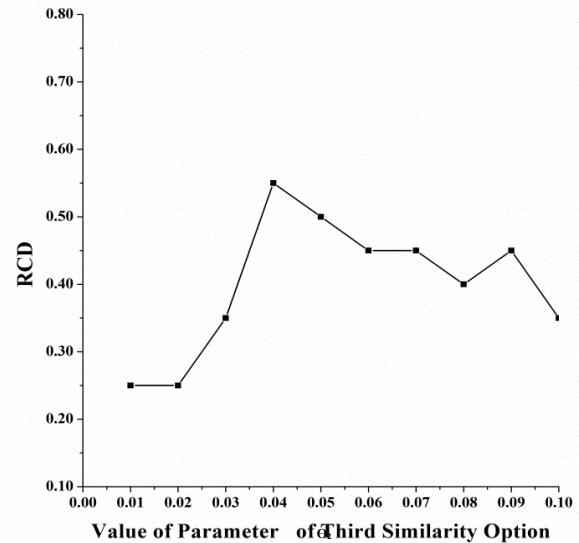
According to first similarity option (Equation 3.6 in Section 3.1.4), from Tables C-1 to C-10, the design pattern with highest Cosine Similarity (CS) value is highlighted and recommended as a right design pattern suggested through the outperform unsupervised learner of the proposed approach. The second (Equation 3.7 in Section 3.1.4) and third (Equation 3.8 in Section 3.1.4) similarity options aid to find the ranges for the parameters Θ_1 and Θ_2 to select the more right design pattern(s). The comparative range values for Θ_1 and Θ_2 parameters for each design problem to get the highest RCD are shown in Table 6.3.

Table 6.3. Computed ranges for Θ_1 and Θ_2 parameters for design problems of sample-1 in the context of the GoF design pattern collection

Design Problem	CS _{max}	Θ_1	Θ_2
Design Problem 1	0.54	0.31 to 0.53	0.0 to 0.22
Design Problem 2	0.38	0.36 to 0.37	0.0 to 0.01
Design Problem 3	0.46	0.43 to 0.45	0.0 to 0.02
Design Problem 4	0.48	0.34 to 0.47	0.0 to 0.13
Design Problem 5	0.58	0.35 to 0.57	0.0 to 0.22
Design Problem 6	0.60	0.53 to 0.59	0.0 to 0.06
Design Problem 7	0.58	0.33 to 0.57	0.0 to 0.24
Design Problem 8	0.60	0.52 to 0.59	0.0 to 0.07
Design Problem 9	0.63	0.48 to 0.62	0.0 to 0.14
Design Problem 10	0.64	0.61 to 0.63	0.0 to 0.02



(a)



(b)

Figure 6.7. Parameters of similarity options for Sample-1 of 10 design problems (Related to GoF pattern collection)

a) Mean RCD for similarity option using equation 3.6 b) Mean RCD for similarity option using equation 3.7

The average RCD for all design problems of Sample-1 in term of Θ_1 and Θ_2 relevant similarity options are depicted in Figure 6.7. As shown in Figure 6.7(a-b), the highest RCD values 0.50 and 0.55 are obtained when Θ_1 and Θ_2 are equal to 0.5 and 0.04.

6.3. Second Case Study (Douglass Design Patterns in the Domain of Real Time Application)

The aim of second case study is to use the proposed approach in the context of software design patterns related to designing issues of real-time applications. The Douglass design patterns are widely used by real-time application development community (Detail is given in the section 4.2.2). Subsequently, these design patterns can be used for the design problems given in the context of operating systems issues, such as memory management and concurrency.

6.3.1. Organization of Design Patterns

Firstly, to determine the best weighting method for each unsupervised learning technique, the experiment 1 is performed according to given guidance in the section 5.3.1. The results are shown in the Figure 6.8.

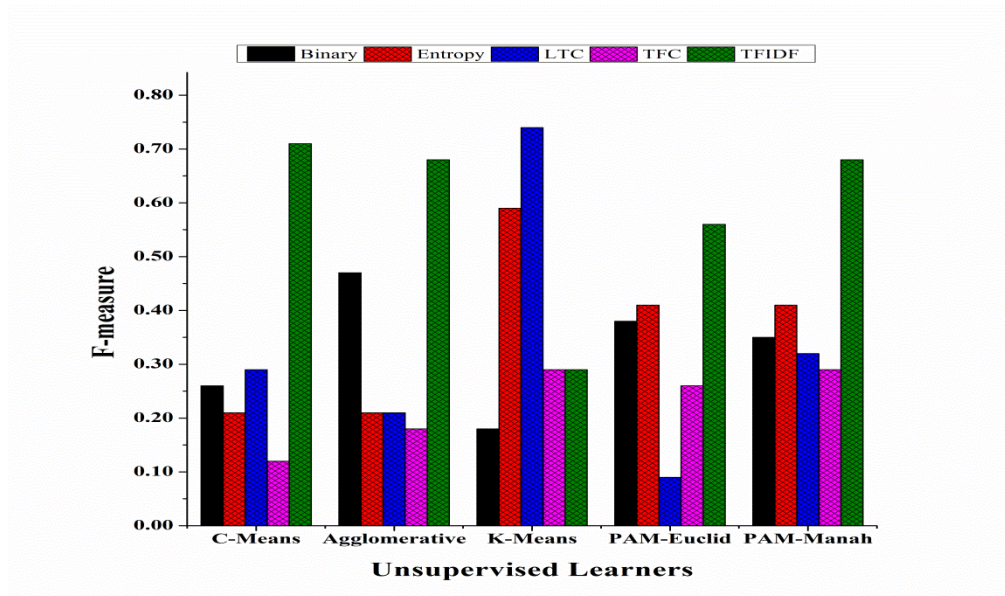


Figure 6.8. Identification of best weighting method for unsupervised learners weighting methods in the context of Douglass design pattern collection

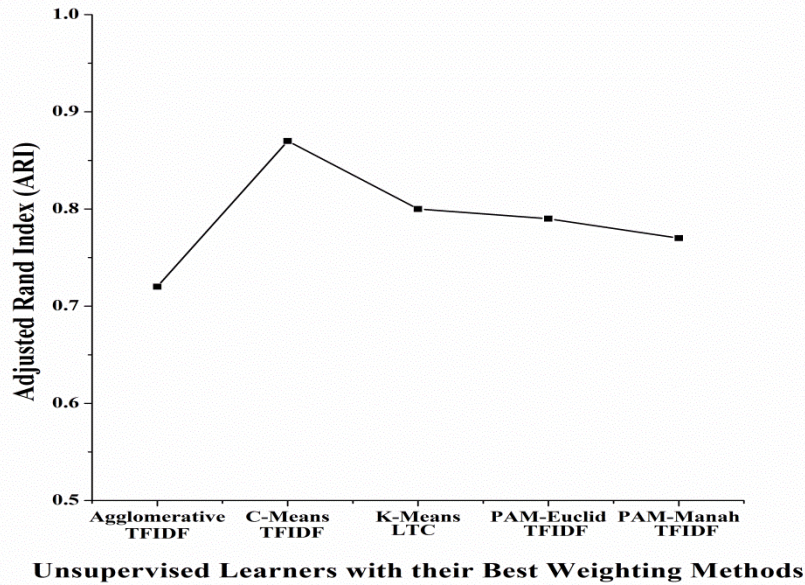


Figure 6.9. Identification of outperform unsupervised learner with best weighting method in the context of Douglass design pattern collection

The weighting method with highest F-measure value of unsupervised learner is recommended as a best weighting method for the corresponding learner. For example, the TFIDF is recommended as a best weighting method with highest F-measure values 0.71, 0.68, 0.56 and 0.68 for Fuzzy C-means, Agglomerative, PAM-Euclidean, PAM-Manhattan respectively. Subsequently, the LTC is recommended as a best weighting method with highest F-measure value 0.74 for K-means (Response to RQ-4).

Secondly, to determine the outperformed unsupervised learner, the experiment 2 is performed according to given guidance in the section 5.3.2. The performance of each unsupervised learner is analyzed with best weighting methods. The results are shown in the Figure 6.9. The result of Figure 6.9 indicates that Fuzzy c-means with highest ARI value (i.e. $ARI=0.87$) is recommended as an outperform unsupervised learner as compared to Agglomerative ($ARI=0.72$), K-means (0.80), PAM-Euclidean ($ARI=0.79$), and PAM-Manhattan ($ARI=0.77$) learners of the proposed approach.

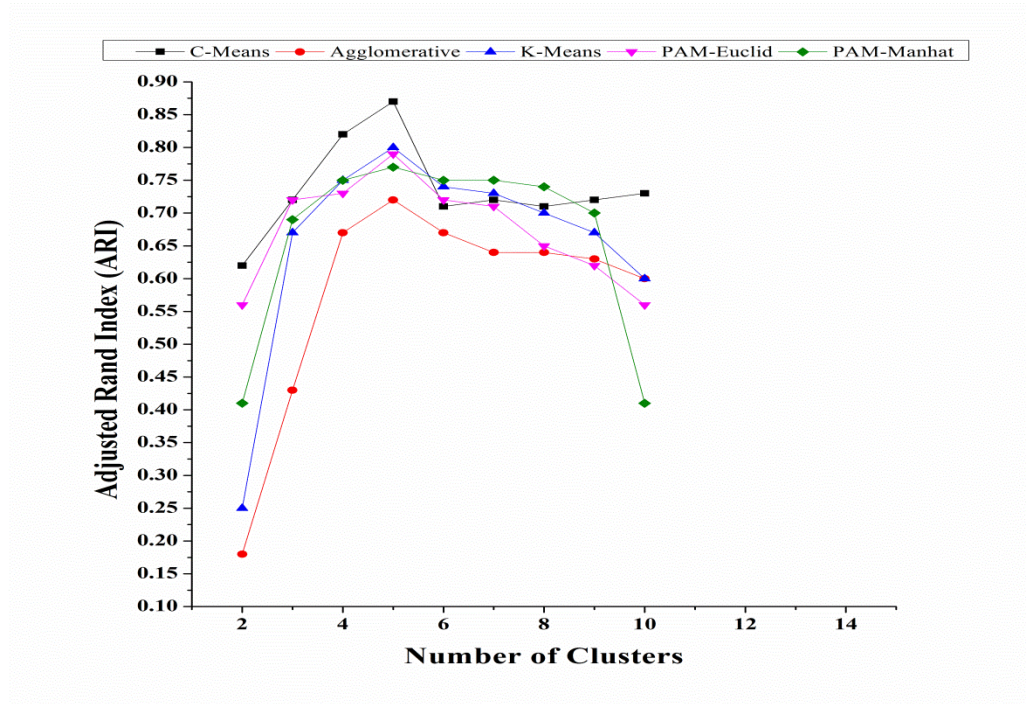


Figure 6.10. The performance of unsupervised learners with best weighting methods and K clusters in the context of Douglass design pattern collection

Third, to determine the effectiveness of the proposed approach in terms of organization of design patterns with respect to the classification scheme of experts, the experiment 3 is performed according to given guidance in the section 5.3.3. The performance of each unsupervised learner is analyzed with best weighting methods. The results are shown in the Figure 6.10. The highest ARI value of each unsupervised learner at K=5 (according to expert opinion) indicates the effectiveness of the proposed approach (Response to RQ-1).

Fourth, in order to investigate the effectiveness of feature selection methods on the performance of unsupervised learners of the proposed approach, the experiment 4 is performed according to given guidance in the section 5.3.4. Therefore, we applied six feature selection methods Chi-Square, Document Frequency (DF), Correlation, Information Gain, Gain Ratio including our proposed technique named Ensemble-IG (to overcome the multi-class problem) and evaluate their impact on the performance of Outperform unsupervised learner Fuzzy c-means. The evaluation results are shown in the Figure 6.11. The x-axis of Figure 6.11 (labeled with Number of Rank Features) refers to top n features rank by each feature selection method. According to F-measure criterion, the feature selection method with the highest increase in the performance of Fuzzy c-means as compared to its actual performance F-measure=0.71 before feature selection (illustrated in the Figure 6.8) is suggested. For example, with top 10 features (depicted on the x-axis of Figure 6.11) rank by each feature selection method, there is an increase from 1.40% to 5.63% (in term of F-measure) in the performance of Fuzzy c-means. The proposed feature selection method Ensemble-IG with the highest increase in the performance (i.e. 5.63%) is suggested for the fuzzy c-means. We have replicated this procedure with Top n (where n ranges from 50 to 350) features and illustrated the evaluation results in the Figure 6.5 (Response to RQ-4 and RQ-5).

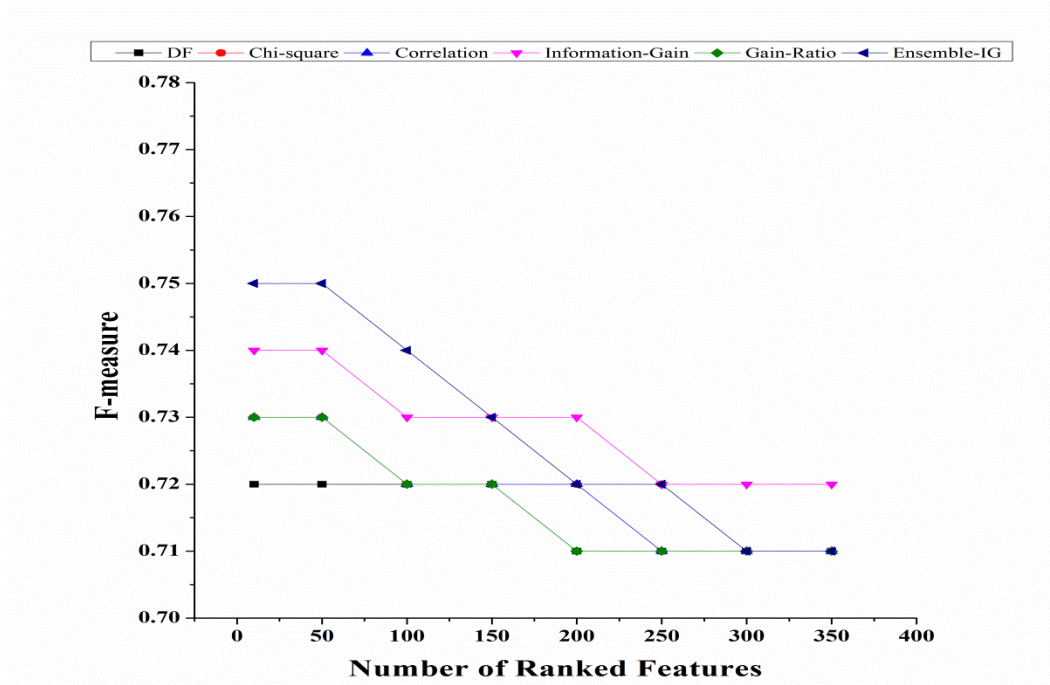


Figure 6.11. The effect of feature selection methods on the performance of outperform unsupervised learner C-means with best weighting method in the context of Douglass design pattern collection

6.3.2. Determination of design pattern class for the sample of given design problems

The second objective of the proposed approach is to find the right design pattern class for the sample of given design problems. In this regard, the experiment 5 is performed. In study 35 design problems are considered. The experiment 5 is performed with two samples (i.e. P) of randomly selected design problems to evaluate the effectiveness of the proposed approach. In the case of sample 1, 10 design problems are considered. Subsequently, in the case of sample 2, 20 (including 10 design problems of sample 1) are considered for analysis (Appendix B). Moreover, it is assured that each sample should include at least one design pattern from each design pattern class (such as 5 pattern classes in the Douglass pattern collection). The performance of each unsupervised learner with their best weighting method is evaluated in terms of determining the effectiveness of the proposed approach. For each design problem p of sample P (i.e. $p \in P$), the ARI value for each unsupervised learner is computed. Subsequently, average ARI value for each learner is computed for the first and second sample of design problems. The performance of each unsupervised learner is analyzed with best weighting methods. The results for the first and second sample of design problem are shown in the Figure 6.12 (a-b). The main consequences from the result of Figure 6.12 (a-b) are.

- Performance of C-means remains better as compared to other unsupervised learners in terms of ARI with both samples of selected design problems.
- There is no significant decrease in the average performance of each unsupervised learner when the numbers of design problems are increased in the sample.

- In the comparative evaluation with sample-1 and sample-2 design problems, performance of PAM-Manhattan decreases 12.16%, the highest decrease as compared to other unsupervised learners.

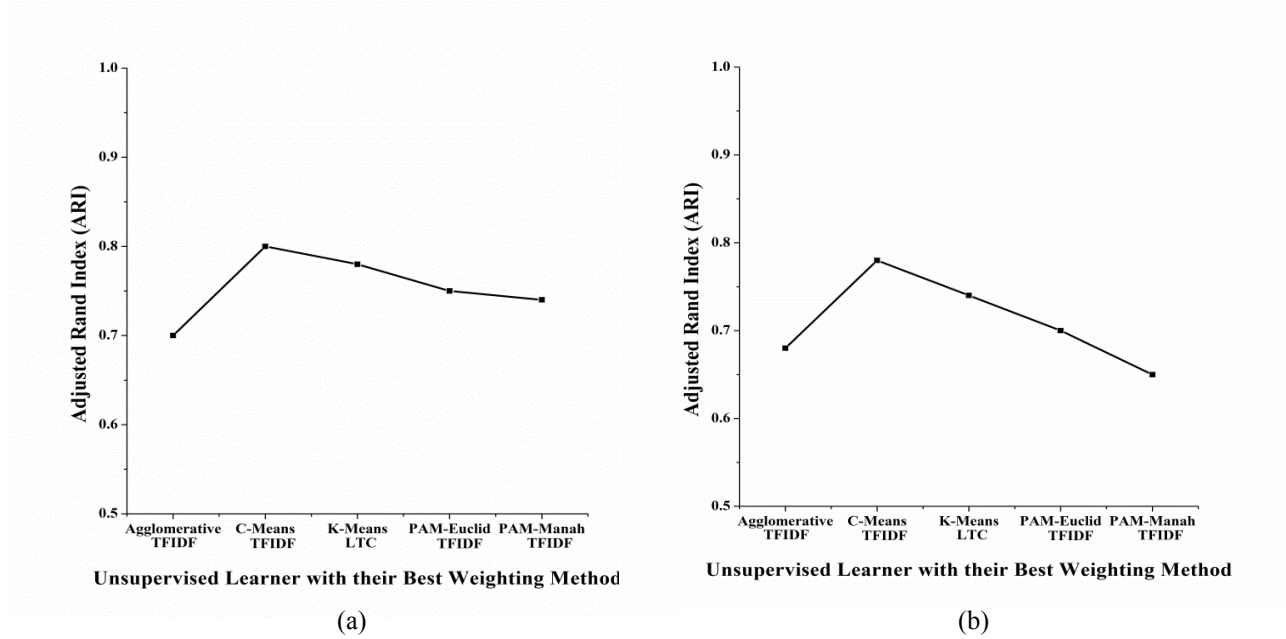


Figure 6.12. Identification of outperform unsupervised learner with best weighting method to determine pattern class for the a) Sample-1 and b) Sample-2 design problems in the context of Douglass design pattern collection

Since the Fuzzy c-means is the outperformed unsupervised learner of the proposed approach in the context of the Douglass design pattern collection. Consequently, we compute the class membership degree values to describe the closeness between the given design problems and design pattern classes. The highest class membership value of c-means aids us to recommend the right design pattern class for any given design problem of the sample-1 and sample-2. The class member value is not computed for the rest of unsupervised learners. The class membership values for Sample-1 and Sample-2 of GoF design problems with real and predicted design pattern classes are depicted in Table 6.4 and Table 6.5 respectively. For each design problem shown in Table 6.4 and Table 6.5, the highest class membership value aid to suggest the design pattern class. For example, in Table 6.4, for Design Problem 1, the Memory design pattern class with highest class membership value (i.e. 0.48) is recommended by Fuzzy c-means. Subsequently, same recommendations are given for rest of design problems in Table 6.4 and Table 6.5 respectively.

Table 6.4. Fuzzy c-means (i.e. Outperform unsupervised learner) recommended class membership values for Douglass design problems of Sample 1.

Design Problems	Actual Pattern Class	Predicted Pattern Class	Class Membership Degree Value Recommended By Fuzzy C-means				
			Memory	Resource	Distribution	Safety and Reliability	Concurrency
Design Problem 1	Memory	Memory	<u>0.48</u>	0.20	0.07	0.15	0.10
Design Problem 2	Distribution	Distribution	0.06	0.04	<u>0.50</u>	0.20	0.20
Design Problem 3	Memory	Resource	0.40	<u>0.45</u>	0.05	0.07	0.08
Design Problem 4	Resource	Resource	0.15	<u>0.40</u>	0.30	0.10	0.05
Design Problem 5	Distribution	Safety and Reliability	0.10	0.20	0.22	<u>0.38</u>	0.10
Design Problem 6	Concurrency	Concurrency	0.10	0.18	0.22	0.04	<u>0.46</u>
Design Problem 7	Concurrency	Concurrency	0.12	0.20	0.10	0.07	<u>0.51</u>
Design Problem 8	Safety and Reliability	Safety and Reliability	0.13	0.17	0.15	<u>0.45</u>	0.10
Design Problem 9	Resource	Resource	0.09	<u>0.49</u>	0.02	0.21	0.19
Design Problem 10	Memory	Distribution	0.30	0.08	<u>0.42</u>	0.08	0.12

6.3.3. Selection of design pattern for the sample of given design problems

The third objective of the proposed approach is to find the right design pattern for the sample of given design problems using the cosine similarity measure and suggested similarity options (Section 3.1.4). In this regard, the experiment 5 and 6 (Sections 5.3.6 and 5.3.7) are performed to find the closeness between design problems and design patterns of suggested pattern class and best similarity option. For example, in Table 6.4, for the Design Problem 1, Memory design pattern class is suggested via the fuzzy c-means. Firstly, for each design problem of target sample (Sample-1 of 10 Design Problems), experiment 5 is performed to find its closeness with design patterns of suggested pattern class. The results are shown in Appendix C. Secondly, the experiment 6 is performed to select the best similarity option in terms of determining the right design patterns for the target sample of design problems (Response to RQ-3).

Table 6.5. Fuzzy c-means (i.e. Outperform unsupervised learner) recommended class membership values for Douglass design problems of Sample 2.

Design Problems	Actual Pattern Class	Predicted Pattern Class	Class Membership Degree Value Recommended By Fuzzy C-means				
			Memory	Resource	Distribution	Safety and Reliability	Concurrency
Design Problem 1	Memory	Memory	<u>0.48</u>	0.20	0.07	0.15	0.10
Design Problem 2	Distribution	Distribution	0.06	0.04	<u>0.50</u>	0.20	0.20
Design Problem 3	Memory	Resource	0.40	<u>0.45</u>	0.05	0.07	0.08
Design Problem 4	Resource	Resource	0.15	<u>0.40</u>	0.30	0.10	0.05
Design Problem 5	Distribution	Safety and Reliability	0.10	0.20	0.22	<u>0.38</u>	0.10
Design Problem 6	Concurrency	Concurrency	0.10	0.18	0.22	0.04	<u>0.46</u>
Design Problem 7	Concurrency	Concurrency	0.12	0.20	0.10	0.07	<u>0.51</u>
Design Problem 8	Safety and Reliability	Safety and Reliability	0.13	0.17	0.15	<u>0.45</u>	0.10
Design Problem 9	Resource	Resource	0.09	<u>0.49</u>	0.02	0.21	0.19
Design Problem 10	Memory	Distribution	0.30	0.08	<u>0.42</u>	0.08	0.12
Design Problem 11	Concurrency	Concurrency	0.07	0.08	0.20	0.13	<u>0.52</u>
Design Problem 12	Memory	Resource	0.25	<u>0.35</u>	0.15	0.15	0.10
Design Problem 13	Memory	Memory	<u>0.48</u>	0.12	0.10	0.12	0.18
Design Problem 14	Memory	Safety and Reliability	0.08	0.07	0.12	<u>0.53</u>	0.20
Design Problem 15	Resource	Resource	0.13	<u>0.47</u>	0.16	0.07	0.17
Design Problem 16	Resource	Resource	0.15	<u>0.55</u>	0.10	0.14	0.04
Design Problem 17	Distribution	Distribution	0.08	0.17	<u>0.45</u>	0.17	0.13
Design Problem 18	Distribution	Distribution	0.10	0.12	<u>0.48</u>	0.09	0.21
Design Problem 19	Distribution	Safety and Reliability	0.12	0.07	0.30	<u>0.33</u>	0.18
Design Problem 20	Distribution	Distribution	0.10	0.05	<u>0.65</u>	0.09	0.11

According to first similarity option (Equation 3.6 in Section 3.1.4), from Table C-11 to C-20, the design pattern with highest Cosine Similarity (CS) value is highlighted and recommended as a right design pattern suggested through the outperform unsupervised learner of the proposed approach. The second (Equation 3.7 in Section 3.1.4) and third (Equation 3.8 in Section 3.1.4) similarity options aid to find the ranges for the parameters Θ_1 and Θ_2 to select the more appropriate design pattern(s). The comparative range values for Θ_1 and Θ_2 parameters for each design problem to get the highest RCD are shown in Table 6.6.

Table 6.6. Computed ranges for Θ_1 and Θ_2 parameters for design problems of sample-1 in the context of the Douglass design pattern collection

Design Problem	$CS_{\max}$	Θ_1	Θ_2
Design Problem 1	0.57	0.54 to 0.56	0.0 to 0.02
Design Problem 2	0.43	0.34 to 0.42	0.0 to 0.08
Design Problem 3	0.64	0.53 to 0.63	0.0 to 0.10
Design Problem 4	0.54	0.40 to 0.53	0.0 to 0.13
Design Problem 5	0.43	0.34 to 0.42	0.0 to 0.08
Design Problem 6	0.48	0.41 to 0.47	0.0 to 0.06
Design Problem 7	0.53	0.31 to 0.52	0.0 to 0.21
Design Problem 8	0.52	0.46 to 0.51	0.0 to 0.05
Design Problem 9	0.45	0.31 to 0.44	0.0 to 0.13
Design Problem 10	0.53	0.42 to 0.52	0.0 to 0.10

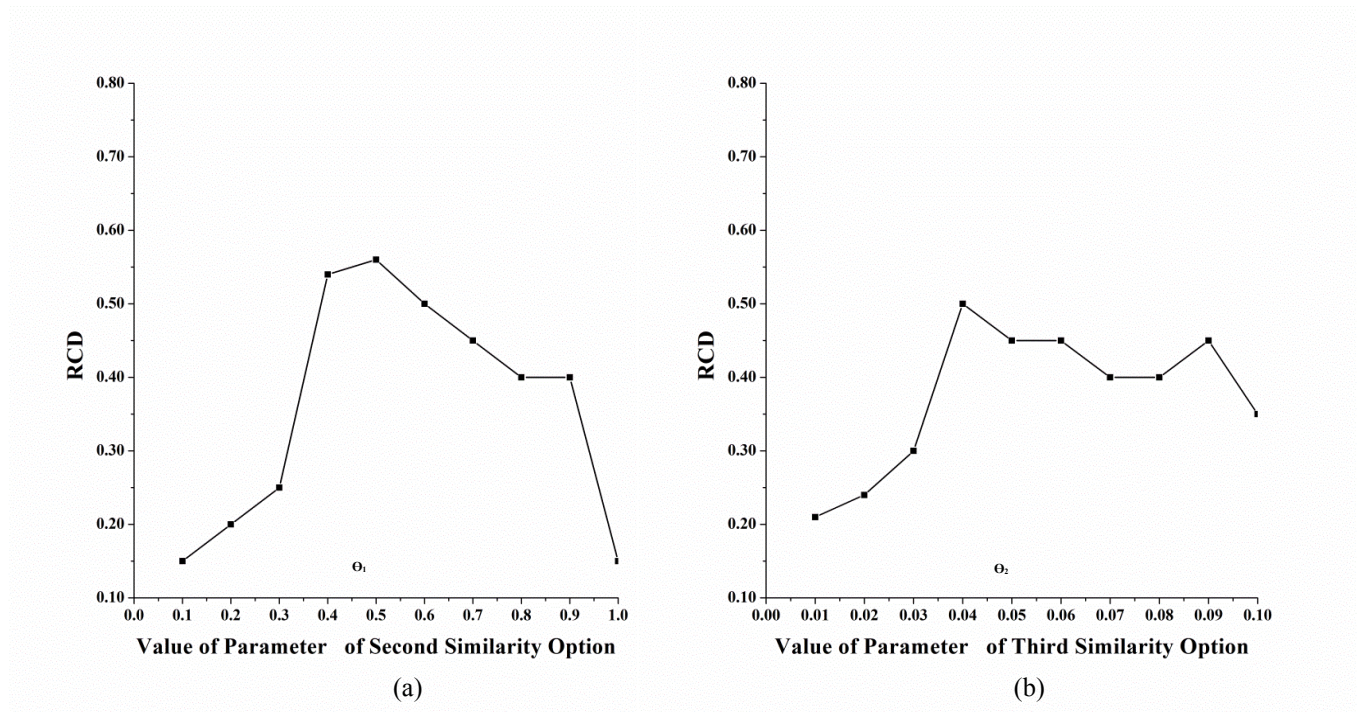


Figure 6.13. Parameters of similarity options for Sample-1 of 10 design problems (Related to Douglass pattern collection)
a) Mean RCD for similarity option using equation 3.6 b) Mean RCD for similarity option using equation 3.7

The average RCD for all design problems of Sample-1 in term of Θ_1 and Θ_2 relevant similarity options are depicted in Figure 6.13. As shown in Figure 6.13(a-b), the highest RCD values 0.56 and 0.50 are obtained when Θ_1 and Θ_2 are equal to 0.5 and 0.05.

6.4. Third Case Study (Schumacher et al Design Patterns in the Domain of Security Systems)

The aim of second case study is to use the proposed approach in the context of software design patterns related to designing issues of security based applications. Subsequently, these design patterns are widely used by the development community (Detail is given in the section 4.2.3) related to security based applications.

6.4.1. Organization of Design Patterns

Firstly, to determine the best weighting method for each unsupervised learning technique, the experiment 1 is performed according to given guidance in the section 5.3.1. The results are shown in the Figure 6.14.

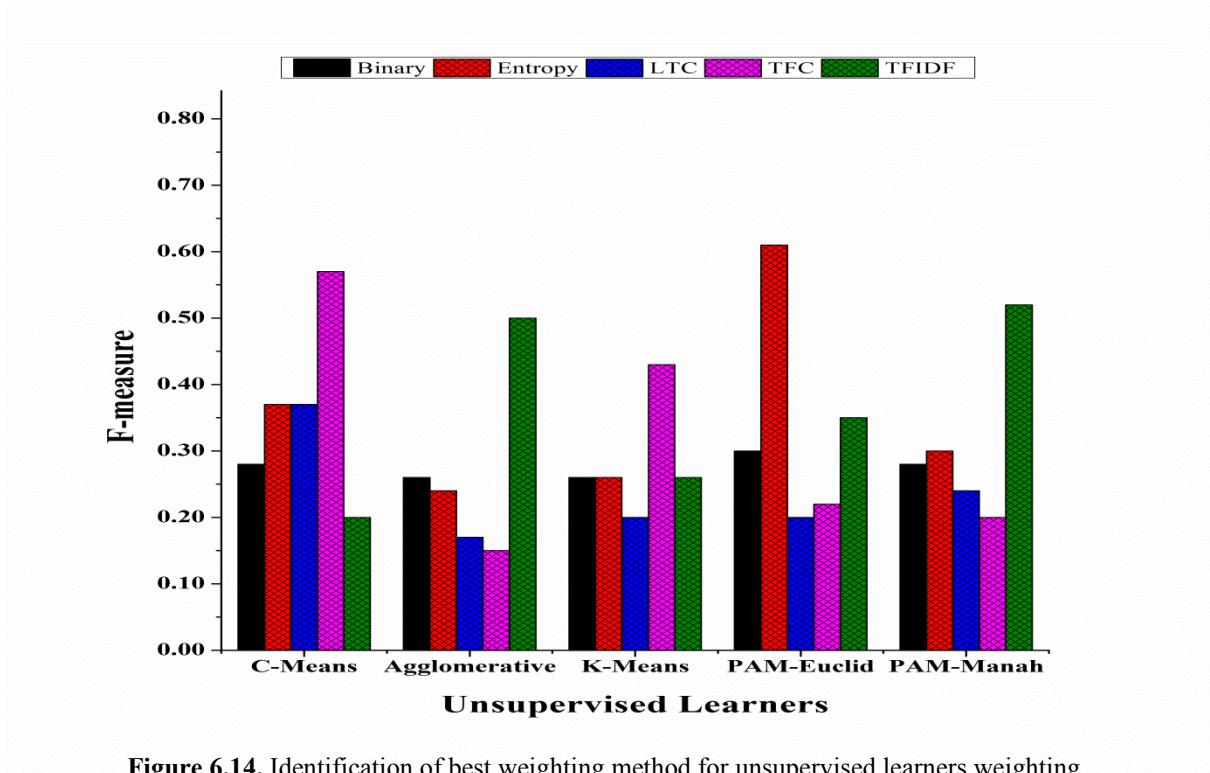


Figure 6.14. Identification of best weighting method for unsupervised learners weighting methods in the context of Security design pattern collection

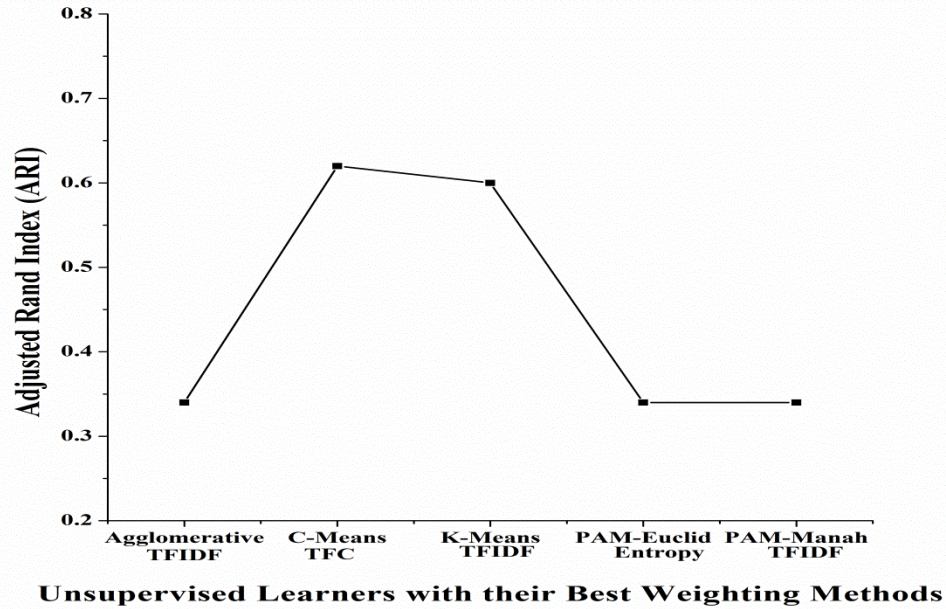


Figure 6.15. Identification of outperform unsupervised learner with best weighting method in the context of Douglass design pattern collection

The weighting method with highest F-measure value of unsupervised learner is recommended as a best weighting method for the corresponding learner. For example, the TFIDF is recommended as a best weighting method with highest F-measure values 0.50 and 0.52 for Agglomerative and PAM-Manhattan respectively. The TFC is recommended as a best weighting method with highest F-measure values 0.57 and 0.43 for C-means and K-means respectively. Subsequently, the Entropy is recommended as a best weighting method with highest F-measure value 0.61 for PAM-Manhattan (Response to RQ-4).

Secondly, to determine the outperformed unsupervised learner, the experiment 2 is performed according to given guidance in the section 5.3.2. The performance of each unsupervised learner is analyzed with best weighting methods. The results are shown in the Figure 6.15. The result of Figure 6.15 indicates that Fuzzy c-means with highest ARI value (i.e. $ARI=0.62$) is recommended as an outperformed unsupervised learner as compared to Agglomerative ($ARI=0.34$), K-means ($ARI=0.60$), PAM-Euclidean ($ARI=0.34$), and PAM-Manhattan ($ARI=0.34$) learners of the proposed approach. We observed the minor difference between the performance of C-means (i.e. $ARI=0.62$) and K-means ($ARI=0.60$).

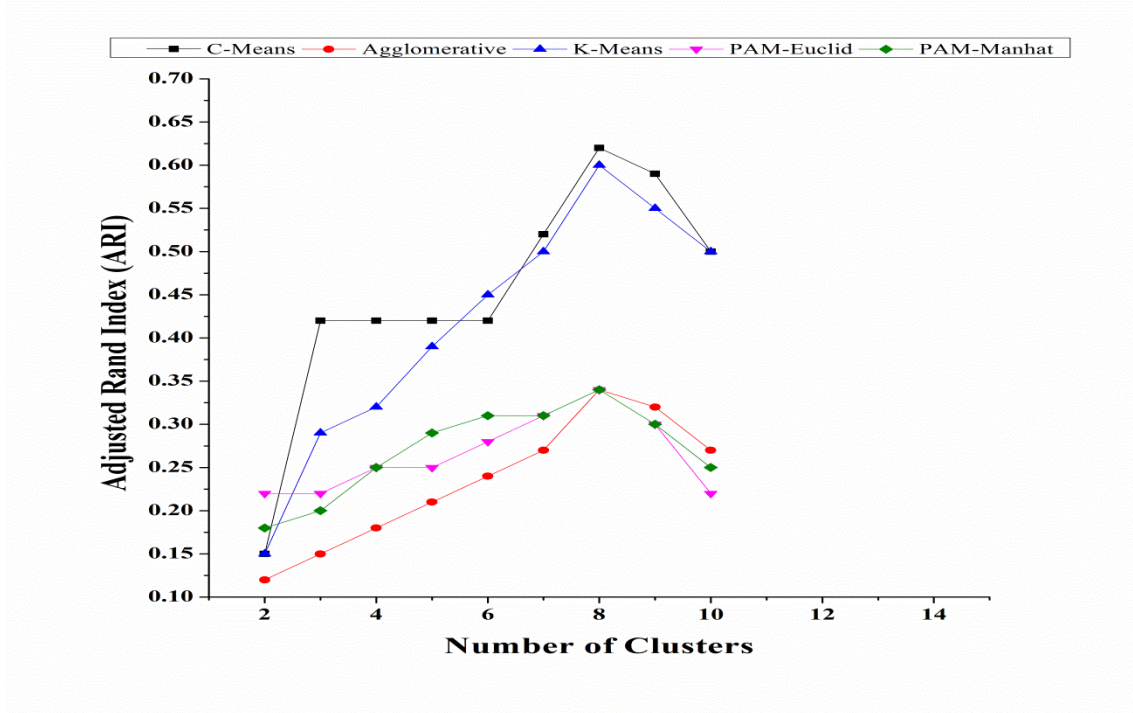


Figure 6.16. The performance of unsupervised learners with best weighting methods and K clusters in the context of Security design pattern collection

Third, to determine the effectiveness of the proposed approach in terms of organization of design patterns with respect to the classification scheme of experts, the experiment 3 is performed according to given guidance in the section 5.3.3. The performance of each unsupervised learner is analyzed with best weighting methods. The results are shown in the Figure 6.16. The highest ARI value of each unsupervised learner at K=8 (according to expert opinion) indicates the effectiveness of the proposed approach (Response to RQ-1).

Fourth, in order to investigate the effectiveness of feature selection methods on the performance of unsupervised learners of the proposed approach, the experiment 4 is performed according to given guidance in the section 5.3.4. Therefore, we applied six feature selection methods Chi-Square, Document Frequency (DF), Correlation, Information Gain, Gain Ratio including our proposed technique named Ensemble-IG (to overcome the multi-class problem) and evaluate their impact on the performance of Outperform unsupervised learner Fuzzy c-means. The evaluation results are shown in the Figure 6.17. The x-axis of Figure 6.17 (labeled with Number of Rank Features) refers to top n features rank by each feature selection method. According to F-measure criterion, the feature selection method with the highest increase in the performance of Fuzzy c-means as compared to its actual performance F-measure=0.71 before feature selection (illustrated in the Figure 6.14) is suggested. For example, with top 10 features (depicted on the x-axis of Figure 6.17) rank by each feature selection method, there is an increase from 5.26% to 14.03% (in term of F-measure) in the performance of Fuzzy c-means. The proposed feature selection method Ensemble-IG with the highest increase in the performance (i.e. 14.03%) is suggested for the fuzzy c-means. We have replicated this procedure with Top n (where n ranges from 50 to 350) features and illustrated the evaluation results in the Figure 6.17 (Response to RQ-4 and RQ-5).

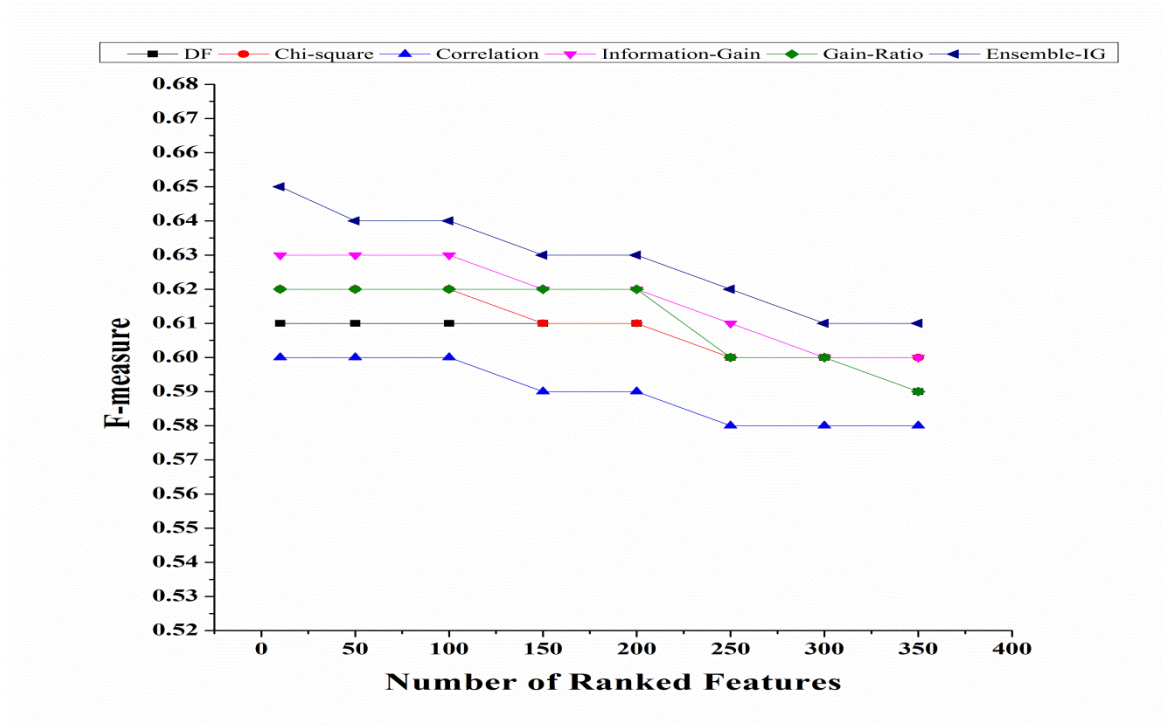


Figure 6.17. The effect of feature selection methods on the performance of outperform unsupervised learner C-means with best weighting method in the context of Security design pattern collection

6.4.2. Determination of design pattern class for the sample of given design problems

The second objective of the proposed approach is to find the right design pattern class for the sample of given design problems. In this regard, the experiment 5 is performed. In study 49 design problems are considered. The experiment 5 is performed with two samples (i.e. P) of randomly selected design problems to evaluate the effectiveness of the proposed approach. In the case of sample 1, 12 design problems are considered. Subsequently, in the case of sample 2, 30 (including 12 design problems of sample 1) are considered for analysis (Appendix B). Moreover, it is assured that each sample should include at least one design pattern from each design pattern class (such as 8 pattern classes in the Security pattern collection). The performance of each unsupervised learner with their best weighting method is evaluated in terms of determining the effectiveness of the proposed approach. For each design problem p of sample P (i.e. $p \in P$), the ARI value for each unsupervised learner is computed. Subsequently, average ARI value for each learner is computed for the first and second sample of design problems. The performance of each unsupervised learner is analyzed with best weighting methods. The results for the first and second sample of design problem are shown in the Figure 6.18 (a-b). The main consequences from the result of Figure 6.18 (a-b) are (Response to RQ-2).

- Performance of C-means remains better as compared to other unsupervised learners in terms of ARI with both samples of selected design problems.

- There is no significant decrease in the average performance of each unsupervised learner when the numbers of design problems are increased in the sample.
- In the comparative evaluation with sample-1 and sample-2 design problems, the performance of Fuzzy C-means decreases 5.71%, the highest decrease as compared to other unsupervised learners.

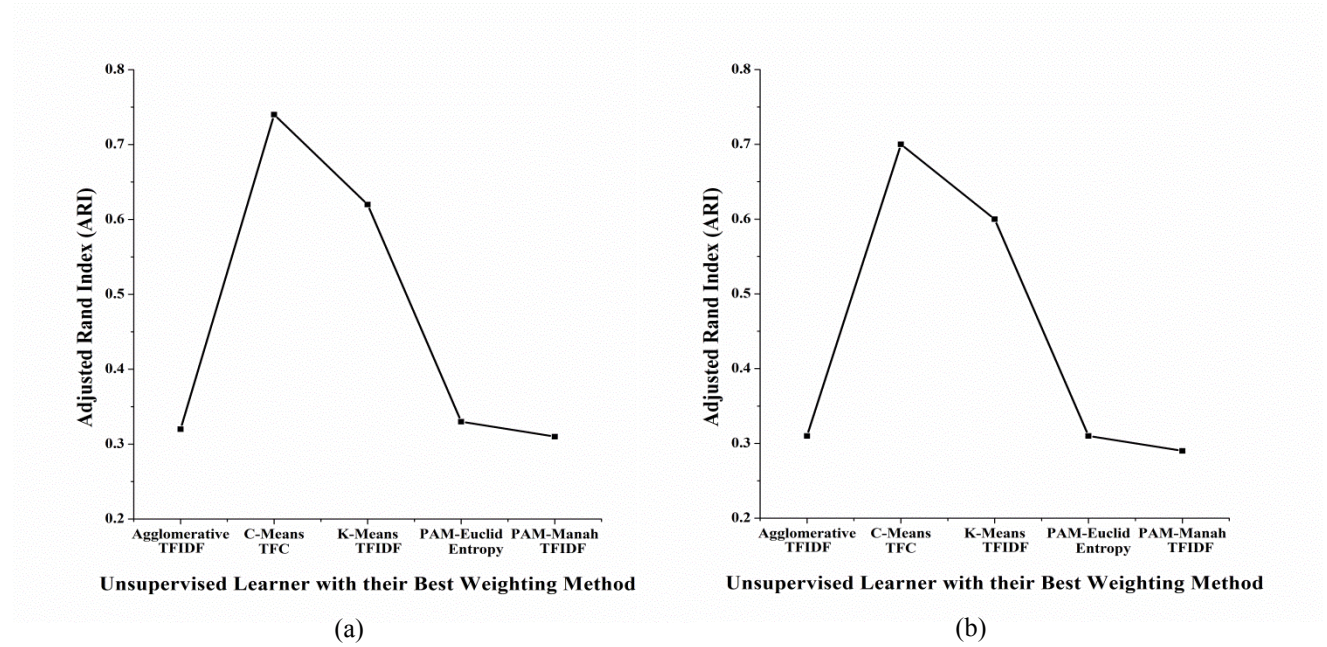


Figure 6.18. Identification of outperform unsupervised learner with best weighting method to determine pattern class for the a) Sample-1 and b) Sample-2 design problems in the context of Security design pattern

Since the Fuzzy c-means is the outperformed unsupervised learner of the proposed approach in the context of the Security design pattern collection. Consequently, we compute the class membership degree values to describe the closeness between the given design problems and design pattern classes. The highest class membership value of c-means aids us to recommend the right design pattern class for any given design problem of the sample-1 and sample-2. The class member value is not computed for the rest of unsupervised learners. The class membership values for Sample-1 and Sample-2 of GoF design problems with real and predicted design pattern classes are depicted in Table 6.7 and Table 6.8 respectively. For each design problem shown in Table 6.7 and Table 6.8, the highest class membership value aid to suggest the design pattern class. For example, in Table 6.7, for Design Problem 1, the Memory design pattern class with highest class membership value (i.e. 0.40) is recommended by Fuzzy c-means. Subsequently, same recommendations are given for rest of design problems in Table 6.7 and Table 6.8 respectively.

Table 6.7. Fuzzy c-means (i.e. Outperform unsupervised learner) recommended class membership values for Security design problems of Sample 1.

Design Problems	Actual Pattern Class	Predicted Pattern Class	Class Membership Degree Value Recommended By Fuzzy C-means							
			Accounting	Access Control Models	Operating System Access Control	Identification and Authentication Patterns	Firewall Architecture	Enterprise Security and Risk Management	Security Internet Application	System Access Control Architecture
Design Problem 1	Accounting	Accounting	<u>0.40</u>	0.10	0.07	0.03	0.12	0.08	0.07	0.13
Design Problem 2	Access Control Models	Accounting	<u>0.37</u>	0.35	0.03	0.05	0.10	0.03	0.03	0.04
Design Problem 3	Operating System Access Control	Operating System Access Control	0.08	0.07	<u>0.43</u>	0.05	0.07	0.12	0.09	0.09
Design Problem 4	Access Control Models	Access Control Models	0.11	<u>0.39</u>	0.10	0.05	0.06	0.07	0.20	0.02
Design Problem 5	Accounting	Firewall Architecture	0.30	0.04	0.07	0.03	<u>0.36</u>	0.05	0.05	0.10
Design Problem 6	Operating System Access Control	Operating System Access Control	0.06	0.07	<u>0.42</u>	0.11	0.04	0.20	0.05	0.05
Design Problem 7	Identification and Authentication Patterns	Firewall Architecture	0.07	0.06	0.11	0.29	<u>0.34</u>	0.03	0.04	0.06
Design Problem 8	Security Internet Application	Security Internet Application	0.05	0.03	0.03	0.04	0.22	0.09	<u>0.45</u>	0.09
Design Problem 9	Security Internet Application	Security Internet Application	0.05	0.06	0.03	0.24	0.11	0.06	<u>0.39</u>	0.06
Design Problem 10	Firewall Architecture	Accounting	<u>0.35</u>	0.10	0.07	0.03	0.25	0.06	0.06	0.08
Design Problem 11	Enterprise Security and Risk Management	Enterprise Security and Risk Management	0.11	0.06	0.03	0.10	0.12	<u>0.38</u>	0.13	0.07
Design Problem 12	System Access Control Architecture	Accounting	<u>0.33</u>	0.07	0.05	0.01	0.17	0.05	0.02	0.30

Table 6.8. Fuzzy c-means (i.e. Outperform unsupervised learner) recommended class membership values for Security design problems of Sample 2.

Design Problems	Actual Pattern Class	Predicted Pattern Class	Class Membership Degree Value Recommended By Fuzzy C-means							
			Accounting	Access Control Models	Operating System Access Control	Identification and Authentication Patterns	Firewall Architecture	Enterprise Security and Risk Management	Security Internet Application	System Access Control Architecture
Design Problem 1	Accounting	Accounting	<u>0.40</u>	0.10	0.07	0.03	0.12	0.08	0.07	0.13
Design Problem 2	Access Control Models	Accounting	<u>0.37</u>	0.35	0.03	0.05	0.10	0.03	0.03	0.04
Design Problem 3	Operating System Access Control	Operating System Access Control	0.08	0.07	<u>0.43</u>	0.05	0.07	0.12	0.09	0.09
Design Problem 4	Access Control Models	Access Control Models	0.11	<u>0.39</u>	0.10	0.05	0.06	0.07	0.20	0.02
Design Problem 5	Accounting	Firewall Architecture	0.30	0.04	0.07	0.03	<u>0.36</u>	0.05	0.05	0.10
Design Problem 6	Operating System Access Control	Operating System Access Control	0.06	0.07	<u>0.42</u>	0.11	0.04	0.20	0.05	0.05
Design Problem 7	Identification and Authentication Patterns	Firewall Architecture	0.07	0.06	0.11	0.29	<u>0.34</u>	0.03	0.04	0.06
Design Problem 8	Security Internet Application	Security Internet Application	0.05	0.03	0.03	0.04	0.22	0.09	<u>0.45</u>	0.09
Design Problem 9	Security Internet Application	Security Internet Application	0.05	0.06	0.03	0.24	0.11	0.06	<u>0.39</u>	0.06
Design Problem 10	Firewall Architecture	Accounting	<u>0.35</u>	0.10	0.07	0.03	0.25	0.06	0.06	0.08
Design Problem 11	Enterprise Security and Risk Management	Enterprise Security and Risk Management	0.11	0.06	0.03	0.10	0.12	<u>0.38</u>	0.13	0.07
Design Problem 12	System Access Control Architecture	Accounting	<u>0.33</u>	0.07	0.05	0.01	0.17	0.05	0.02	0.30
Design Problem 13	System Access Control	System Access Control	0.20	0.06	0.06	0.03	0.15	0.05	0.02	<u>0.43</u>

Design Problem 14	Architecture System Access Control	Architecture Accounting	<u>0.39</u>	0.05	0.31	0.05	0.10	0.04	0.02	0.04
Design Problem 15	Architecture Accounting	Architecture Accounting	<u>0.43</u>	0.05	0.08	0.07	0.09	0.12	0.07	0.09
Design Problem 16	Accounting	System Access Control	0.20	0.04	0.07	0.03	0.10	0.12	0.08	<u>0.36</u>
Design Problem 17	Enterprise Security and Risk	Enterprise Security and Risk	0.08	0.11	0.05	0.05	0.12	<u>0.40</u>	0.07	0.12
Design Problem 18	Management Operating System Access Control	Management Operating System Access Control	0.11	0.06	<u>0.45</u>	0.03	0.14	0.06	0.11	0.04
Design Problem 19	Operating System Access Control	Operating System Access Control	0.04	0.04	<u>0.40</u>	0.13	0.13	0.12	0.08	0.06
Design Problem 20	Operating System Access Control	Accounting	<u>0.41</u>	0.04	0.29	0.06	0.08	0.05	0.03	0.04
Design Problem 21	Security Internet Application System	Security Internet Application System	0.15	0.07	0.05	0.05	0.18	0.05	<u>0.41</u>	0.04
Design Problem 22	Access Control	Access Control	0.12	0.07	0.05	0.08	0.20	0.05	0.05	<u>0.38</u>
Design Problem 23	Architecture Firewall Architecture	Architecture Firewall Architecture	0.05	0.08	0.07	0.23	<u>0.40</u>	0.07	0.05	0.05
Design Problem 24	Security Internet Application System	Accounting	<u>0.35</u>	0.05	0.08	0.01	0.17	0.05	0.28	0.02
Design Problem 25	Access Control	System Access Control	0.10	0.06	0.06	0.10	0.16	0.05	0.05	<u>0.42</u>
Design Problem 26	Architecture Enterprise Security and Risk	Architecture Firewall Architecture	0.09	0.05	0.09	0.03	<u>0.37</u>	0.26	0.04	0.07
Design Problem 27	Management Identification and Authenticatio n Patterns	Identification and Authenticatio n Patterns	0.08	0.04	0.08	<u>0.45</u>	0.18	0.07	0.03	0.07
Design Problem 28	Security Internet Application	Firewall Architecture	0.08	0.07	0.07	0.03	<u>0.41</u>	0.08	0.04	0.22
Design Problem 29	Enterprise Security and	Enterprise Security and	0.08	0.13	0.06	0.04	0.12	<u>0.42</u>	0.09	0.10

	Risk Management Operating System Access Control	Risk Management System Access Control Architecture								
Design Problem 30			0.07	0.06	<u>0.36</u>	0.05	0.12	0.06	0.10	0.18

6.4.3. Selection of design pattern for the sample of given design problems

The third objective of the proposed approach is to find the right design pattern for the sample of given design problems using the cosine similarity measure and suggested similarity options (Section 3.1.4). In this regard, the experiment 5 and 6 (Sections 5.3.6 and 5.3.7) are performed to find the closeness between design problems and design patterns of suggested pattern class and best similarity option. For example, in Table 6.5, for the Design Problem 1, Accounting design pattern class is suggested via the fuzzy c-means. Firstly, for each design problem of target sample (Sample-1 of 12 Design Problems), experiment 5 is performed to determine its closeness with design patterns of suggested pattern class. The results are shown in Appendix C. Secondly; the experiment 6 is performed to select the best similarity option in terms of determining the appropriate design patterns for the target sample of design problems (Response to RQ-3).

According to first similarity option (Equation 3.6 in Section 3.1.4), from Tables C-21 to C-32, the design pattern with highest Cosine Similarity (CS) value is highlighted and recommended as a right design pattern suggested through the outperform unsupervised learner of the proposed approach. The second (Equation 3.7 in Section 3.1.4) and third (Equation 3.8 in Section 3.1.4) similarity options aid to find the ranges for the parameters Θ_1 and Θ_2 to select the more right design pattern(s). The comparative range values for Θ_1 and Θ_2 parameters for each design problem to obtain the highest RCD are shown in Table 6.9.

The average RCD for all design problems of Sample-1 in term of Θ_1 and Θ_2 relevant similarity options are depicted in Figure 6.19. As shown in Figure 6.19 (a-b), the highest RCD values 0.40 and 0.45 are obtained when Θ_1 and Θ_2 are equal to 0.5 and 0.05.

6.5. Fourth Case Study (Duyne Design Patterns in the Human Computer Interaction)

The aim of the fourth case study is to use the proposed approach in the context of software design patterns related to graphic designing issues of web-based applications. Subsequently, these design patterns are widely used by the development community (Detail is given in the section 4.2.4) related to the web based applications.

Table 6.9. Computed ranges for Θ_1 and Θ_2 parameters for design problems of sample-1 in the context of the Security design pattern collection

Design Problems	$CS_{\max}$	Θ_1	Θ_2
Design Problem 1	0.51	0.35 to 0.50	0.0 to 0.15
Design Problem 2	0.35	0.32 to 0.34	0.0 to 0.02
Design Problem 3	0.45	0.34 to 0.44	0.0 to 0.10
Design Problem 4	0.52	0.35 to 0.51	0.0 to 0.16
Design Problem 5	0.45	0.39 to 0.44	0.0 to 0.06
Design Problem 6	0.55	0.34 to 0.54	0.0 to 0.20
Design Problem 7	0.35	0.31 to 0.34	0.0 to 0.03
Design Problem 8	0.47	0.32 to 0.46	0.0 to 0.14
Design Problem 9	0.49	0.36 to 0.48	0.0 to 0.12
Design Problem 10	0.37	0.33 to 0.36	0.0 to 0.03
Design Problem 11	0.58	0.31 to 0.57	0.0 to 0.26
Design Problem 12	0.47	0.43 to 0.46	0.0 to 0.03

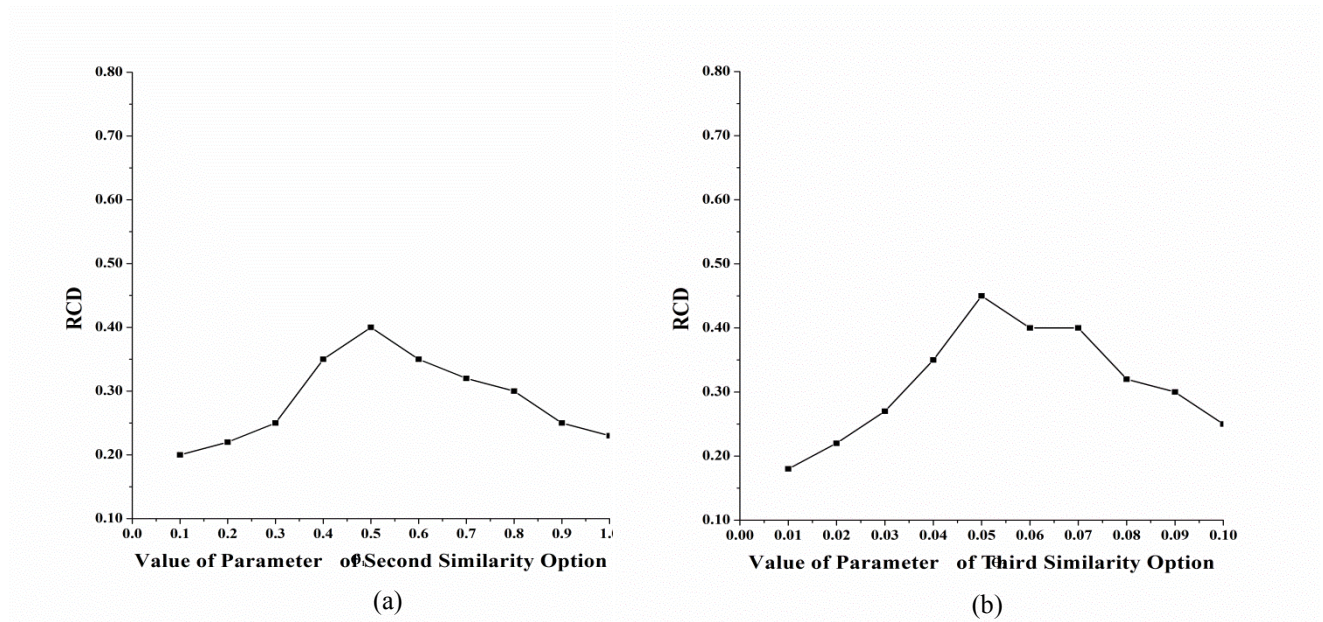


Figure 6.19. Parameters of similarity options for Sample-1 of 12 design problems (Related to Security pattern collection)

a) Mean RCD for similarity option using equation 3.6 b) Mean RCD for similarity option using equation 3.7

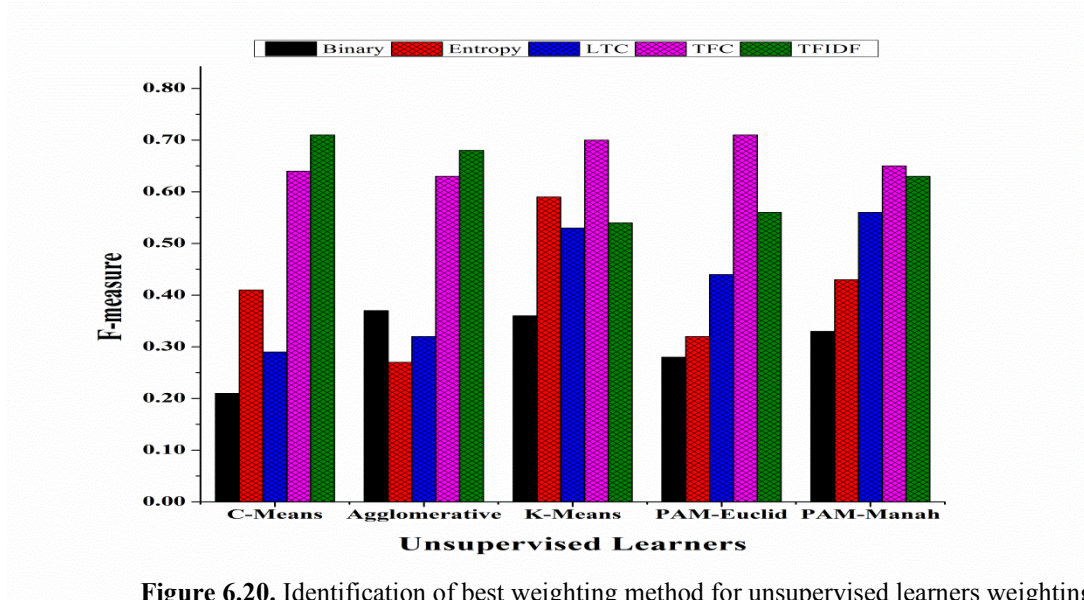


Figure 6.20. Identification of best weighting method for unsupervised learners weighting methods in the context of Duyne design pattern collection

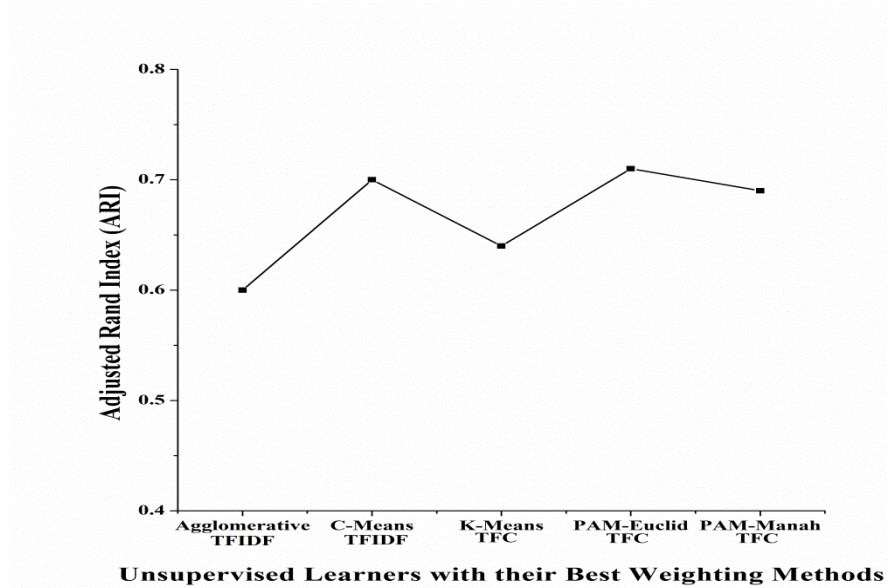


Figure 6.21. Identification of outperform unsupervised learner with best weighting method in the context of Duyne design pattern collection

6.5.1. Organization of Design Patterns

Firstly, to determine the best weighting method for each unsupervised learning technique, the experiment 1 is performed according to given guidance in the section 5.3.1. The results are shown in the Figure 6.20. The weighting method with highest F-measure value of unsupervised learner is recommended as a best weighting method for the corresponding learner. For example, the TFIDF is recommended as a best weighting method with highest F-measure values 0.71 and 0.68 for Agglomerative and C-means respectively. The TFC is recommended as a best weighting

method with highest F-measure values 0.70, 0.70, and 0.65 for K-means, PAM-Euclidean, and PAM-Manhattan and respectively (Response to RQ-4).

Secondly, to determine the outperformed unsupervised learner, the experiment 2 is performed according to given guidance in the section 5.3.2. The performance of each unsupervised learner is analyzed with best weighting methods. The results are shown in the Figure 6.21. The result of Figure 6.21 indicates that PAM-Euclidean with highest ARI value (i.e. $ARI=0.71$) is recommended as an outperformed unsupervised learner as compared to Agglomerative ($ARI=0.60$), K-means ($ARI=0.64$), C-means ($ARI=0.70$), and PAM-Manhattan ($ARI=0.69$) learners of the proposed approach. We observed PAM-Euclidean (i.e. $ARI=0.71$) as an outperformed unsupervised learner with minor difference to the performance of C-means (i.e. $ARI=0.70$) and PAM-Manhattan (i.e. $ARI=0.69$).

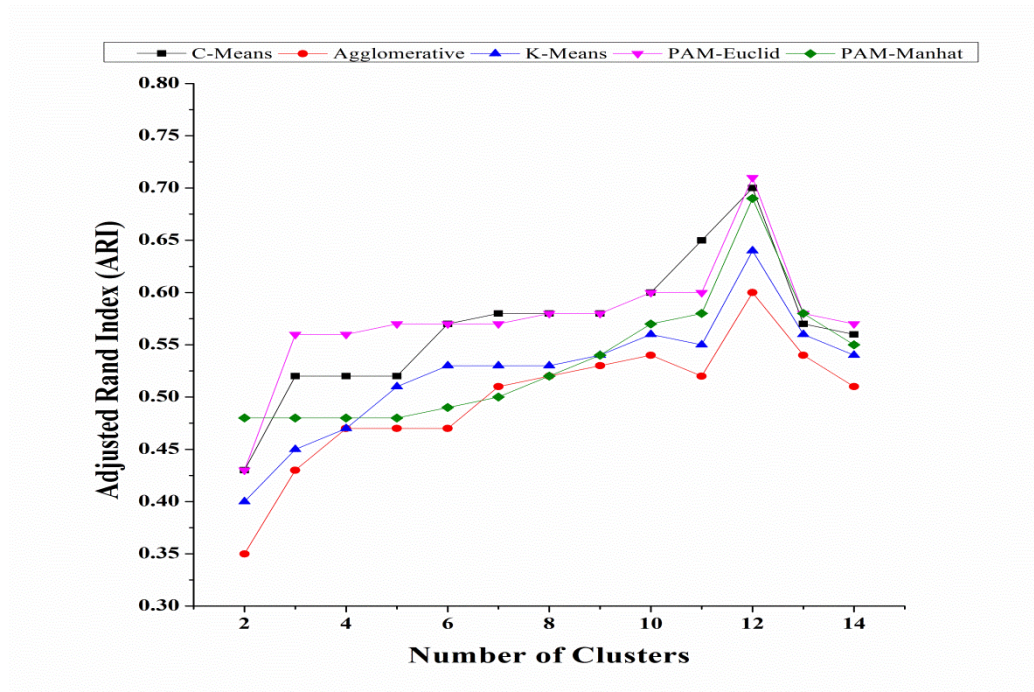


Figure 6.22. The performance of unsupervised learners with best weighting methods and K clusters in the context of Duyne design pattern collection

Third, to determine the effectiveness of the proposed approach in terms of organization of design patterns with respect to the classification scheme of experts, the experiment 3 is performed according to given guidance in the section 5.3.4. The performance of each unsupervised learner is analyzed with best weighting methods. The results are shown in the Figure 6.16. The highest ARI value of each unsupervised learner at $K=12$ (according to expert opinion) indicates the effectiveness of the proposed approach (Response to RQ-1).

Fourth, to investigate the effectiveness of feature selection methods on the performance of unsupervised learners of the proposed approach, the experiment 4 is performed according to given guidance in the section 5.3.4. Therefore, we applied six feature selection methods Chi-Square, Document Frequency (DF), Correlation, Information Gain, Gain Ratio including our proposed technique named Ensemble-IG (to overcome the multi-class problem) and evaluate their impact on the performance of Outperform unsupervised learner PAM-Euclidean. The evaluation

results are shown in the Figure 6.23. The x-axis of Figure 6.23 (labeled with Number of Rank Features) refers to top n features rank by each feature selection method. According to F-measure criterion, the feature selection method with the highest increase in the performance of PAM-Euclidean as compared to its actual performance F-measure=0.70 before feature selection (illustrated in the Figure 6.20) is suggested. For example, with top 10 features (depicted on the x-axis of Figure 6.23) rank by each feature selection method, there is an increase from 7.89% to 11.39% (in term of F-measure) in the performance of PAM-Euclidean. The proposed feature selection method Ensemble-IG with the highest increase in the performance (i.e. 11.39%) is suggested for the PAM-Euclidean. We have replicated this procedure with Top n (where n ranges from 50 to 350) features and illustrated the evaluation results in the Figure 6.23 (Response to RQ-4 and RQ-5).

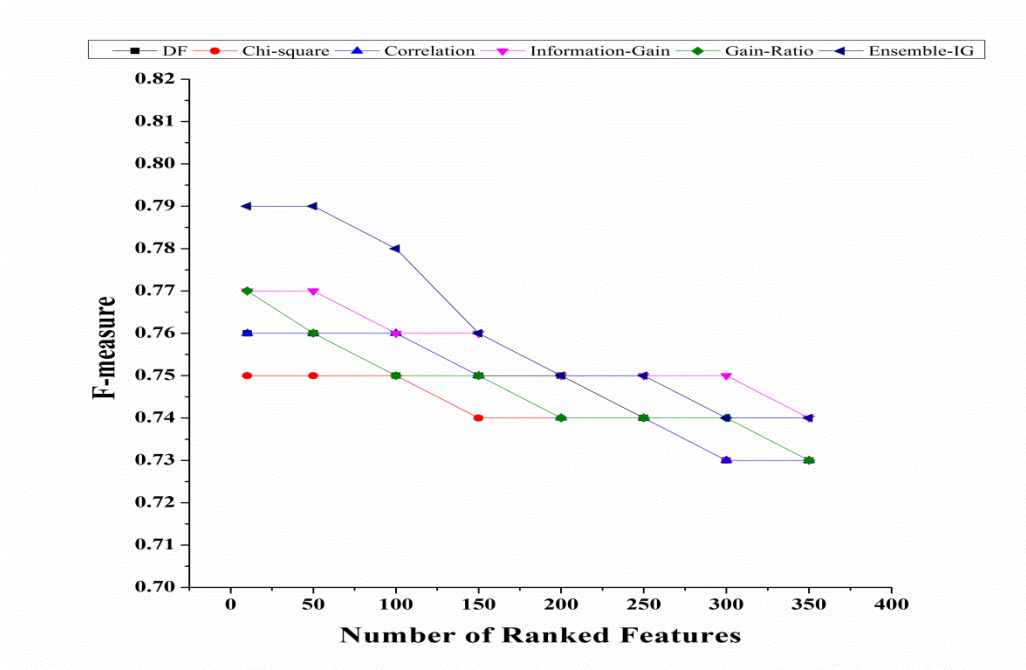


Figure 6.23. The effect of feature selection methods on the performance of outperform unsupervised learner PAM-Euclidean with best weighting method in the context of Duyne design pattern collection

6.5.2. Determination of design pattern class for the sample of given design problems

The second objective of the proposed approach is to find the right design pattern class for the sample of given design problems. In this regard, the experiment 5 is performed. In study 40 design problems are considered. The experiment 5 is performed with two samples (i.e. P) of randomly selected design problems to evaluate the effectiveness of the proposed approach. In the case of sample 1, 15 design problems are considered. Subsequently, in the case of sample 2, 25 (including 15 design problems of sample 1) are considered for analysis (Appendix B). Moreover, it is assured that each sample should include at least one design pattern from each design pattern class (such as 12 pattern classes in Duyne pattern collection). The performance of each unsupervised learner with their best weighting method is evaluated in terms of determining the effectiveness of the proposed approach. For each design problem p of sample P (i.e. $p \in P$), the ARI value for each unsupervised learner is computed. Subsequently,

average ARI value for each learner is computed for the first and second sample of design problems. The performance of each unsupervised learner is analyzed with best weighting methods. The results for the first and second sample of design problem are shown in the Figure 6.24 (a-b). The main consequences from the result of Figure 6.24 (a-b) are (Response to RQ-2).

- Performance of PAM-Euclidean remains better as compared to other unsupervised learners in terms of ARI with both samples of selected design problems.
- There is no significant decrease in the average performance of each unsupervised learner when the numbers of design problems are increased in the sample.
- In the comparative evaluation with sample-1 and sample-2 design problems, performance of Agglomerative and PAM-Manhattan means decreases 12% and 10.16% respectively, this is the highest decrease as compared to other unsupervised learners.

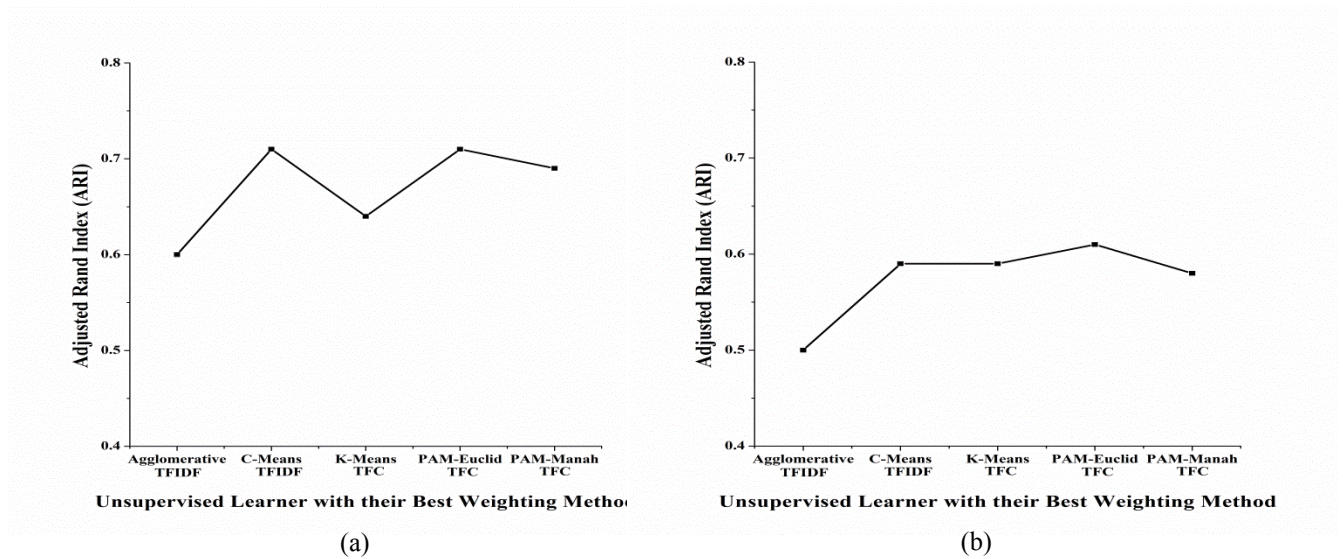


Figure 6.24. Identification of outperform unsupervised learner with best weighting method to determine pattern class for the a) Sample-1 and b) Sample-2 design problems in the context of Duyne design pattern collection

Since the PAM-Euclidean rather than C-means is the outperformed unsupervised learner of the proposed approach in the context of the Duyne design pattern collection. Consequently, we do not compute the class membership degree values to describe the closeness between the given design problems and design pattern classes.

6.5.3. Selection of design pattern for the sample of given design problems

The third objective of the proposed approach is to find the right design pattern for the sample of given design problems using the cosine similarity measure and suggested similarity options (Section 3.1.4). In this regard, the experiment 5 and 6 (Sections 5.3.6 and 5.3.7) are performed to find the closeness between design problems and design patterns of suggested pattern class and best similarity option respectively. Firstly, for each design problem of

target sample (Sample-1 of 15 Design Problems), experiment 5 is performed to find its closeness with design patterns of suggested pattern class. The results are shown in Appendix C. Secondly, the experiment 6 is performed to select the best similarity option in terms of determining the right design patterns for the target sample of design problems (Response to RQ-3).

According to first similarity option (Equation 3.6 in Section 3.1.4), from Tables C-33 to C-47, the design pattern with highest Cosine Similarity (CS) value is highlighted and recommended as a right design pattern suggested through the outperform unsupervised learner of the proposed approach. The second (Equation 3.7 in Section 3.1.4) and third (Equation 3.8 in Section 3.1.4) similarity options aid to find the ranges for the parameters Θ_1 and Θ_2 to select the more right design pattern(s). The comparative range values for Θ_1 and Θ_2 parameters for each design problem to obtain the highest RCD are shown in Table 6.10.

Table 6.10. Computed ranges for Θ_1 and Θ_2 parameters for design problems of sample-1 in the context of the Duynne design pattern collection

Design Problems	$CS_{\max}$	Θ_1	Θ_2
Design Problem 1	0.52	0.37 to 0.51	0.0 to 0.14
Design Problem 2	0.55	0.45 to 0.54	0.0 to 0.09
Design Problem 3	0.43	0.40 to 0.42	0.0 to 0.02
Design Problem 4	0.49	0.35 to 0.48	0.0 to 0.13
Design Problem 5	0.52	0.35 to 0.51	0.0 to 0.16
Design Problem 6	0.62	0.42 to 0.61	0.0 to 0.19
Design Problem 7	0.50	0.43 to 0.49	0.0 to 0.06
Design Problem 8	0.45	0.39 to 0.44	0.0 to 0.05
Design Problem 9	0.43	0.35 to 0.42	0.0 to 0.07
Design Problem 10	0.48	0.34 to 0.47	0.0 to 0.13
Design Problem 11	0.51	0.42 to 0.50	0.0 to 0.08
Design Problem 12	0.62	0.47 to 0.61	0.0 to 0.14
Design Problem 13	0.52	0.48 to 0.51	0.0 to 0.03
Design Problem 14	0.58	0.49 to 0.57	0.0 to 0.08
Design Problem 15	0.57	0.48 to 0.56	0.0 to 0.08

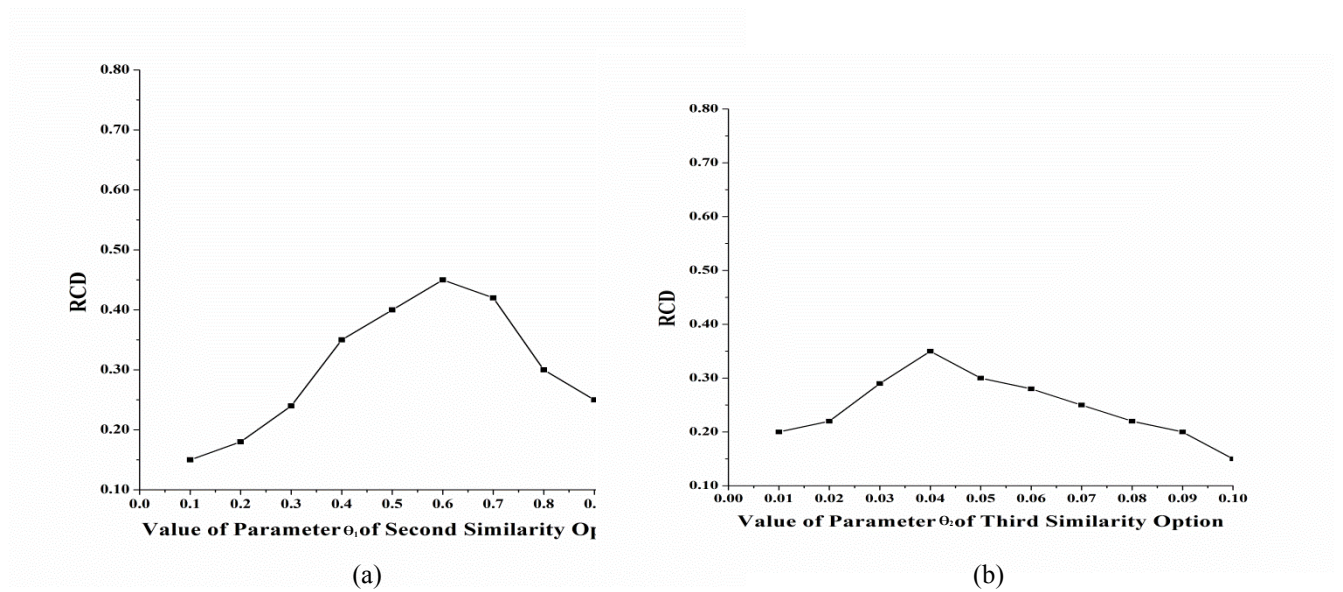


Figure 6.25. Parameters of similarity options for Sample-1 of 15 design problems (Related to Duyne pattern collection)

a) Mean RCD for similarity option using equation 3.6 b) Mean RCD for similarity option using equation 3.7

The average RCD for all design problems of Sample-1 in term of θ_1 and θ_2 relevant similarity options are depicted in Figure 6.25. As shown in Figure 6.25 (a-b), the highest RCD values 0.45 and 0.35 are obtained when θ_1 and θ_2 are equal to 0.6 and 0.04.

7.1. Implications of Deep Learning in different domains

Numerous deep learning algorithms have been introduced and adopted by the research community to do certain tasks in the domain of software engineering. The number of hidden layers and characteristics of datasets are the two potential issues in the training of a conventional deep learning algorithm. The researchers have started to use distributed processing to build more extensive deep learning networks which can enable the research community to manage the large datasets [159]. We have summarized the work domain and application of deep learning algorithms in Table 7.1, which present its significance. We recommend [160] a study for those readers who have quested to know thoroughly about the application of the deep learning approach.

Table 7.1. Implication of deep learning in different domains

Study	Domain	Study Objective
Wang et al. [161]	Defect Prediction	Authors leveraged the deep learning algorithm (i.e. Deep Belief Network (DBN)) to learn semantic features from the token vectors which are extracted from programs' Abstract Syntax Tree in order to build accurate prediction models.
Yang et al. [160]	Defect Prediction	Authors leverage the deep learning to produce the features of the existing set of features and used them to predict buggy commit.
Lam et al. [162]	Fault Localization	In this work, the authors combined the information retrieval techniques and deep learning algorithms in order to enhance the fault localization.
White et al. [163]	Code Suggestion	In this work, the authors used deep learning to model the programming languages to suggest the code.
Pascanu et al. [164]	Malware Classification	In this work, the authors present an approach similar to natural language modelling, which learn the language of malware spoken through the executed instructions and extract the features.

Recently, researchers have adopted the use of deep learning algorithms to improve certain research tasks such as to improve fault localization [162], model the programming languages to suggest the code [163], to retrieve program's structural information, and produce features from program source code for the prediction of defects [161, 164]. Subsequently, the applicability of unsupervised feature learning and deep learning in certain areas have been reported [165]. In this paper, we look the capacity of deep learning [166] and propose an approach by leveraging a powerful learning algorithm Deep Belief Network [167], to learn features from feature vectors extracted from the problem domain of design pattern collection [167, 168]. The aim of the proposed approach is the fill the gap between the semantic relationship between documents(or Patterns) and the features which are used for the organization of design patterns. The Deep Belief Network (DBN) is a generative graphical model to learn a representation which can aid to reconstruct the training data with a high probability. Before applying DBN to learn features from patterns catalog, we performed some pre-processing tasks such as remove stop words and words stemming to preserve the features (with their frequencies) in the feature vector form [14]. Subsequently, these

feature vectors are used as input to the DBN algorithm. We performed certain experiments to evaluate the effectiveness of the proposed approach in the context of organization of design patterns and three well-known supervised learning technique (Appendix D) named Naive Bayes, C4.5 Decision Tree, and Support Vector Machines (SVM).

7.2. Proposed Method

We consider the design pattern organization as an information retrieval problem [32, 33, 168]. We look at the capacity of deep learning algorithm to address the gap between the semantic relationship between design patterns and the features which are used for the organization of design patterns.

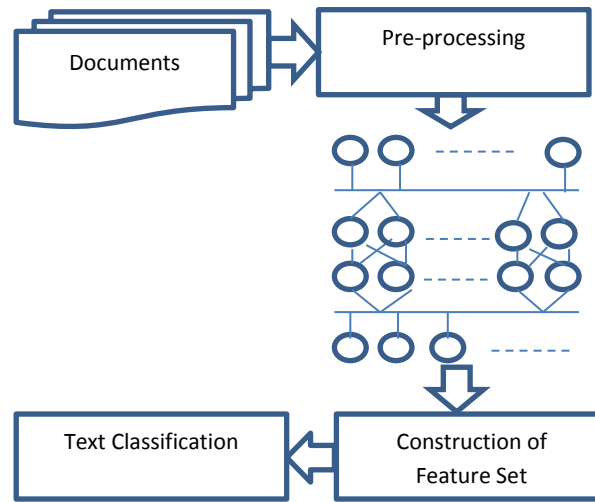


Figure 7.1. Overview of the proposed method

The aim of the proposed method is to construct a more illustrative feature set to improve the performance of classifiers employed in the text categorization based automated system. The proposed approach (shown in Figure 7.1) is applied in three phases named Pre-processing, Construction of Feature Set and text classification.

7.2.1. Preprocessing

In the preprocessing phase, four activities (shown in the Figure 7.2) are performed to transform documents into strings and formulate them into feature vector form. The first activity is to remove stop words that are more frequent words and carry no information, such as Preposition, Conjunction, and Pronouns. The second activity is word stemming, which is performed to make a group of words that have same conceptual meaning. Subsequently, the numbers of words in the documents collection are decreased. Porter's stemmer is a well-recognized stemming algorithm [70], used to transform English words into their stem iteratively through a set of rules. The third activity is related to present the feature vectors into integer vector as prerequisite for DBN algorithm [166]. In the text classification numerous weighting methods such as Term Frequency (TF), Binary, Term Frequency Inverse Document Frequency (TFIDF), Term Frequency Collection (TFC), Length Term Collection (LTC), and Entropy are

recommended. In this study, we use TF for representation of an integer vector. The size of feature vectors is same as prerequisite inputs to DBN algorithm. Finally, the feature selection methods besides the proposed approach are applied to construct the feature sets for training of classifiers.

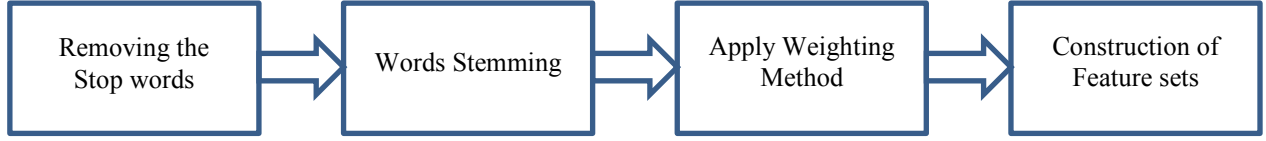


Figure 7.2. Set of preprocessing activities

7.2.2. Construction of Feature Set

In the construction of Feature set phase, the DBN algorithm is applied to generate the features for the classification of documents into proper classes. The aim of DBN learns from given input data (in our case feature vectors) via multi-level neural network and reconstructs the content of input data with high probability. The DBN includes one input, N hidden, and one output (i.e. Top) layer with a set of several nodes. The number of hidden layers and nodes can be tuned by user accordingly. The equation 7.1 describes the joint distribution of input and hidden layers.

$$P(v, h^1, \dots, h^l) = P(v | h^1) \left(\prod_{k=1}^l P(h^k | h^{k+1}) \right) \quad (7.1)$$

Where the term v referred as feature vector from input layer, l is referred as total number of hidden layers and h^k referred to a feature vector of k^{th} layer. The conditional distribution for two adjacent layers (for example k and $k+1$) is referred as $P(h^k | h^{k+1})$ and calculated through Restricted Boltzmann Machines (RBM). In case of node, DBN learns probabilities from any current node to the upper level's nodes. The backward validation is performed by DBN to reconstruct the input feature set by tuning the weights between nodes across the layers.

7.2.3. Text Classification

There are several supervised learning techniques can be used in the text classification based automated system. It cannot be decided that which one is better since no classifier can get better results for all problems. Consequently, it is recommended to find the best classifier for a new problem. In the text classification domain, researchers have used and reported the classification performance of C4.5 Decision Tree, Naïve Bayes, and Support Vector Machine (SVM) supervised learning techniques [47]. The detail of each supervised learning algorithm is given in the Appendix D. In this study, we used these classifiers to evaluate the effectiveness of the proposed approach.

7.3. Experimental Setup

We performed a set of experiments to investigate the effectiveness of the proposed approach against the individual effect of four global filter-based feature selection methods named Document Frequency (DF), Improved Gini Index, Gain Ratio, and Information Gain, on the performance of classifiers for the organization of a target pattern catalog. We have summarized the experimental procedure and presented through the following pseudocode.

Pseudocode: Pattern_Organization (PC, SC, N, M)

Input:

- PC = {The set of problem domain description of design patterns in the given Pattern Collection}
- SL = {Set of Supervised Learners under study, such as Naïve Bayes (NB), C4.5 Decision Tree, and Support Vector Machine (SVM))}
- N = The number of design patterns in a collection, For example, in case of GoF and Douglass pattern collection the value of N is 23 and 34 respectively.
- M = The number of unique words, For example, in case of GoF and Douglass pattern collection the value of M is 1465 and 1271 respectively.

Procedure Begin:

- Step-1.** Perform the first three activities of the preprocessing phase of the proposed approach to present N feature vectors with M unique words.
- Step-2.** Apply a weighting method (such as Binary, TF, TFIDF, TFC, LTC, and Entropy)
- Step-3.** Apply a Supervised learning technique to organize the N design patterns.
- Step-4.** Apply the evaluation criteria (Section 4.1.1) to assess the performance of the supervised learner with the corresponding weighting method.
- Step-5.** Repeat Steps 2, 3 and 4 until the all possible assessments (according to number of classifiers and weighting methods used in the proposed study) have done.
- Step-6.** For each Supervised learner select the best weighting method, and classifies the patterns into appropriate categories.
- Step-7.** Apply the evaluation criteria (Section 4.1.1) to assess the performance of the supervised learner with the their best weighting methods.
- Step-8.** In order to evaluate the improvement in the effectiveness of supervised learner, apply feature selection method (Such as Document Frequency (DF), Information Gain (IG), Mutual Information (MI), Correlation Coefficient (CC) and Chi-Square (CHI)) and proposed approach (Exploit through a deep learning algorithm DBN) to rank the top n features.
- Step-9.** Apply the evaluation criteria (Section 4.1.1) to assess the performance of the supervised learner with the their best weighting methods and top n ranked features (Step-8)

Procedure End:

Output:

The outperform supervised learner with best weighting and feature selection method for the organization of design patterns catalog (i.e. PC).

In this case study, we consider three often used design pattern collection named Gang-of-Four (GoF), Douglass, and Security pattern collections.

7.4. Results and Discussion

For each design pattern collection under study, the pseudocode (Section 5.1) is called to find the 1) best weighting method for each supervised learner, 2) outperform classifiers with their best weighting method, 3) effect of the constructed feature set (via proposed approach and global filter-based feature selection method under study) on the performance of classifiers, and 4) improvement in the performance of classifiers by applying the proposed approach. Subsequently, in each experiment, We consider the pattern category as dependent variable while extracted features as independent variables. Subsequently, we performed widely used 10-fold cross validation [169] to improve the classification.

7.4.1. Organization of GoF Design Patterns

The patterns in the GoF pattern collection are divided into three categories. The preprocessing activities of the proposed approach (Section 7.2.1) are applied to construct the dataset. We follow the procedure (described in pseudocode) and use the F-measure criterion (Section 4.1.1) to find the best weighting method for each supervised learner. According to the F-measure criterion, the evaluation result in the selection of best weighting method for each supervised learner is shown in Figure 7.3. Subsequently, the results about the best weighting for each classifier are shown in the Table 7.2. The main consequences from the result of Figure 7.3 and Table 7.2 are as follows.

- We observe the better performance of NB, SVM, and DT with TFC, TFC and TDIDF weighting methods.
- We observe the DT (F-measure = 0.80) with its best weighting method TDIDF as outperform classifiers as compared to NB (F-measure = 0.76) and SVM (F-measure = 0.78).

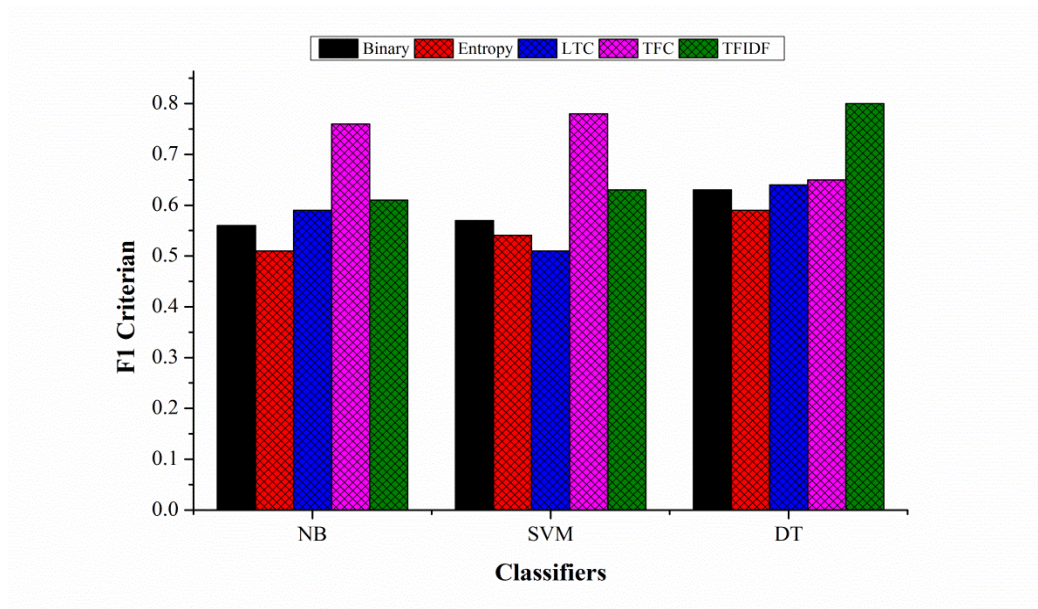
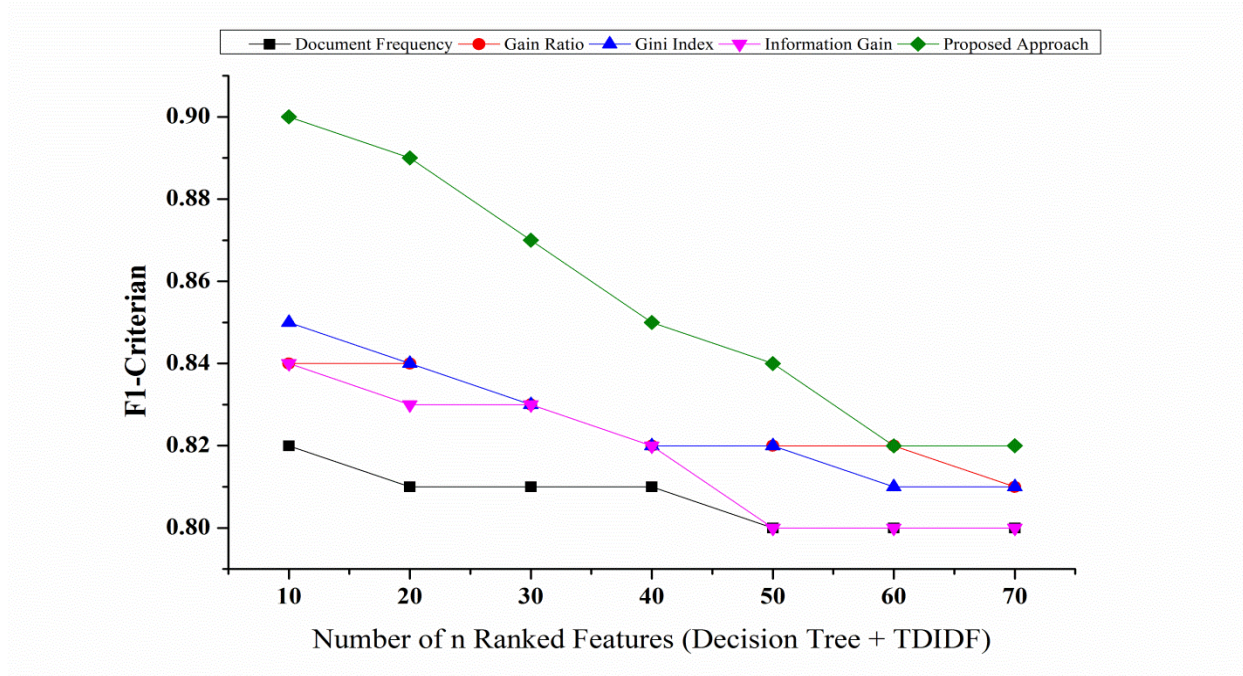


Figure 7.3. Performance evaluation of classifiers for the weighting methods on the GoF design pattern collection

Table 7.2. Performance evaluation of classifiers on different design pattern catalogs

Pattern Catalog (i.e. Dataset)	Classifiers	Best Weighting Method	F-measure
GoF Pattern Catalog	DT	TDIDF	<u>0.80</u>
	NB	TFC	0.76
	SVM	TFC	0.78
Douglass Pattern Catalog	DT	TDIDF	0.80
	NB	TFC	0.79
	SVM	TFIDF	<u>0.83</u>
Security Pattern Catalog	DT	TFC	0.75
	NB	LTC	0.79
	SVM	TFC	<u>0.82</u>

Subsequently, to evaluate the effectiveness and improvement (in terms of the F-measure criterion) in the performance of supervised learners used in the proposed framework, we applied the feature selection methods and the proposed approach to construct a more representative feature set. We performed experiments with all supervised learners. Subsequently, we depict the effect of feature selection methods on the performance of outperform supervised learner that is the Decision Tree (with TDIDF) in Figure 7.4. The x-axis of Figure 7.4 (labelled with Number of Rank Features) refers to top n features rank by each feature selection method and the proposed approach. The y-axis refers to the corresponding F-measure values. The main consequences from the result of Figure 7.4 are.

**Figure 7.4.** Performance evaluation of classifiers with feature sets constructed through feature selection method and proposed approach GoF pattern collection

- We observed the significant impact of the feature set constructed through the employed feature selection methods and proposed approach on the performance of classifiers.
- The average improvement in the classification performance of outperform supervised learner (i.e. Decision Tree) is ranges from 2.5% to 11.25% (in terms of F-measure) with top 10 features ranked by each feature selection method under study.
- The highest increase in the performance (i.e. 12.5%) of Decision Tree with top 10 features ranked through the proposed approach, which aid us to describe the significance of the proposed approach to construct a more representative feature set.

7.4.2. Organization of Douglass Design Patterns

The patterns in the Douglass pattern collection are divided into five categories. The preprocessing activities of the proposed approach (Section 7.2.1) are applied to generate the dataset. We follow the procedure (described in pseudocode) and use the F-measure criterion (Section 4.1.1) to find the best weighting method for each supervised learner. According to the F-measure criterion, the evaluation result in the selection of best weighting method for each supervised learner is shown in Figure 7.5. Subsequently, the results about the best weighting for each classifier are shown in the Table 7.2. The main consequences from the result of Figure 7.5 and Table 7.2 are.

- We observe the better performance of NB, SVM, and DT with TFC, TDIDF and TDIDF weighting methods.
- We observe the SVM (F-measure = 0.83) with its best weighting method TDIDF as outperform classifiers as compared to NB (F-measure = 0.79) and DT (F-measure = 0.80).

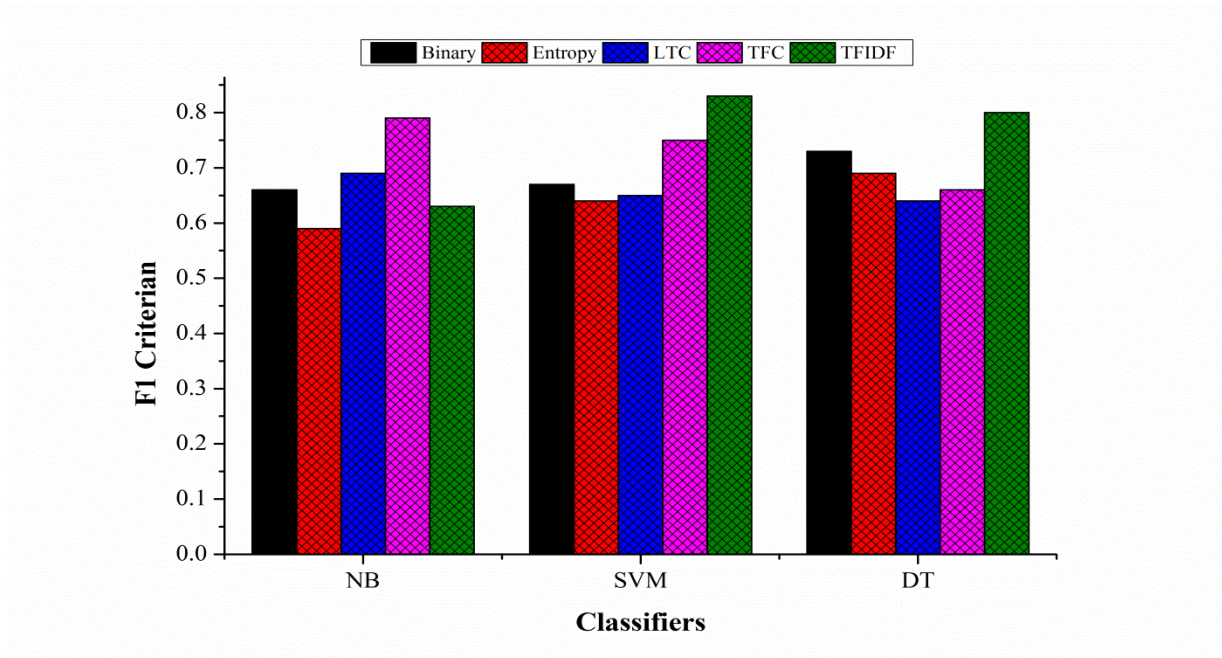


Figure 7.5. Performance evaluation of classifiers for the weighting methods on the Douglass Design pattern collection

Subsequently, to evaluate the effectiveness and improvement (in terms of the F-measure criterion) in the performance of supervised learners, we applied all feature selection methods and the proposed approach to construct a more representative feature set. We performed experiments with all supervised learners. Subsequently, we depict the effect of feature selection methods on the performance of outperform supervised learner that is SVM (with TDIDF) in Figure 7.6. The x-axis of Figure 7.6 (labeled with Number of Rank Features) refers to top N features rank by each feature selection method and the proposed approach. The y-axis refers to the corresponding F-measure values. The main consequences from the result of Figure 7.6 are.

- We observed the significant impact of the feature set constructed through the employed feature selection methods and proposed approach on the performance of classifiers.
- The average improvement in the classification performance of outperform supervised learner (i.e. SVM) is ranges from 2.40% to 10.84% (in terms of F-measure) with top 10 features ranked by each feature selection method under study.

The highest increase in the performance (i.e. 10.84%) of SVM with top 10 features ranked through the proposed approach, which aid us to describe the significance of the proposed approach to construct a more representative feature set.

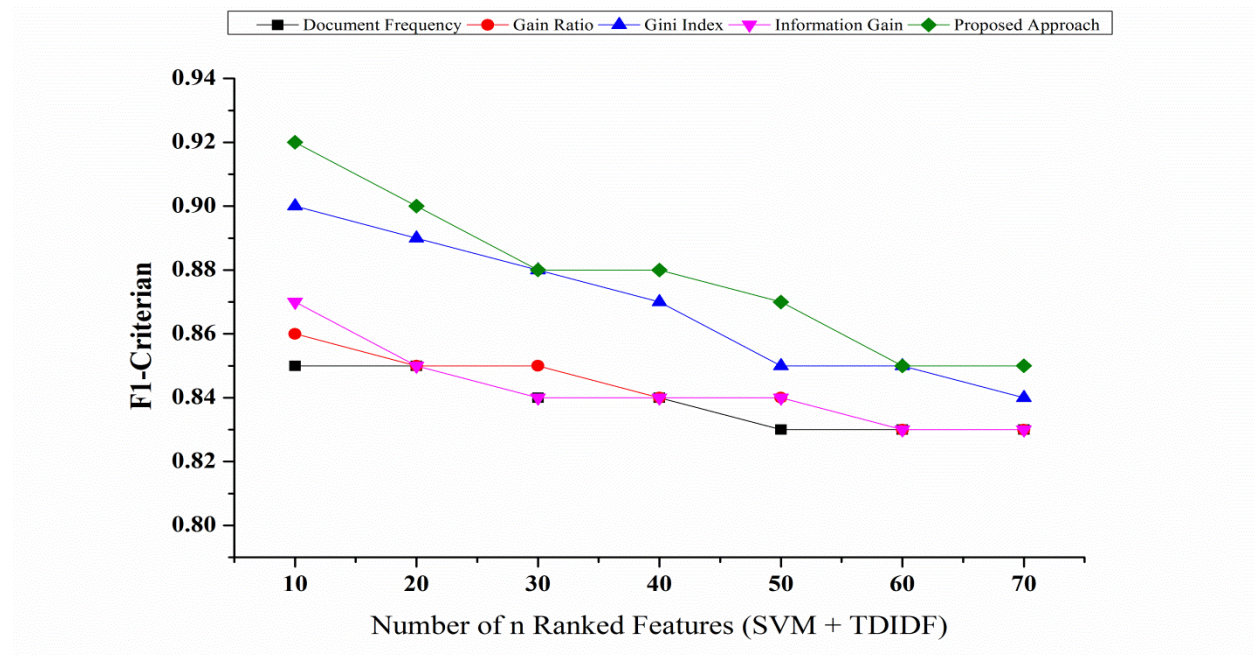


Figure 7.6. Performance evaluation of classifiers with the feature sets constructed through feature selection method and proposed approach on Douglass pattern collection

7.4.3. Organization of Security Design Patterns

The patterns in the Security pattern collection are divided into eight categories. The preprocessing activities of the proposed approach (Section 7.2.1) are applied to generate the dataset. We follow the procedure (described in pseudocode) and use the F-measure criterion (Section 4.1.1) to find the best weighting method for each supervised learner. According to the F-measure criterion, the evaluation result in the selection of best weighting method for

each supervised learner is shown in Figure 7.7. Subsequently, the results about the best weighting for each classifier are shown in the Table 7.2. The main consequences from the result of Figure 7.7 and Table 7.2 are.

- We observe the better performance of NB, SVM, and DT with TFC, TDIDF and TDIDF weighting methods respectively.
- We observe the SVM (F-measure = 0.82) with its best weighting method TFC as outperform classifiers as compared to NB (F-measure = 0.79) and DT (F-measure = 0.75).

Subsequently, to evaluate the effectiveness and improvement (in terms of the F-measure criterion) in the performance of supervised learners, we applied all feature selection methods and the proposed approach to construct a more representative feature set. We performed experiments with all supervised learners. Subsequently, we depict the effect of feature selection methods on the performance of outperform supervised learner that is SVM (with TFC) in Figure 7.8.

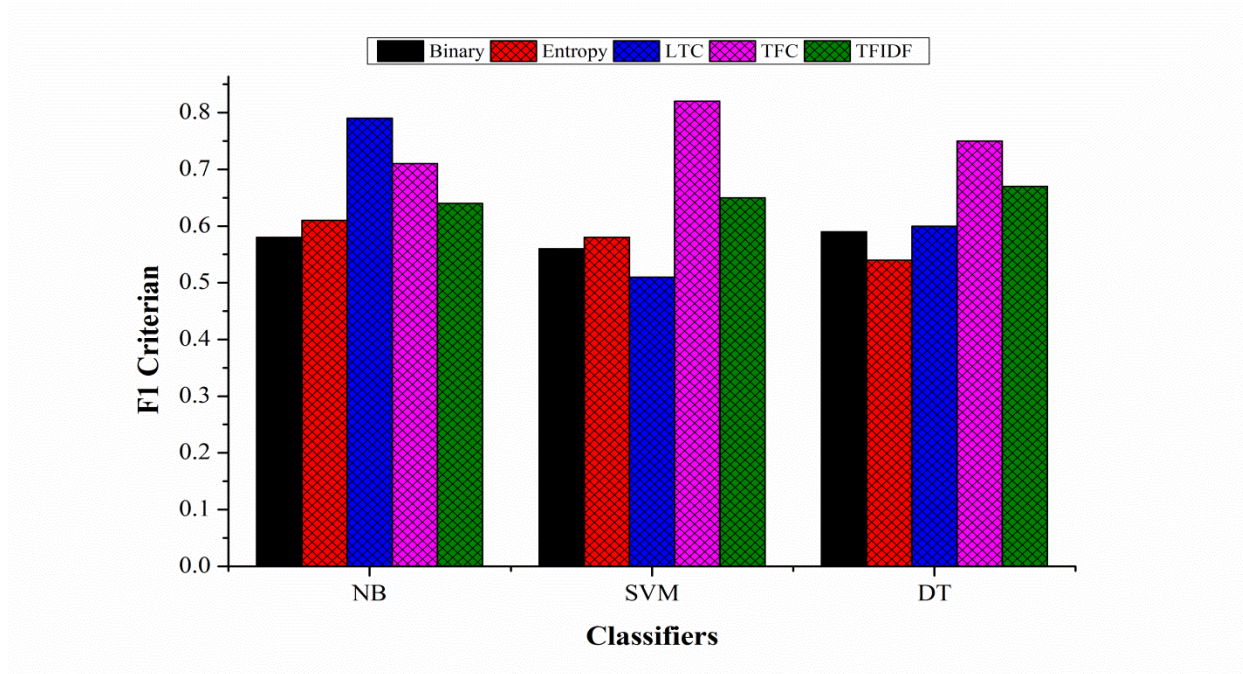


Figure. 7.7. Performance evaluation of classifiers for the weighting methods of the Security design pattern collection

The x-axis of Figure 7.8 (labeled with Number of Rank Features) refers to top N features rank by each feature selection method and the proposed approach. The y-axis refers to the corresponding F-measure values. The main consequences from the result of Figure 7.8 are.

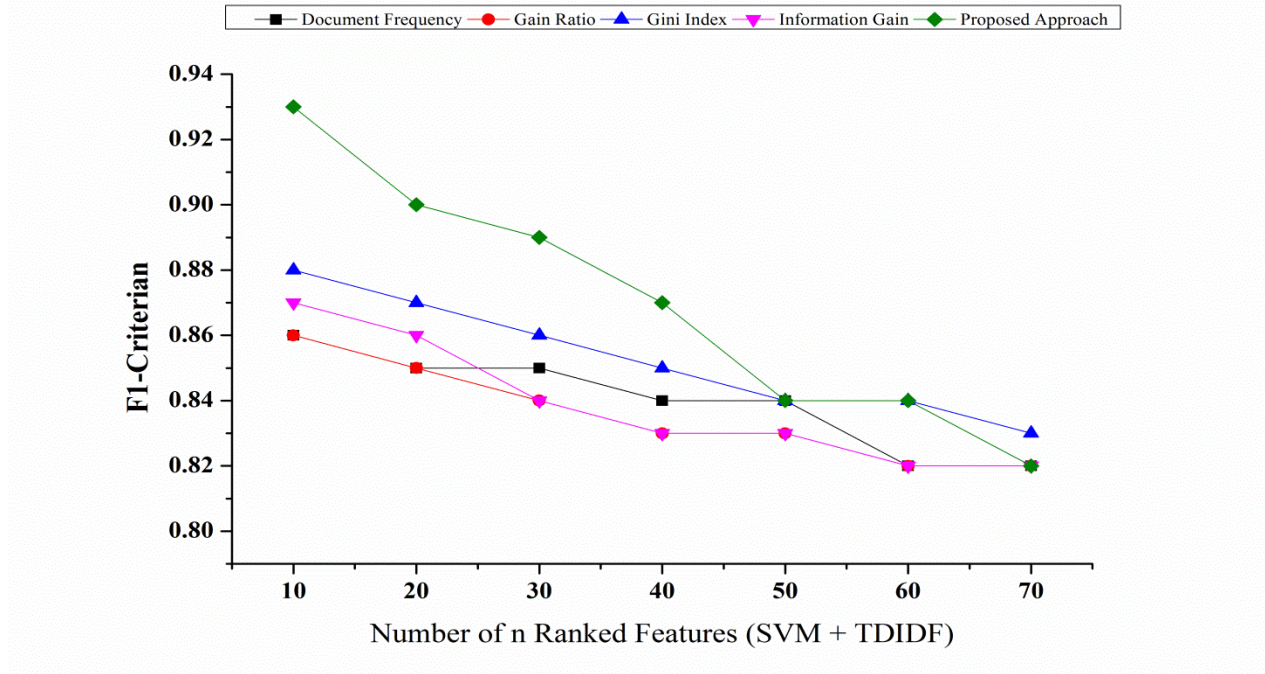


Figure 7.8. Performance evaluation of classifiers with the feature sets constructed through feature selection method and proposed approach on Security pattern collection

- We observed the significant impact of the feature set constructed through the employed feature selection methods and proposed approach on the performance of classifiers.
- The average improvement in the classification performance of outperform supervised learner (i.e. SVM) is ranges from 4.80% to 31.41% (in terms of F-measure) with top 10 features ranked by each feature selection method under study.
- The highest increase in the performance (i.e. 13.41%) of SVM with top 10 features ranked through the proposed approach, which aid us to describe the significance of the proposed approach to construct a more representative feature set.

7.5. Consequences of Case Study

The main objective of the proposed framework is to use a well-known deep learning algorithm to construct a more representative feature set and evaluate its impact on the organization of design patterns. We employed the proposed approach within the context of three widely used design pattern collection. The main consequences of our proposed study are.

- We applied three supervised learning techniques to evaluate the effectiveness of the proposed approach. In the evaluation results, we observe the significant increase in the performance of classifiers for the organization of design patterns. For example, for the organization of Security design patterns, we observe 13.41% (in terms of F-measure) improve in the actual performance (F-measure = 0.82) of SVM.

- The proposed approach is employed through supervised learning techniques. Consequently, before performing comparative investigations, we used different weighting methods (such as TFIDF, TFC, LTC, Binary, and Entropy) to increase the efficacy of learning techniques. The evaluation results indicate that no single weighting method can be suggested for all supervised learner. For example, in the case of the organization of Douglass and Security design patterns, SVM outperform with different weighting methods, such as TFC for security patterns and TFIDF for the Douglass patterns.
- From the experimental results, we observed the effect of feature selection methods on the performance of supervised learners but it is depend on the target pattern catalog. For example, the improvement in the performance of outperform classifier SVM is 10.84% and 13.41% for Douglass and Security pattern collection respectively.
- We observed the significant increase in the performance of classifiers with a feature set constructed through the proposed approach as compared to feature sets constructed through other feature selection methods. For example, in case of Security patterns, the performance of SVM increases up to 13.41% with a feature set constructed through the proposed approach.

8.1. Implications and Conclusions

The experimental results illustrate that the proposed approach based on text categorization approach via unsupervised learners aid to give a promising base for selection of more right design pattern(s) for a particular design problem. While existing automatic techniques based on UML-based, Ontology-based, and text categorization approaches aims to reduce the preprocessing cost and computation time to select a right design pattern(s) for the given design problem. The formal specification, the existence of the ground reality of class labels for design patterns, multi-class problem, and a large sample size is required for these automatic techniques. To address this issue, the proposed approach is evaluated with the use of three design pattern collections and descriptions of many real design problems. Moreover, we also proposed a feature selection method Ensemble-IG by leveraging an Improved Global Feature Selection Scheme (IGFSS) and ensemble Odds Ratio OR) with Information Gain (IG) to overcome the multi-class problem. The proposed approach has three phases, Preprocessing, Pattern Organization and Determination of design pattern class, and Suggestion of Design Pattern(s). The Fuzzy c-means, Agglomerative, K-means, Partition Around Medoids (PAM) (with Euclidean and Manhattan distance measures) unsupervised learning techniques are used in the proposed approach. Numerous performance measures are used to evaluate the quality of data clustering. We also propose an evaluation model by leveraging the well-known measures to evaluate the effectiveness of the proposed approach. Numerous experiments are performed to meet the target objectives mentioned in Section 1.4.

Numerous experiments are performed to investigate the certain aspects related to the effectiveness of unsupervised learner employed in the proposed approach, such as the effect of weighting method and feature selection techniques. The main consequences and conclusion of the experimental results are as follows.

1. We used different weighting methods (such as TFIDF, TFC, LTC, Binary, and Entropy) in order to increase the efficacy of proposed approach, but, no single weighting method can be recommended. For example, In case of pattern organization of GoF (F-measure = 0.73), Douglass(F-measure = 0.71), Duyne(F-measure = 0.71) pattern collection, the highest F-measure values for TDIDF recommend it as a best weighting method. In case of security (F-measure = 0.57) design pattern collection, the TFC is considered as a best weighting method. In regard of selecting a best weighting method, other implications are.

- In many cases during the analysis of four case studies, TFC and TFIDF are recommended as the best weighting method for unsupervised learners.
- The analysis of four case studies indicates that no single weighting method cannot be recommended for the unsupervised learner of the proposed approach.
- In many cases such as Douglass, Security and Duyne pattern collection, the Binary weighting method is not recommended as a best weighting method for unsupervised learners employed in the proposed approach.

Conclusion-1: In the regard of organization of design patterns, the performance of an unsupervised learner cannot be decided by recommending a single weighting method, such as the summarized results (Table 8.1) of all experiments (for organization of design patterns) show that the effectiveness of unsupervised learning techniques cannot be credited to a single weighing method. We need to replicate the experiments to find the best weighting

method for an unsupervised learner within the context of any given design pattern collection. In the proposed study, we observed the better performance of unsupervised learner with TFIDF and TFC weighting methods.

Table 8.1. Performance of unsupervised learner with all weighting methods in terms of organization of design pattern collections under study

Case Study	Outperform Unsupervised Learner	Best Weighthing Method	F-measure
GoF Design Patterns	Fuzzy c-means	TFIDF	0.73
	Agglomerative	Entropy	0.74
	K-means	Binary	0.70
	PAM-Euclidean	TFIDF	0.78
	PAM-Manhattan	Binary	0.65
Douglass Design Patterns	Fuzzy c-means	TFIDF	0.71
	Agglomerative	TFIDF	0.68
	K-means	LTC	0.74
	PAM-Euclidean	TFIDF	0.56
	PAM-Manhattan	TFIDF	0.68
Security Design Patterns	Fuzzy c-means	TFC	0.57
	Agglomerative	TFIDF	0.50
	K-means	TFC	0.43
	PAM-Euclidean	Entropy	0.61
	PAM-Manhattan	TFIDF	0.52
Duyne Design Patterns	Fuzzy c-means	TFIDF	0.71
	Agglomerative	TFIDF	0.68
	K-means	TFC	0.70
	PAM-Euclidean	TFC	0.71
	PAM-Manhattan	TFC	0.65

2. We applied the widely known unsupervised learning techniques in the proposed approach and get better results. The performance of unsupervised learning technique varies with respect to target design pattern collection and design problem under study. For example, in case of pattern organizations of GoF (ARI=0.65), Douglass(ARI=0.87), and Security (ARI=0.62) design pattern collections, the Fuzzy c-mean is recommended as an outperform unsupervised learner in terms of Adjusted Rand Index (ARI), while in case of Duyne(ARI=0.71) design pattern collection, PAM-Euclidean is considered as an outperform unsupervised learner. Consequently, no single unsupervised learner can be recommended to organized patterns and determination of right pattern class for the given design problem(s). In regard of determining the outperform unsupervised learner, other implications are.

- In many cases during the analysis of four case studies, the Agglomerative does not perform well as compared to other unsupervised learner of the proposed approach.

- In many cases during the analysis of four case studies, we did not observe the significance difference between the performance of PAM-Euclidean and PAM-Manhattan.

Conclusion-2: In the regard of organization of design patterns, Fuzzy c-means can be recommended for to exploit the proposed approach, such as shown in Table 8.2. A developer can replicate the case study via Fuzzy c-means for design pattern collections used in the proposed study. The performance of fuzzy c-means might be effected in the context of other design pattern collection, which are not considered in the proposed study.

Table 8.2. Outperform learners for the organization of design patterns of target collections

Pattern Collection	Outperform Unsupervised Learner	Best Weighthing Method	F-measure
GoF Design Patterns	Fuzzy c-means	TFIDF	0.73
Douglass Design Patterns	Fuzzy c-means	TFIDF	0.71
Security Design Patterns	Fuzzy c-means	TFC	0.57
Duyne Design Patterns	PAM-Euclidean	TFC	0.70

3. Subsequently, in terms of determining the right pattern class for the sample of given design problems, we observed the Fuzzy c-means as an outperform unsupervised learner. For example, in case of GoF (ARI=0.55), Douglass (ARI=0.80), Security (ARI=0.74), and Duyne (ARI=0.69) design pattern collections, the Fuzzy c-mean is recommended as an outperform unsupervised learner in terms of the Adjusted Rand Index (ARI).

Conclusion-3: In the regard of determining the right design pattern class for the design problems in the given sample, the performance of outperform unsupervised learner of each case study remains better (Table 8.3) which show that outperformed unsupervised learner with its best weighting method can be recommended to find the design pattern class for the given sample of design problems.

Table 8.3. Effectiveness of Outperform learners to determine the right pattern class for the given samples of design problems.

Pattern Collection	Outperform Unsupervised Learner	Best Weighthing Method	ARI For Sample-1 Design Problems	ARI For Sample-2 Design Problems
GoF Design Patterns	Fuzzy c-means	TFIDF	0.55	0.50
Douglass Design Patterns	Fuzzy c-means	TFIDF	0.80	0.79
Security Design Patterns	Fuzzy c-means	TFC	0.70	0.70
Duyne Design Patterns	PAM-Euclidean	TFC	0.71	0.68

4. We applied each unsupervised learning technique in the proposed approach with their default parameter setting. For example, in case of Fuzzy c-means, weighting component (i.e. m) and weight Matrix (i.e. A) are the two main parameters of the objective function. The aim of m (i.e. $1 \leq m < \infty$) is to control the relative weights which are used for the squared error D_{ikA}^2 . The increase in the value of m inclines to reduce the membership for the fuzziest state. The best value of m can be derived through an experimental investigation which is usually in $1.5 \leq m \leq 3.0$ range. The second parameter of the objective function J_m of Fuzzy c-means is the weight matrix A , which controls the

cluster's shape in $\mathbb{R}$. There are several A-norms such as Euclidean ($A = I$), Diagonal ($A = D_y^{-1}$), and Mahalonobis ($A = C_y^{-1}$) Norms are available. The performance of unsupervised learning techniques might be affected by tuning the default setting of their parameters.

5. The results of feature selection methods for the Fuzzy c-means indicate that learning precision can be increased by reducing the features until it is found at maximum. We observed that the Fuzzy c-means is more sensitive to proposed feature selection method Ensemble-IG as compared to other feature selection methods. Subsequently, in our experiment, learning precision is evaluated with top N (from 10 to 350) features rank by feature selection methods. We observe an increase in the learning precision (in term of F-measure) of Fuzzy c-means with proposed Ensemble-IG as compared to other feature selection methods used in the study. In regard of selecting a best feature selection method, other implications are.

- In case of pattern composition of GoF design pattern collection, we observed a significance increase (15.06%) in the performance of the outperform unsupervised learner Fuzzy c-means (with TDIDF best weighting method) with Ensemble-IG in terms of F-measure.
- In case of pattern composition of Douglass design pattern collection, we observed a significance increase (5.63%) in the performance of the outperform unsupervised learner Fuzzy c-means (with TDIDF best weighting method) with Ensemble-IG in terms of F-measure.
- In case of pattern composition of Security design pattern collection, we observed a significance increase (14.03%) in the performance of the outperform unsupervised learner Fuzzy c-means (with TFC best weighting method) with Ensemble-IG in terms of F-measure.
- In case of pattern composition of Security design pattern collection, we observed a significance increase (11.39%) in the performance of the outperform unsupervised learner PAM-Euclidean (with TFC best weighting method) with Ensemble-IG in terms of F-measure.

Table 8.4. Improvement in the performance of outperform unsupervised learner via the best and proposed feature selection methods (DF=Document Frequency, IG=Information Gain, Corr=Correlation, GR=Gain Ratio)

Case Study	Outperform Unsupervised Learner	Best Weighting method	F-measure	With top 10 Features					Improvement (%)
				DF	Ch-Sq	IG	Corr	GR	
First	Fuzzy c-means	TFIDF	0.73	0.78	0.79	0.80	0.83	0.80	13.69
Second	Fuzzy c-means	TFIDF	0.71	0.72	0.73	0.74	0.73	0.73	4.22
Third	Fuzzy c-means	TFC	0.57	0.61	0.62	0.63	0.60	0.62	10.52
Fourth	PAM-Euclid	TFC	0.71	0.76	0.75	0.77	0.76	0.79	11.26

Conclusion-4: In the regard of the sensitivity of the proposed approach with feature selection methods, the performance of outperform unsupervised learners can be significantly increased by constructing a representative feature set via global filter based feature selection methods, as shown in Table 8.4. Subsequently, the results of

Table 8.5 indicate the significant increase in the performance of outperform unsupervised learner by constructing the representative feature set via proposed Ensemble-IG feature selection method.

Table 8.5. Improvement in the performance of outperform unsupervised learner via the best and proposed feature selection methods.

Pattern Collections	Outperform Unsupervised Learner	Best Weighthing Method	F-measure	Best Feature Selection (F-measure)	Ensemble-IG (F-measure)
GoF Design Patterns	Fuzzy c-means	TFIDF	0.73	Correlation (0.83)	0.84
Douglass Design Patterns	Fuzzy c-means	TFIDF	0.71	Information Gain (0.74)	0.75
Security Design Patterns	Fuzzy c-means	TFC	0.57	Information Gain (0.63)	0.65
Duyne Design Patterns	PAM-Euclidean	TFC	0.70	Gain Ratio(0.77)	0.79

6. In case of design pattern selection, cosine similarity measure and three similarity options are suggested. The best similarity option in the proposed approach is recommended on the base of RCD (Ratio of Corrected Design patterns) value. The values of Θ_1 (Second similarity option) and Θ_2 (Third similarity option) parameters depend on the selected design pattern collection and the sample of selected design problems. The developer can replicate these experiments with any sample of design problems and can tune these parameters.

Conclusion-5: In the regard of right pattern selection via the proposed approach, the first similarity option with highest RCD value is recommended to developer to select the right design pattern for the given samples of design problem. Such as, in the case of GoF, Douglass, Security, and Duyne's patterns selection of design problems (Sample-1), the corresponding RCD values are 0.55, 0.80, 0.70, and 0.71 (for first similarity option).

In a future work, the proposed approach can aid software practitioners in several aspects such as:

- We will consider more unsupervised learner and will use the semantic of two contiguous words instead of the single word in the feature vector to improve the effectiveness of the proposed approach.
- Subsequently, we will also consider the Stack Exchange sites such as Information Security, Graphic Design, Web Application, Board and Card Games and Game Development to analyze the variation of problems and suggestion of design patterns.
- We used the deep learning approach to construct more representative feature sets and employed via the supervised learning techniques. The promising results of the case study can motivate a software practitioner of any domain to adopt this approach to improve the performance of text categorization based automated systems.
- The promising results of proposed approach can motivate the novice designer for the selection of design pattern(s) for a design problem at hand.

Appendix A: Templates of Design Patterns

Part A: GoF Design Pattern Template (Composite Design Pattern as an Example)

Design Pattern Name	Composite
Problem Domain	
Intent	“Compose objects into tree structures to represent part-whole hierarchies. Composite lets client treat individual objects and composition of objects uniformly”.
Motivation	“Graphics applications like drawing editors and schematic capture system let users build complex diagrams out of simple components. The user can group components to form larger components, which in turn can be grouped to form still larger components. A simple implementation could define classes for graphical primitives such as Text and Lines plus other classes that act as containers for these primitives. But there’s a problem with this approach: Code that uses these classes must treat primitives and container objects differently, even if most of the time they use treats them identically. Having to distinguish these objects makes the application more complex. The Composite pattern describes how to use recursive composition so that clients don’t have to make this distinction”. The rest of motivation section is available in [3]
Applicability	“The composite pattern applies when there is a part-whole hierarchy of objects and a client needs to deal with objects uniformly regardless of the fact that an object might be a leaf or a branch”.
Solution Domain	
Structure	<pre> classDiagram class Leaf { +Operation() } class Component { +Operation() +Add(Component) +Remove(Component) +GetChild(int) } class Composite { +Operation() +Add(Component) +Remove(Component) +GetChild(int) } Leaf < -- Component Component < -- Composite Composite --> Leaf </pre>
Participant Collaborations	Component, Leaf, Composite and Client.
Consequences	<p>The Composite Pattern</p> <ul style="list-style-type: none"> • “Defines class hierarchies consisting of primitive objects and composite objects. Primitive objects can be composed and so on recursively. Whatever client code expects a primitive object, it can also take composite object”. • “Makes the client simple. The client can treat composite structure and

individual object uniformly. The client normally don't know (and should not care) whether they are dealing with a leaf or composite component. This simplifies client code, because it avoids having to write tag-and-case-statement-style functions over the class that defines the composition".

- "Makes it easier to ass new kind of components. Newly defined Composite or leaf subclasses work automatically with existing structure and client code. Clients don't have to be changed for new Component classes".
- "Can make your design overall general. The disadvantage of making it easy to add new components is that make it harder to restrict the components of a composite. Sometime you want a composite to have only certain components. With Composite, you can't rely on the type system to enforce those constraints for you. You will have to use run-time checks instead".

Implementation

"The implementation of Composite Design Pattern is available in [5] using C++ constructs".

Related Patterns

"Decorator Pattern: When decorators and composites are used together, they will usually have a common parent class".

Part B: Douglass Design Pattern Template (Pool Allocation Pattern as an Example)

Design Pattern Name	Pool Allocation
Problem Domain	
Abstract	<p>“In many applications, objects may be required by a large number of clients. For example, many clients may need to create data objects or message objects as the system operates in a complex, changing environment. The need for these objects may come and go, and it may not be possible to predict an optimal dispersement of the objects even if it is possible to bound the total number of objects needed. In this case, it makes sense to have pools of these objects—created but not necessarily initialized and available upon request. Clients can request them as necessary and release them back to the pool when they're done with them.”</p>
Problem	<p>“The prototypical candidate system for the Pooled Allocation Pattern is a system that cannot deal with the issues of dynamic memory allocation, but it is too complex to permit a static allocation of all objects. Typically, a number of similar, typically small, objects, such as events, messages, or data objects, may need to be created and destroyed but are not needed a priori by any particular clients for the entire run-time of the system.”</p>
Solution Domain	
Structure	<pre> classDiagram Client --> ResourcePoolManager : object server class ResourcePoolManager { <<abstract>> } class ResourceClass ResourcePoolManager ..> ResourceClass ResourcePoolManager ..> GenericPoolManager class GenericPoolManager { + Initialize + Allocate + Release } GenericPoolManager *-- PooledClass1 : PooledClass GenericPoolManager *-- PooledClass2 : PooledClass PooledClass1 ..> GenericPoolManager : HasObject </pre>
Collaboration Roles	Client, Generic Pool Manager, PooledClass, Resource Pool Manager
Consequences	<p>“The Pooled Allocation Pattern has a number of advantages over the Static Allocation Pattern. Since memory is allocated at startup and never deleted, there is no problem with the nondeterministic timing of dynamic memory allocation during run-time or memory fragmentation. The system, however, can handle nondeterministic allocation of certain classes of objects. Thus, the pattern scales to more complex systems than do the Static Allocation Pattern. It is especially applicable to systems where a number of common objects may be needed by many different clients, but it cannot be determined at design time how to distribute these objects. This pattern allows the objects to be distributed on an as-needed basis during the execution of the system. Because all pooled objects are created at system startup, the decision about the optimal number of different kinds of objects must be made at design time. For example, it might be decided that as a</p>

worst case, 1000 message objects, 5000 data sample objects, and so forth, may be required. If this turns out to be an erroneous decision, the system may fail at startup or later during execution. Further, the system cannot grow to meet new system demands, so the pattern is best applied to systems that are well understood and relatively predictable in their demands on system resources.”

Implementation
Strategies

“This common pattern is fairly easy to implement. To make the pattern easier to use in C++, it is common to rewrite new and delete operators to use the pool manager for the various pooledClass types. That way, the issue of dynamic versus pooled allocation can be hidden from the application programmer.”

Related Patterns

“The Pooled Allocation Pattern is but one of many approaches to managing memory allocation. The Static Allocation Pattern can be used for systems that are simpler, and the Dynamic Allocation Pattern and its variants can be used for more complex needs.”

Part C: Security Design Pattern Template (Identification and Authentication as an Example)

Design Pattern Name	Identification and Authentication
Problem Domain	
Context	“The scope is known to be situations in which both identification and authentication are needed. Other situations exist in which only identification is needed without authentication, but those situations are not addressed in this pattern.”
Problem	“Requirements for I&A often conflict with each other, and trade-offs among them are often necessary. The conflict stated in the Example section is that strength of protection with I&A tends to conflict with ease of use. I&A comprises both associating an identifier with an actor (identification) and verifying that the association is correct (authentication). I&A is a security service whose results are often used by other security services, including access control and accounting. A basic set of generic I&A requirements exists for all types of use and circumstances. However, these generic requirements need to be specialized for a given I&A domain. In addition, the relative importance of the requirements will vary based on the circumstances. What is needed is (1) to capture the specific set of requirements, and (2) to understand how to differentiate the relative importance of the requirements in specific circumstances to balance or resolve the conflicts. How can you determine specific requirements for an I&A service, and their relative importance?”
Solution Domain	
Solution	<p>“Specify a set of I&A requirements for a specific I&A domain, and determine the relative importance of each requirement. The solution has two aspects: a requirements process and a common set of generic requirements.</p> <p>A. Requirements Specification and Prioritization Process</p> <p>A system requirements engineer, in conjunction with an enterprise architect, typically perform the requirements process. An important first step is explicitly to define the domain for which you are specifying I&A requirements, such as a specific system or facility. Factors such as enterprise constraints that affect specialization and the importance of requirements also need to be defined. The I&A requirements for the target I&A domain are then specified, using the generic requirements provided below. The final activity is to define the relative importance of the specified requirements.</p> <p>B. Generic Requirements Description</p> <p>The following set of generic requirements responds to the problem and forces described above. The first two represent I&A functional requirements. The remaining requirements represent I&A non.”</p>
Example Resolved	“Samuel the systems engineer and Edward the enterprise architect identify each situation from the museum example above as a separate domain, with a separate set of requirements. The first domain for which they specify I&A requirements is that of the museum employees who access the museum information systems. Table 7.3 shows the requirements, they specified for this domain. The first column contains two sets of information: the generic requirement, followed by the specific requirement for the museum. The second column presents the relevant factor for the requirement in this domain, and the third column discusses the resulting importance of each requirement.”

Known Uses	<p>“The general I&A requirements and the process of specifying I&A requirements described in this pattern represent a consolidation of MITRE Corporation’s experience in working with multiple customers over several decades. The approach is generally used informally by those customers, as opposed to being codified or published. However, some discussions of I&A requirements exist [15].”</p>
Consequences	<ul style="list-style-type: none">• “The pattern fosters explicit definition of I&A domains and a clear connection of requirements to I&A domains. This increases understanding of the full set of domains that are involved in I&A and understanding of the scope of each set of requirements.”• “It facilitates conscious selection of I&A requirements, so that decisions about selecting I&A mechanisms have a clear basis, rather than occurring in a vacuum.”• “It promotes explicit analysis of trade-offs that encourages balancing and prioritizing of conflicting requirements. It helps avoid stronger than necessary I&A, which makes it difficult for valid users, and at the same time it helps to avoid weaker than necessary I&A, which makes it easy for imposters to defeat and therefore provide inadequate protection.”• “It results in documentation of I&A requirements that communicates to all interested parties, and also provides information for security audits.”

Part D: Duyne Design Pattern Template (News Mosaics as an Example)

Design Pattern Name	News Mosaics
Problem Domain	
Intent/Background	“This pattern forms the core that makes news sites useful. Use it separately or in conjunction with any other genere pattern, such as Personal E-Commerce if you sell products or services, Grassroots information sites for expert analysis or content and links to the best sources of information on the web, or community conferences for ways your customers can discuss the topics of the day.
Problem	“Many Readers come to websites to learn about their world through news and history. These sites must deliver the news their readers want, with the depth and breadth of the coverage necessary to engage them and make the historical record available online so that customers can search for older stories. News on the web borrows heavily from that has been learned by offline news organizations, but it also poses new problems unique to the online medium. In terms of time and access, form, and audience, there are key differences between the way news is presented in television, random and print, and the way news is presented on the web.”
Solution Domain	
Solution	“Build a mosaic of news by providing breadth and depth of coverage through a diversity of categories and further refinement through subcategories. Within each category highlight the most important article and lead text, while also providing a breadth of articles that might otherwise be missed. Within each article, provide a high-level summary first, for people who are looking for a quick read, but also provide the more in depth information in the rest of the article. Link together related news items, whether they are articles, radio stories, or video clips. Archive this information in the same place on your servers for historical reference”.
Related Patterns	Ways to Navigate, Clear First Read, Navigation Bar, Headlines and lurbs, Page Template, and organized search results.

Appendix B:

List and Description of Design Problems

Part A:- Design Problems included in Sample-1 and Sample-2 of Experiments and presented in the context of the GoF design pattern collection

Design Problem 1. (Mapping the relationship between request and handler)

“There is a potential variable number of "handler" or "processing element" or "node" objects, and a stream of requests that must be handled. Need to efficiently process the requests without hard-wiring handler relationships and precedence, or request-to-handler mappings.”

Design Problem 2. (Musician problem of control the pitch of sound)

“The Interpreter pattern defines a grammatical representation for a language and an interpreter to interpret the grammar. Musicians are examples of Interpreters. The pitch of a sound and its duration can be represented in musical notation on a staff. This notation provides the language of music. Musicians playing the music from the score are able to reproduce the original pitch and duration of each sound represented.”

Design Problem 3. (To manage the filing system)

“The Iterator provides ways to access elements of an aggregate object sequentially without exposing the underlying structure of the object. Files are aggregate objects. In office settings where access to files is made through the administrative or secretarial staff, the Iterator pattern is demonstrated with the Secretary acting as the Iterator. Several television comedy skits have been developed around the premise of an executive trying to understand the secretary's filing system. To the executive, the filing system is confusing and illogical, but the secretary is able to access files quickly and efficiently.”

Design Problem 4. (To provide a simplified interface for clients)

“A segment of the client community needs a simplified interface to the overall functionality of a complex subsystem.”

Design Problem 5. (To organize the objects internal states)

“The Memento captures and externalizes an object's internal state so that the object can later be restored to that state. This pattern is common among do-it-yourself mechanics repairing drum brakes on their cars. The drums are removed from both sides, exposing both the right and left brakes. Only one side is disassembled and the other serves as a Memento of how the brake parts fit together. Only after the job has been completed on one side is the other side disassembled. When the second side is disassembled, the first side acts as the Memento. On early television sets, a dial was used to change channels. When channel surfing, the viewer was required to move the dial through each

channel position, regardless of whether or not that channel had reception. On modern television sets, a next and previous button are used. When the viewer selects the 'next' button, the next tuned channel will be displayed. Consider watching television in a hotel room in a strange city. When surfing through channels, the channel number is not important, but the programming is. If the programming on one channel is not of interest, the viewer can request the next channel, without knowing its number."

Design Problem 6. (Problem of country president relationship)

"The Singleton pattern ensures that a class has only one instance and provides a global point of access to that instance. It is named after the singleton set, which is defined to be a set containing one element. The office of the President of the United States is a Singleton. The United States constitution specifies the means by which a president is elected, limits the term of office, and defines the order of succession. As a result, there can be at most one active presidents at any given time. Regardless of the personal identity of the active president, the title, 'The President of the United States' is a global point of access that identifies the person in the office."

Design Problem 7. (Problem of arithmetic expressions)

"The Composite composes objects into tree structures and lets clients treat individual objects and compositions uniformly. Although the example is abstract, arithmetic expressions are Composites. An arithmetic expression consists of an operand, an operator (+ - * /), and another operand. The operand can be a number, or another arithmetic expression. Thus, $2 + 3$ and $(2 + 3) + (4 * 6)$ are both valid expressions."

Design Problem 8. (Static binding between an interface and its implementation)

"Hardening of the software arteries has occurred by using subclassing of an abstract base class to provide alternative implementations. This locks in compile-time binding between interface and implementation. The abstraction and implementation cannot be independently extended or composed."

Design Problem 9. (Problem of lazy initialization)

"The application needs one, and only one, instance of an object. Additionally, lazy initialization and global access are necessary."

Design Problem 10. (On demand instantiation of resource hungry objects)

"You need to support resource-hungry objects, and you do not want to instantiate such objects unless and until they are actually requested by the client"

Design Problem 11. (To manage automobile systems)

"This pattern is found in the sheet metal stamping equipment used in the manufacture of Japanese automobiles. The stamping equipment is an Abstract Factory, which creates auto body parts. The same machinery is used to stamp right hand doors, left hand doors, right front fenders, left front fenders, hoods, etc. for different models of cars."

Through the use of rollers to change the stamping dies, the concrete classes produced by the machinery can be changed within three minutes.“

Design Problem 12. (Architectural modeling for applications)

“A framework needs to standardize the architectural model for a range of applications, but allow for individual applications to define their own domain objects and provide for their instantiation.“

Design Problem 13. (Adding new functionality of an object at runtime)

“You want to add behavior or state to individual objects at run-time. Inheritance is not feasible because it is static and applies to an entire class.“

Design Problem 14. (Development of language irrelevant applications)

“A class of problems occurs repeatedly in a well-defined and well-understood domain. If the domain were characterized with a "language", then problems could be easily solved with an interpretation "engine".“

Design Problem 15. (Abstraction between data structure and algorithms)

“Need to "abstract" the traversal of wildly different data structures so that algorithms can be defined that are capable of interfacing with each transparently. “

Design Problem 17. (Implementation of rollback issue in object's state)

“Need to restore an object back to its previous state (e.g. "undo" or "rollback" operations).“

Design Problem 18. (Controlling lights)

“The Bridge pattern decouples an abstraction from its implementation, so that the two can vary independently. A household switch controlling lights, ceiling fans, etc. is an example of the Bridge. The purpose of the switch is to turn a device on or off. The actual switch can be implemented as a pull chain, simple two position switch, or a variety of dimmer switches. “

Design Problem 19. (Customer's ordering systems)

“The Facade defines a unified, higher level interface to a subsystem that makes it easier to use. Consumers encounter a Facade when ordering from a catalog. The consumer calls one number and speaks with a customer service representative. The customer service representative acts as a Facade, providing an interface to the order fulfillment department, the billing department, and the shipping department.“

Design Problem 20. (Bidder and auction problem)

“The Observer defines a one-to-many relationship so that when one object changes state, the others are notified and updated automatically. Some auctions demonstrate this pattern. Each bidder possesses a numbered paddle that is

used to indicate a bid. The auctioneer starts the bidding, and "observes" when a paddle is raised to accept the bid. The acceptance of the bid changes the bid price which is broadcast to all of the bidders in the form of a new bid."

Design Problem 22. (Vending machine problem)

"The State pattern allows an object to change its behavior when its internal state changes. This pattern can be observed in a vending machine. Vending machines have states based on the inventory, amount of currency deposited, the ability to make a change, the item selected, etc. When currency is deposited and a selection is made, a vending machine will either deliver a product and no change, deliver a product and change, deliver no product due to insufficient currency on deposit, or deliver no product due to inventory depletion."

Design Problem 23. (Air traffic to control several airplanes)

"Design the communications of one plane to the approach of an airport. When a plane is in approach of the airport, it must announce to all the other planes which are around that it intends to be posed, and await their confirmation with all before carrying out the operation. It is the control tower of the airport, which guarantees the regulation of the air traffic, by making sure that there is no trajectory conflict or destination between several planes. Besides the class diagram, represent by a collaboration (diagram of collaboration or diagram of objects and sequence) the landing of a plane between two wanting to land and one wanting to take off."

Design Problem 24. (Children's meals at restaurants)

"The Builder pattern separates the construction of a complex object from its representation so that the same construction process can create different representations. This pattern is used by fast food restaurants to construct children's meals. Children's meals typically consist of a main item, a side item, a drink, and a toy (e.g., a hamburger, fries, Coke, and toy dinosaur). Note that there can be variation in the content of the children's meal, but the construction process is the same. Whether a customer order a hamburger, cheeseburger, or chicken, the process is the same. The employee at the counter directs the crew to assemble a main item, side item, and toys. These items are then placed in a bag. The drink is placed in a cup and remains outside of the bag. This same process is used at competing restaurants."

Design Problem 25. (Bank transactions)

"The Proxy provides a surrogate or placeholder to provide access to an object. A check or bank draft is a proxy for funds in an account. A check can be used in place of cash for making purchases and ultimately controls access to cash in the issuer's account."

Design Problem 26. (ATM handling)

"The Chain of Responsibility pattern avoids coupling the sender of a request to the receiver by giving more than one object a chance to handle the request. ATM use the Chain of Responsibility in money giving mechanism."

Design Problem 27. (Controlling payment systems for waiters)

“The Command pattern allows requests to be encapsulated as objects, thereby allowing clients to be parameterized with different requests. The "check" at a diner is an example of a Command pattern. The waiter or waitress takes an order or command from a customer and encapsulates that order by writing it on the check. The order is then queued for a short order cook. Note that the pad of ‘checks’ used by each waiter is not dependent on the menu, and therefore they can support commands to cook many different items.”

Design Problem 28. (To decorate the Christmas tree)

“The Decorator attaches additional responsibilities to an object dynamically. The ornaments that are added to pine or fir trees are examples of Decorators. Lights, garland, candy canes, glass ornaments, etc., can be added to a tree to give it a festive look. The ornaments do not change the tree itself, which is recognizable as a Christmas tree regardless of particular ornaments used. As an example of additional functionality, the addition of lights allows one to ‘light up’ a Christmas tree.”

Design Problem 29. (To construct window GUI interface)

“Design a help manager of a Java application. A help manager allows the show of a help message depending on the objects on which a client has clicked. For example, the "?", sometimes located near the contextual menu in a Windows dialog box, allows the show of the help of the button or the area where we click. If the button on which one clicks does not contain help, it is the area containing which displays its help, and so on. If no object contains help, with final, the manager displays "Not help available for this area". Instantiate your class diagram in a sequence diagram of on the example of a printing window. This window (JDialog) consists in an explanatory text (JLabel), and in a container (JPanel). This last contains a Print button (JButton) and a Cancel button (JButton). The Print button contains help ‘Launches the impression of the document’. The Cancel button, the text as well as the window do not contain help. Lastly, the container contains help "Click on one of the buttons". In the sequence diagram, reveal the scénarios: ‘The user asks for the help of the Print button’, ‘the user asks for the help of the Cancel button', and ‘ the user asks for the help of the text.’”

Design Problem 30. (To design a calculator)

“Design a tutorial to learn how to program a calculator. This calculator executes the four basic arithmetic operations. The goal of this tutorial is to make it possible to take a set of operations to be executed sequentially. The tutorial presents a button by an arithmetic operation, and two input fields for the operands. After each click on a button of an operation, the user has then the choice to start again or execute the suite of operations to obtain the result. It is probable that this teachware evolves in order to make it possible to the user to remove the last operation of the list and to take into account the operation of modulo.”

Part B:- Design Problems included in Sample-1 and Sample-2 of Experiments and presented in the context of the Douglass design pattern collection

Design Problem 1. (Dynamic memory allocation)

“The prototypical candidate system for the Pooled Allocation Pattern is a system that cannot deal with the issues of dynamic memory allocation, but it is too complex to permit a static allocation of all objects. Typically, a number of similar, typically small, objects, such as events, messages, or data objects, may need to be created and destroyed, but are not needed a priori by any particular clients for the entire run-time of the system.”

Design Problem 2. (Asymmetric distribution architectures)

“In addition to the problems addressed by the Proxy Pattern (such as communication infrastructure transparency), a limitation of most of the distribution patterns is that they require a priori knowledge of the location of the servers. This limits their use to asymmetric distribution architectures. Ideally, the solution should provide a means that can locate and then invoke services at the request of the client, including subscription to published data.”

Design Problem 3. (CPU utilization in case of multiprogramming)

“How to protect the programs from one another and the kernel from them all and how to handle relocation in case, when more than one program reside in main memory, waiting either for I/O (Input/Output) or CPU resource, for better CPU utilization.”

Design Problem 4. (Management of airplanes using control tower)

“Design the communications of one plane to the approach of an airport. When a plane is in approach of the airport, it must announce to all the other planes which are around that it intends to be posed, and await their confirmation with all before carrying out the operation. It is the control tower of the airport, which guarantees the regulation of the air traffic, by making sure that there is no trajectory conflict or destination between several planes. Besides the class diagram, represent by a collaboration (diagram of collaboration or diagram of objects and sequence) the landing of a plane between two wanting to land and one wanting to take off.”

Design Problem 5. (To manage unrelated operations in aggregate structure)

“Many distinct and unrelated operations need to be performed on node objects in a heterogeneous aggregate structure. You want to avoid ‘polluting’ the node classes with these operations. And, you don't want to have to query the type of each node and cast the pointer to the correct type before performing the desired operation.”

Design Problem 6. (Multithreading)

“In most multithreaded systems, threads must synchronize and share information with others. Two primary things must be accomplished. First, the tasks must synchronize to permit sharing of the information. Second, the information must be shared in such a way that there is no chance of corruption or race conditions.”

Design Problem 7. (Event handling)

“In a system in which certain events are highly urgent, such as those that can occur at high frequencies or those in which the response must be as fast as possible, a means is needed to quickly identify and respond to those events. This can occur in many different kinds of systems from simple to complex.”

Design Problem 8. (Faults avoideness)

“The Heterogeneous Redundancy Pattern provides protection against both kinds of faults-systematic errors as well as random failures. Assuming that the design includes independence of faults, the pattern provides single fault safety in the same way as the Homogeneous Redundancy Pattern-that is, when the primary channel detects a fault, the secondary channel takes over.”

Design Problem 9. (Priority inversions)

“The unbounded priority inversion problem is discussed in the chapter introduction in some detail. The problem addressed by this pattern is to bound the maximum amount of priority inversion.”

Design Problem 10. (Threads management)

“The problem this pattern addresses is the need for a timely synchronization or data exchange between threads. In such cases, it may not be possible to wait for an asynchronous rendezvous. A synchronous rendezvous can be made more timely, but this must be done carefully to avoid data corruption and erroneous computation.”

Design Problem 11. (Task scheduling)

“The Round Robin Pattern addresses the issue of moving an entire set of tasks forward at more or less equal rates, particularly tasks that do not complete within a single scheduling cycle.”

Design Problem 12. (Dynamic memory allocation)

“The prototypical candidate system for the Pooled Allocation Pattern is a system that cannot deal with the issues of dynamic memory allocation, but it is too complex to permit a static allocation of all objects. Typically, a number of similar, typically small, objects, such as events, messages, or data objects, may need to be created and destroyed, but are not needed a priori by any particular clients for the entire run-time of the system.”

Design Problem 13. (Memory fragmentation)

“One of the key problems with dynamic memory allocation is memory fragmentation. Memory fragmentation is the random inter-mixing of free and allocated memory in the heap. For memory fragmentation to occur, the following conditions must be met. The order of memory allocation is unrelated to the order in which it is released. Memory is allocated in various sizes from the heap.”

Design Problem 14. (Garbage collection)

“The Garbage Collection Pattern addresses the problem of how we can make sure we won't have any memory leaks. Many high-availability or high-reliability systems must function for long periods of time without being periodically shut down. Since memory leaks lead to unstable behavior, it may be necessary to completely avoid them in such systems. Furthermore, reference counting Smart Pointers (see Smart Pointer Pattern, earlier in this chapter) has the disadvantages that they require programmer discipline to use correctly and cannot be used when there are cyclic object references.”

Design Problem 15. (Resource sharing)

“The main problem addressed by the Critical Section Pattern is how to robustly share resources that may have, at most, a single owner at any given time.”

Design Problem 16. (Deadlock avoidence)

“The problem of deadlock is such a serious one in highly reliable computing that many systems designs in specific mechanisms to detect it or avoid it. As previously discussed, deadlock occurs when a task is waiting on a condition that can never, in principle, be satisfied. There are four conditions that must be true for deadlock to occur, and it is sufficient to deny the existence of any one of these. The Simultaneous Locking Pattern breaks condition 2, not allowing any task to lock resources while waiting for other resources to be free.”

Design Problem 17. (Data sharing in case of multiprocessing)

“Many systems have to share data between multiple processors—this is the essence of distribution, after all. In some cases, the access to the data may persist for a long period of time, and the amount of data shared may be large. In such cases, sending messages may be an inefficient method for sharing such information. Multiple computers may need to update this "global" data, such as in a shared database, or they may need to only read it, as is the case with executable code that may run on many processors or configuration tables. A means by which such data may be effectively shared is needed.”

Design Problem 18. (Client server interaction)

“The programming model used to invoke services locally is very well understood: dereference an association (most commonly implemented as a pointer or reference) and invoke the service. This is done in a synchronous fashion—the caller blocking until the server completes the request and returns whatever values were requested. What is needed is a means to do the same thing even when the client and server are not collocated.”

Design Problem 19. (Data sharing in client server interaction)

“Many systems need to share many different data among a mixture of servers and clients, some of whom might not be known when the client or data is designed. This pattern solves the problem by providing a central storage facility into which data that is to be shared may be plugged along with metadata that describes its contents.”

Design Problem 20. (Address space issues in multiprocessing)

“The design of modern embedded systems must often be deployed across multiple address spaces, such as different processors. Often, such details are subject to change during the design process or, even worse, during the implementation of the system. It is problematic to "hard-code" the knowledge that a server may be remote because this may change many times as the design progresses. Further, the clients and servers may be redeployed in other physical architectures and using different communications media. If the clients are intimately aware of these design details, then porting the clients to the new platforms is more difficult.”

Part C:- Design Problems included in Sample-1 and Sample-2 of Experiments and presented in the context of the Security design pattern collection

Design Problem 1. (Requirements management for security accounting systems)

“Security accounting is an activity that takes in the detectable data from an event and provides some security-relevant information about that event to a human. A basic accounting sequence is completed when security-relevant information associated with an event of interest is provided to the accounting user. You need a clear set of requirements to ensure that the strategy employed in a security accounting system actually satisfies the needs of the organization or system. Requirements for security accounting often conflict with each other, and trade-offs between them are often necessary. You need to prioritize these requirements to determine under what circumstances you should put more emphasis on one requirement over another.”

Design Problem 2. (Resource management in systems)

“We need to have a way to control access to resources, including information. The first step is to declare who is authorized to access resources in specific ways. Otherwise, any active entity (user, process) could access any resource and we could have confidentiality and integrity problems. How do we describe who is authorized to access specific resources in a system?”

Design Problem 3. (Security problem during data sharing)

“Files may contain valuable information and access to them must be controlled carefully. In several recent attacks, hackers obtained lists of credit card numbers by accessing customer data illegally. Because files need to be shared, security becomes harder to enforce.”

Design Problem 4. (Privileges policy for role based access control systems)

“The ROLE-BASED ACCESS CONTROL model is used now in many systems. However, the different component frameworks (.NET, J2EE) provide support only to define roles and to write authorization rules, and do not say anything about where the rights come from. It is not easy for system designers or for administrators to define the required roles and their corresponding rights. How can we assign appropriate rights to the roles when we want to implement a least privilege policy?”

Design Problem 5. (Tradeoff between audit requirements)

“Audit is a security service that scrutinizes logs, audit trails or other captured information and attempts to discern more detailed information about an event. It analyzes the event information for any indication of security violations. You need a clear set of requirements to ensure that the audit strategy employed actually satisfies the needs of the organization or system. Requirements for audit often conflict with each other, and tradeoffs among them are often necessary. The conflict stated above in the example is that the need to provide an audit trail must be balanced with resource and privacy constraints. What types of information are appropriate or required for an audit system to analyze?”

Design Problem 6. (Process management in case of accessing memory)

“Processes must be controlled when accessing memory, otherwise they could overwrite each other’s memory areas or gain access to private information. While relatively small amounts of data can be directly compromised, illegal access to system areas could allow a process to force a higher execution privilege level and thus access files and other resources.”

Design Problem 7. (Implementation of identified and authenticate services)

“Requirements for I&A often conflict with each other, and tradeoffs among them are often necessary. The conflict stated in the Example section is that strength of protection with I&A tends to conflict with ease of use. I&A comprises both associating an identifier with an actor (identification) and verifying that the association is correct (authentication). I&A is a security service whose results are often used by other security services, including access control and accounting.”

Design Problem 8. (Load balancing in case server and vendor’s solution)

“You want to implement your Web site using different servers, or use different vendors solutions for your Web site. How do you provide everything in a consistent Web application space without showing your server topology to users? How do you gain flexibility in network topology, for example by adding or removing servers without surprising users? How do you provide failover switching or load balancing if an application server gets overloaded?”

Design Problem 9. (Security issues in Multi user systems)

“How do you provide a single sign on for several Web applications or services? The solution to this problem must resolve the following forces: you want a single user identity for all applications, even when existing Web applications already carry their own user data base. You do not want users to have to provide their password for each application separately, depending on your security policy. You might want to force users to identify themselves several times, to avoid misuse of a user’s session that is left alone for some time.”

Design Problem 10. (Payload attacks)

“Some of the hosts on other networks may try to attack the local network through their IP-level payloads. These payloads may include viruses or application-specific attacks. How can we identify and block those hosts?”

Design Problem 11. (Management of harmful events in enterprize)

“Enterprise assets face a barrage of attacks and hazardous events from all directions. Without effectively acknowledging the origins and frequency of these threats, an enterprise may never recognize the extent to which their assets are at risk. How can an enterprise identify harmful events and determine the likelihood of their occurrence?”

Design Problem 12. (Sharing of resources for multiple users)

“Systems shared by multiple users, either via terminals or via a network, have become commonplace. Instead of single-user non-networked systems like the-now almost extinct-DOS PCs, shared or networked systems need to account for a user’s actions and ensure users only have access to areas for which they have privileges.”

Design Problem 13. (System integrity in case of external client)

“Whenever a system is used by an external client such as a user, the system’s integrity is in danger. Often, such systems require some security property, like protection from misuse or damage. One means is to check every interaction with an external client to determine whether it is authorized. When the system has a non-trivial inner structure and consists of multiple parts or subsystems, an external interaction of the system can result in many different interactions of the client with the individual parts of the system.”

Design Problem 14. (Log-in mechanism for authorization and authentication of users)

“In addition, you want to consider the change of requirements for identification and authorization (I&A) that might occur over time, either because you need to address new threats, or because you learn from its use. One example for handling that situation is the development of a protected system in which a developer will use a dummy I&A implementation to test the system, without the hassle of logging into the system for every test. Later on, the deployed system needs to be protected by a log-in mechanism that authenticates and authorizes its users. How can you provide an architecture that allows you to effectively protect system access while still being able to tune I&A to evolving needs without impact to the system you protect?”

Design Problem 15. (Event managements for audit trail and login)

“An organization needs to observe events and to revisit data related to those events to help achieve security properties in a system or domain, and to understand when and how security properties have been compromised. Audit trails and logging (AT&L) is a security service that automates the capturing of information about events and activities that occur within the organization. Audit trails are a series of records about system events or user activities. Audit trails can be used to reconstruct events, determine who is responsible for events, what malicious or unwanted activities have occurred, and analysis of any problems. Logs are individual trails of information that may be combined into an audit trail.”

Design Problem 16. (Participant engagement in the context of non-repudiation systems)

“Non-repudiation is a security service that captures and maintains evidence so that the participants of a transaction or interaction cannot deny having participated in that activity. The need is to identify the common requirements that drive the design of this service. The model in the figure places non-repudiation in the context of the identity of the participants, in terms of the activity that it collects, and the facts and evidence that it provides. This ensures that the participants cannot deny having engaged in the activity. Non-repudiation needs information about the event that will disallow the participants from denying their participation. If the participants are allowed to deny their involvement

in the activity, then the integrity of the activity will be jeopardized and other participants may suffer negative consequences. For example, if a purchaser receives a book that they ordered from Amazon.com, and then denies receiving it, Amazon may need to send another copy of the book, which is a financial loss to them.”

Design Problem 17. (Security integration in the business models)

“To fully integrate security into the business model, business planners need to identify the security services needed to protect each category of enterprise asset. Selection of security services will need to balance the resources the museum is willing to allocate in order to address security approaches appropriately. At the business level, planners provide direction about how much emphasis to focus on preventing security incidents, detecting incidents after the fact, and the level of focus for responding to security incidents. Some services, such as access control, emphasize a prevention approach. Other services, such as accounting, emphasize detection and response. Still others, such as identification and authentication, support both prevention and detection. How do you select and integrate security services across the organization to support security properties using preferred security approaches?”

Design Problem 18. (Concurrent processing)

“A user executes an application composed of several concurrent processes. Processes are usually created through system calls to the operating system. A process that needs to create a new process gets the operating system to create a child process that is given access to some resources. A computing system uses many processes or threads. Processes need to be created according to application needs, and the operating system itself is composed of processes. If processes are not controlled, they can interfere with each other and access data illegally. Their rights for resources should be carefully defined according to appropriate policies, for example ‘need-to-know.’”

Design Problem 19. (Dynamic management of resource and objects)

“In a computing environment, executing applications need to create objects in their work. Some objects are created at program initialization, while others are created dynamically during execution. The access rights of processes with respect to objects must be defined when these objects are created, or there may be opportunities for the processes to misuse them. Applications also need resources such as I/O devices and others that may come from resource pools: when these resources are allocated, the application must be given rights to them.”

Design Problem 20. (Process authorizations)

“Restricting a process to a specific set of resources is a basic step towards controlling malicious behavior. Otherwise, unauthorized processes could destroy or modify information in files or databases, with obvious results, or could interfere with the execution of other processes.”

Design Problem 21. (Vulnerable management)

“Even if you install a simple PACKET FILTER FIREWALL your Web server can remain vulnerable to attacks that exploit weaknesses in its protocol implementation. How can you protect your Web server infrastructure in the light of its potential vulnerability to attacks using its protocol?”

Design Problem 22. (Handling system's functionality)

“Presenting all potentially-available functionality to users not privileged to use it can represent a security problem. You might want users that don't have access to a functionality to not even be aware that it exists. A system utilized by people with varying skills and access rights can be very hard to use if every possible option is presented to every user, as is proposed by FULL ACCESS WITH ERRORS. It is much more user friendly if a user can only see or select the options actually available. How can you present a system's functionality and ensure that users can only access those parts or data of a system to which they are entitled?”

Design Problem 23. (Blocking malicious sites)

“PACKET FILTER FIREWALL only inspects the network addresses when deciding whether to allow access for a request. We can only block supposedly malicious sites. It is hard to know about all of those sites, and we need further defenses. Also, how do we protect our network from potential attacks that might be embedded within the data segment of the packets?”

Design Problem 24. (Customer validation in case of partner organizations)

“We want to provide a system that allows us to collaborate with an organization either as a customer or as a business partner. How can we validate the identity of an organization so that we can be sure they are who we think they are, and they can be sure that we are who we say we are?”

Design Problem 25. (Assigning rights to users)

“When designing the user interface for a system with partial access restrictions, you face the challenge of whether to present functionality that a user might not be able to access within their current role or set of access rights, and how to do so. To complicate the issue, you might not know in advance what possible combinations of access rights will be used. This problem generalizes to any interface you design whenever there are multiple modes of use, such as different access rights. How do you present available functionality that might be partially inaccessible?”

Design Problem 26. (Enterprise protection for its systems in case of external partners)

“When an enterprise engages in a business relationship, it typically exchanges information and allows users and/or applications to access privileged resources. Not only can there be a risk of theft or manipulation of data, but also risk of unauthorized access to resources by another organization. Furthermore, you may trust the partner with whom you entered into a relationship, but can you trust their contractors, application vendors, networks, or firewall configuration? A breach in their network may lead to a breach in your own. How can an enterprise protect its systems and data while communicating with external partners?”

Design Problem 27. (Tradeoff between authentication techniques)

“I&A is a common need for systems and enterprises. Multiple techniques exist for achieving I&A. Different techniques emphasize different types of authenticators. No one technique is the best in all situations. Trade-offs and

weighting are typically necessary, because in general the techniques have differing and often complementary strengths and weaknesses.”

Design Problem 28. (Unauthorized attack)

“How do we ensure that sensitive data gathered and stored by our system are protected from unauthorized access?”

Design Problem 29. (Information leakage)

“How can you control access in an environment with sensitive documents so as to prevent leakage of information?”

Design Problem 30. (Process management to access to common memory)

“Processes must be controlled when accessing memory, otherwise they could overwrite each other’s memory areas or gain access to private information. While relatively small amounts of data can be directly compromised, illegal access to system areas could allow a process to force a higher execution privilege level and thus access files and other resources.”

Part D:- Design Problems included in Sample-1 and Sample-2 of Experiments and presented in the context of the Dyne design pattern collection

Design Problem-1. (Personalize benefits of online ordering system)

“Customers appreciate the convenience of ordering online, but if a site is cumbersome, is veild about its pricing and policies, or does not seem to provide a personal benefit, they leave.”

Design Problem-2. (Managing ideas of online community members)

“Community members want to share ideas, views, and opinions with other like minded individuals, whether they live across the street or across the planet. However, a host of issues must be resolved, such as community usage policies, moderation of forums, anonymity, archives, interaction, trust sociability, growth and sustainability. The Challenge is to strike a balance within the online community.”

Design Problem-3. (Setting multiple navigation ways on the web site)

“Customers navigate websites in many ways. If any of the key navigation tools are hard to find or missing, visitors will find the site tedious to use. Customers navigate through a site to gather information and to accomplish goals. They may look for information, activities, or products in any number of ways.”

Design Problem-4. (Organization of content on web sites)

“Browsing content on a site can be difficult if the information is not organized, or if there are no clear and consistent cues for finding content and returning to it later. Customers group and organized information in many ways to structure the content.”

Design Problem-5. (Targeting certain on the home page of websites)

“Homepages are the portal through which most visitors pass. They must seduce visitors while simultaneously balancing many issues, including branding, navigation, content, and the ability to download quickly.”

Design Problem-6. (Consistency in web page content and navigation)

“A site that is not consistent from page to page is difficult for customers to navigate and hard for site managers to maintain. However, it is challenging to design web pages to consistent because not all pages are the same, and many will need some way to be updated.”

Design Problem-7. (Adjustment of large volumes of content)

“Without a good system, publishing and managing large volumes of content i time-consuming and error prone process. When customers visit your site and find that the homepage is the same as it was a week ago or may be a month ago, they might say ‘Nothing new here’ and leave.”

Design Problem-8. (Security threat for identification of customers during advertizement)

“The Brand is more than an image. Customers need to know where they are and whether they can trust that place to provide something important and unique. The identity created on a site, through advertising and in interactions with people at the company, persist even customers leave their computers, close their magazines, or hang up their phones.”

Design Problem-9. (Communication with customers via web sites)

“Companies need a way of maintaining contact with customers who are interested in what their web site has to offer. E-mail subscriptions are an inexpensive way of maintaining a connection with customers and that are especially effective if customers are interested in what a web site can offer.”

Design Problem-10. (The exit criteria for the online shopping process)

“An e-commerce shopping experience will not be enjoyable, or worse, a purchase might not be completed., if the checkout process is cumbersome, confusing or error prone. By the time your customers get to the checkout process on your site, they may be impatient to finish the order and get out of his door.”

Design Problem-11. (To enhance trust of customers on the sellers)

“When shopping, customers want to see product details to help inform their buying decision. They must also trust a seller before deciding to make a purchase. May sites do not provide enough in depth information about their products or they project an untrustworthy image.”

Design Problem-12. (Recommendation of value-added products)

“Customers find value when sites identify specific products as recommended t features. Otherwise product lists can appear bland an tedious. Having tens, hundred, thousands, even millions of products on a web site give customers a wide selection of choices.”

Design Problem-13. (Recommendation of better products)

“When Choosing a product in stores, people appreciate hearing about related products that are complementary to or better than the products they have chosen, Doing the same thing online requires prudence and planning. People often come to a site looking for something.”

Design Problem-14. (Prioritized tasks according to customers needs)

“Customers often need to complete highly specific tasks on web sites, but pages with tangential links and many questions can prevent them from carrying out these tasks successfully. People enjoy completing the tasks they start.
“

Design Problem-15. (User profile management through the web site)

“Customers need to see and manage the information of a web site keeps about them. Provide a single page that gathers all the account information in one place. Provide a task based organization scheme to let customers modify their information.”

Design Problem-16. (Setting layout of website in existence of certain elements)

“It is difficult to organize the many competing elements of a web page in a cohesive manner without creating clutter and overwhelming the reader. Grid layout is a technique for the graphic design used for organizing page layouts from newspapers, magazines, and other documents.”

Design Problem-17. (Improve the customer’s navigation process on web sites)

“Customers often miss navigation elements and content if they have to scroll down to see them. The term above the fold comes from the newspaper business because the newspaper is folded in half when they are sold on the street.”

Design Problem-18. (Quick navigation of customers on websites)

“Customers sometimes want to quickly jump from one location to another, but search pages are too complex for this. When entering your site, customers sometimes know exactly what they are looking for. A quick and simple search function on the home page is critical to win these visitors confidences.”

Design Problem-19. (How to structure the elements of a website)

“Customers need a structured, organized way of reaching the most important parts of your web site that is easy to understand and use. Large scale web site needs a clear and systematic scheme to make it easy for visitors to navigate them.”

Design Problem-20. (How to organize content in the form files and folders)

“Sites need to let customers navigate through categories to content and give them feedback on where they are. But to make tab rows work well requires including specific details in the visuals. Tab rows are cues that people find familiar. They are reminiscent of tabs on file folders at the office and in school.”

Design Problem 21. (To manage the picture and videos with different grid layouts)

“When you view someone’s YouTube channel, you can choose to see either a Thumbnail Grid, or a list beside a video player (the default). (Both options are shown in this screenshot.) Clicking a thumbnail brings you to the page for that video, where detailed information and discussion are shown. Visitors can browse by looking at playlists, the latest videos added, the most-viewed videos, and the top-rated videos; a search box is also provided, as it is everywhere.”

Design Problem 22. (How to manage the customer's navigation)

“The Import Wizard is an old-school, traditional application wizard with Back/Next buttons, branching, and no sequence map. But it works. Each screen lets you focus on the step at hand, without worrying about what comes next.”

Design Problem 23. (How to manage the list of emails)

“Many email clients use this pattern to show a list of email messages next to the currently selected message. Such listings benefit from being nearly as wide as the whole window, so it makes sense to put the listing on top of the second panel, not to its left.”

Design Problem 24. (How to manage the menus on the website)

“The Fat Menus on the Starbucks website are very well designed. Each menu is a different height but the same width, and follows a strict common page grid (they're all laid out the same way). The style blends in with the site, and the generous white space makes it easy to read. Ads are worked into the design, but not obnoxiously. The non-rectangular shape adds a polished look.”

Design Problem 25. (How to manage the customer's navigation)

“REI's website demonstrates the difference between task-oriented top-of-page global navigation and an effective Sitemap Footer. Shopping, learning, and travel dominate the header, as they should—these are what most site visitors come for. The footer handles secondary tasks that are nevertheless important: “about” information, customer support, membership, and so on.”

Appendix C: Cosine Similarity and Application of First Similarity Option

**Part A:- Cosine Similarity (CS) measures values for 10 design problems (Appendix B) of Sample 1
Related to GoF Design Pattern Collection**

Table C-1. Cosine similarity results for Design Problem 1.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Abstract Factory	0.03
Command	0.22
Composite	0.14
Factory Method	0.05
Mediator	0.31
Singleton	0.10
State	0.26
Chain of Responsibility	<u>0.54</u>

Table C-2. Cosine similarity results for Design Problem 2.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Bridge	0.03
Interpreter	0.38
Memento	0.36
Iterator	0.23
Strategy	0.31
Builder	0.12

Table C-3. Cosine similarity results for Design Problem 3.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Chain of Responsibility	0.30
Command	0.18
Interpreter	0.43
Iterator	<u>0.46</u>
Abstract Factory	0.35
Memento	0.38
Adapter	0.15
Observer	0.24

Table C-4. Cosine similarity results for Design Problem 4.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Proxy	0.12
Fascad	<u>0.48</u>
Composite	0.13
Iterator	0.26
Abstract Factory	0.34

Table C-5. Cosine similarity results for Design Problem 5.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Template	0.31
Command	0.16
Mediator	0.35
Memento	<u>0.58</u>
Adapter	0.05
Observer	0.21

Table C-6. Cosine similarity results for Design Problem 6.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Singleton	<u>0.60</u>
Abstract Factory	0.53
Proxy	0.14
Factory Method	0.51
Composite	0.25

Table C-7. Cosine similarity results for Design Problem 7.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Decorator	0.13
Composite	<u>0.58</u>
Flyweight	0.33
Command	0.16
Fascad	0.25
Observer	0.20

Table C-8. Cosine similarity results for Design Problem 8.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Observer	0.33
Strategy	0.38
Bridge	0.52
Proxy	<u>0.60</u>

Table C-9. Cosine similarity results for Design Problem 9.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Builder	0.23
Prototype	0.48
Singleton	<u>0.63</u>
Composite	0.16
Command	0.23

Table C-10. Cosine similarity results for Design Problem 10.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Proxy	<u>0.61</u>
Builder	0.28
Flyweight	0.64
Bridge	0.26
Decorator	0.35

**Part B:- Cosine Similarity (CS) measures values for 10 design problems (Appndix B) of Sample 1
Related to Douglass Design Pattern Collection**

Table C-11. Cosine similarity results for Design Problem 1.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Critical Section	0.23
Garbage Collection	0.22
Pool Allocation	<u>0.57</u>
Garbage Compactor	0.15
Data Bus	0.54

Table C-12. Cosine similarity results for Design Problem 2.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Broker	<u>0.43</u>
Pool Allocation	0.22
Static Allocation	0.34
Static Priority	0.20
Proxy	0.17
SharedMemory	0.14

Table C-13. Cosine similarity results for Design Problem 3.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Garbage Collection	0.53
Critical Section	0.23
Order Locking	<u>0.64</u>
Static Allocation	0.29
Fixed Size Buffer	0.14

Table C-14. Cosine similarity results for Design Problem 4.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Data Bus	0.13
Simultaneous Locking	0.30
Pool Allocation	0.14
Critical Section	<u>0.54</u>
Proxy	0.13
Static Allocation	0.40

Table C-15. Cosine similarity results for Design Problem 5.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Safety Executive	<u>0.43</u>
Sanity Check	0.33
Data Bus	0.28
Proxy	0.34

Table C-16. Cosine similarity results for Design Problem 6.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Data Bus	0.16
Remote Method Call	0.23
Round Robin	0.41
Rendezvous	0.29
Message Queuing	<u>0.48</u>

Table C-17. Cosine similarity results for Design Problem 7.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Static Priority	0.23
Interrupt	<u>0.53</u>
Broker	0.21
Gaurded Call	0.31

Table C-18. Cosine similarity results for Design Problem 8.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Safety Executive	0.46
Sanity Check	0.13
Watchdog	0.05
Heterogenous redundancy	<u>0.52</u>
Highest Locker	0.12

Table C-19. Cosine similarity results for Design Problem 9.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Priority inheritance	<u>0.45</u>
Order Locking	0.31
Simultaneous Locking	0.14
Smart Pointer	0.24
Static Allocation	0.32
Garbage Collection	0.25
Priority Ceiling	0.13

Table C-20. Cosine similarity results for Design Problem 10.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Broker	0.42
Proxy	<u>0.53</u>
Pool Allocation	0.22
Static Allocation	0.25

**Part C:- Cosine Similarity (CS) measures values for 12 design problems (Appendix B) of Sample 1
Related to Security Design Pattern Collection**

Table C-21. Cosine similarity results for Design Problem 1.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Audit requirements	0.35
Role Right Definition	0.22
Security Accounting Requirments	<u>0.51</u>
Asset Valuation	0.31

Table C-22. Cosine similarity results for Design Problem 2.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Non-Repudiation Requirments	<u>0.35</u>
I&A requirements	0.22
Role Based Access Control	0.32
Security Accounting Requirments	0.29
File Authorization	0.15

Table C-23. Cosine similarity results for Design Problem 3.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Front Door	0.14
Execution Domain	0.20
Role Right Definition	0.34
Authenticator	0.31
File Authorization	<u>0.45</u>

Table C-24. Cosine similarity results for Design Problem 4.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Reference Monitoring	0.18
Risk Determination	0.21
Role Right Definition	0.34
Role Based Access Control	<u>0.52</u>
Authenticator	0.31
Demilitarized Zone	0.35

Table C-25. Cosine similarity results for Design Problem 5.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Statetfull Firewall	0.24
Packet Filter Firewall	<u>0.45</u>
Audit requirements	0.39
Authenticator	0.11
Role Right Definition	0.15

Table C-26. Cosine similarity results for Design Problem 6.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Controlled Virtual Address Space	<u>0.55</u>
Controlled Process Creator	0.34
Security Accounting Requirments	0.23
Front Door	0.17

Table C-27. Cosine similarity results for Design Problem 7.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Statetfull Firewall	0.20
Packet Filter Firewall	<u>0.35</u>
I&A Requirements	0.31
Authenticator	0.24
File Authorization	0.13
Role Right Definition	0.23
Reference Monitoring	0.16

Table C-28. Cosine similarity results for Design Problem 8.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Known Partners	0.32
Execution Domain	0.20
Security Accounting Requirments	0.23
Asset Valuation	0.21
Integration Revrse Proxy	<u>0.47</u>

Table C-29. Cosine similarity results for Design Problem 9.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Front Door	<u>0.49</u>
Known Partners	0.36
Check Points	0.27
Security Session	0.18

Table C-30. Cosine similarity results for Design Problem 10.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Proxy Based Firewall	0.25
Packet Filter Firewall	0.33
Audit requirements	<u>0.37</u>
Role Right Definition	0.21
I&A Requirements	0.31

Table C-31. Cosine similarity results for Design Problem 11.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Asset Valuation	0.31
Execution Domain	0.22
Threat Assessment	<u>0.58</u>
Vulnerability Assessment	0.31
Integration Revrse Proxy	0.17

Table C-32. Cosine similarity results for Design Problem 12.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Front door	0.23
Limited Access	0.29
Security Session	0.43
Audit requirements	0.31
Role Right Definition	0.23
Security Accounting Requirments	<u>0.47</u>
Single Access Point	0.20

**Part D:- Cosine Similarity (CS) measures values for 15 design problems (Appendix A) of Sample 1
Related to Duyne Design Pattern Collection**

Table C-33. Cosine similarity results for Design Problem 1.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Educational Forums	0.37
Personal E-Commerce	<u>0.52</u>
Content Modules	0.15
Category ages	0.21
Home Page Portal	0.31

Table C-34. Cosine similarity results for Design Problem 2.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Enabling Intranets	0.45
Page Templates	0.32
News Mosaics	<u>0.55</u>
Content Modules	0.31
About Us	0.24

Table C-35. Cosine similarity results for Design Problem 3.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Multiple ways to navigate	0.40
Site Accessibility	0.32
Site Branding	<u>0.43</u>
Message Boards	0.25
Content Modules	0.21
Headline and Blurbs	0.29

Table C-36. Cosine similarity results for Design Problem 4.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Printable Pages	0.15
Page Templates	0.32
Category Ages	0.35
Multiple ways to navigate	0.18
Browseable Content	<u>0.49</u>

Table C-37. Cosine similarity results for Design Problem 5.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Home Page Portal	0.15
Site Branding	0.32
Navigation Bar	0.35
Tab Rows	<u>0.52</u>

Table C-38. Cosine similarity results for Design Problem 6.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Personalized Content	0.15
Page Templates	<u>0.62</u>
Secure Connection	0.42
Message Board	0.24
About Us	0.10
Browseable Content	0.21

Table C-39. Cosine similarity results for Design Problem 7.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Shopping Cart	0.18
Easy Return	<u>0.50</u>
Payment Method	0.14
Content Modules	0.43
Feature Products	0.21
Printable Pages	0.22

Table C-40. Cosine similarity results for Design Problem 8.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Site Branding	<u>0.45</u>
Privacy Policy	0.39
Message Board	0.33
Category Ages	0.25

Table C-41. Cosine similarity results for Design Problem 9.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
E-mail Subscriptions	0.35
Category Ages	0.27
About Us	0.31
Process Funnel	0.15
Guest Account	<u>0.43</u>

Table C-42. Cosine similarity results for Design Problem 10.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Quick Flow Checkpoint	<u>0.48</u>
Payment Method	0.34
Gift Giving	0.15
Multiple Destination	0.27

Table C-43. Cosine similarity results for Design Problem 11.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Personalized Recommendation	0.42
Cross Selling and Up Selling	0.26
About Us	0.33
Clean product Details	0.11
Multiple Destination	<u>0.51</u>

Table C-44. Cosine similarity results for Design Problem 12.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Feature Products	<u>0.62</u>
Gift Giving	0.26
Cross Selling and Up Selling	0.47
Secure Connection	0.23
Pop-up Window	0.51

Table C-45. Cosine similarity results for Design Problem 13.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Cross Selling and Up Selling	0.48
Tab Rows	0.25
Search Action Module	<u>0.52</u>
Action Buttons	0.32

Table C-46. Cosine similarity results for Design Problem 14.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Guest Account	0.49
Embedded Link	0.26
Process Funnel	<u>0.58</u>
Account Management	0.21
External Links	0.13
Embedded Links	0.11

Table C-47. Cosine similarity results for Design Problem 15.

Design Patterns of Predicted Pattern Class	Cosine Similarity (CS)
Account Management	0.57
Persistent Customer Session	0.16
Clear First Read	0.48
Secure Connection	0.13
Organized Search Result	0.21

Appendix D

Supervised Learning Techniques

1. Naïve Bayes (NB)

The Naïve Bayes (NB) is a probabilistic classifier which based on Bayes theorem. Naïve Bayes works on an assumption that all features are independent of each other and calculate a probability score by multiplying the conditional probabilities with each other. It is supposed that the pattern class $L(p_i)$ of pattern p_i , has some relation to words appears in the description of Problem Definition of design patterns and design problems. The Bayesian formula describes in Equation D-1 yields the probability of a pattern class given the words used in the description of patterns and design problem.

$$p(L_i | t_1, \dots, t_{m_i}) = \frac{p(t_1, \dots, t_{m_i} | L_k) p(L_k)}{\sum_{l \in L} p(t_1, \dots, t_{m_i} | l) p(l)} \quad (D-1)$$

The prior probability $p(l)$ depicts the probability that design problem belongs to a class label $l \in L$ before its known words. The conditional probability of words (includes in the description of patterns and design problem) given a class L_k is described in Equation D-2.

$$p(t_1, \dots, t_{m_i} | L_k) = \prod_{j=1}^{m_i} p(t_j | L_k) \quad (D-2)$$

Estimation of the probabilities of words $p(t_j | L_k)$ is required to make the learning of Naive Bayes classifier which depends on their frequencies in documents (i.e. Design patterns) of the training set.

2. C4.5 Decision Tree

The decision tree based algorithms generate a set of rules to make the classification decisions. The C4.5 algorithm used the concept of information entropy and makes the decision from the training set. In order to predict the document's class, the word t_i is selected from a training set T of labeled documents. Subsequently, training set T is partitioned into two subsets T^+ (documents with word t_i) and T^- (documents without the word t_i). The same step is repeated and applied to T^+ and T^- . The recursive procedure is stopped when all documents belong to the same class L_k .

3. Support Vector Machines (SVM)

A Support Vector Machine (SVM) is considered as one of the most effective text categorization algorithms, which looks for a decision surface and determines a margin through its closest data points. Usually, an SVM algorithm can separate documents of a training set into two classes, $y=+1$ for the positive and $y=-1$ for the negative class. Subsequently, the Equation D-4 is used to define a hyper plane at $y=0$ for the set of input vectors. Each input vector for the document d is represented as a count of words as depicted in the equation D-3.

$$\vec{t_d} = t_{d1}, \dots, t_{dM} \quad (D-3)$$

$$y = f(\vec{t_d}) = b_0 + \sum_{j=1}^M b_j t_{dj} \quad (D-4)$$

The SVM algorithm classifies a new document with $\vec{t_d}$ into a positive class if $f(\vec{t_d}) > 0$ otherwise into negative class.

References:

- [1] L. Bass, P. Clements, and R. Kazman, "Software architecture in practice," Addison-Wesley Professional. 3rd Edition, 2012.
- [2] W. G. Wood, "A practical example of applying Attribute-Driven Design (ADD)," Version 2.0, Technical Report, Software Engineering Institute, 2007.
- [3] C. Alexander, "The timeless way of building," Oxford University Press, New York, 1979.
- [4] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel, "A pattern language : Towns, building, constructions," Oxford University Press, New York, 1977.
- [5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design patterns : Elements of reusable object-oriented software," Boston, MA : Addison-Wesley, 1995.
- [6] W. Pree, "Design patterns for object-oriented software development," ACM Press/Addison-Wesley, 1995.
- [7] P. Coad, D. North, and M. Mayfield, "Object models : Strategies, patterns, applications," Yourdon Press, NJ, USA, 1995.
- [8] W. F. Tichy, "A catalogue of general-purpose software design patterns," Proceedings of Technology of Object-Oriented Languages and Systems, pp. 330-339, 1997.
- [9] C. Bouhours, H. Leblance, and C. Percebois, "Spoiled patterns : How to extend the GoF," Software Quality Journal, vol. 23, pp. 661-694, 2015.
- [10] M. Adams, J. Coplien, R. Gamoke, R. Hanmer, F. Kieve, and K. Nicodemus, "Fault-tolerant telecommunications system patterns," Proceedings of Pattern Languages of Programs (PLoP), pp. 549-562, 1996.
- [11] L. Rising, "Design patterns in communications software," SIGS Reference Library, ISBN: 9780521790406, Oct. 2001.
- [12] B. P. Douglass, "Real-time design patterns : Robust scalable architecture for real-time systems," Addison-Wesley/Longman Publishing Co, Inc., 2002.
- [13] V. Duyne, J. A. Landay, and J. I. Hong, "The design of sites : patterns, principles and processes for crafting a customer-centered web experience," Addison-Wesley, Boston, 2003.
- [14] J. Tidwell, "Designing interfaces : Pattern for effective interaction design," O'Reilly Media, 2nd Edition, 2010.
- [15] M. Schumacher, E. Fernandez, D. Hybertson, and F. Buschmann, "Security patterns : Integrating security and systems engineering," John Wiley and Sons, 2006.
- [16] H. Baraki, K. Geith, C. Voigtmann, A. Hoffmann, R. Kniewl, B-E. Macek, and J. Zirfas, "Interdisciplinary design patterns for socially aware computing," Proceedings of 37th International Conference on Software Engineering (ICSE), 2015.
- [17] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stahl, "Pattern-oriented software architecture : A system of patterns," vol. 1, Wiley, 1996.
- [18] D. C. Schmidt, M. Stal, H. Rohnert, and F. Buschmann, "Pattern-oriented software architecture : Patterns for concurrent and networked objects," vol. 2, Wiley, 2000.
- [19] M. Kircher, and O. Jain, "Pattern-oriented software architecture : Patterns for resource management," vol. 3, Wiley, 2004.

- [20] F. Buschmann, K. Henney, and D. C. Schmidt, "Pattern-oriented software architecture : A pattern language for distributed computing," vol. 4, Wiley, 2007.
- [21] F. Buschmann, K. Henney, and D. C. Schmidt, "Pattern-oriented software architecture : On patterns and pattern languages," vol. 5, Wiley, 2007.
- [22] A. Birukou, "A survey of existing approaches for pattern search and selection," Proceeding of 17th Pattern Languages of Programs (PLoP), 2010.
- [23] S. Henninger and V. Correa, "Software pattern communities : Current practices and challenges." Proceedings of the 14th Conference on Pattern Languages of Programs, 2007.
- [24] I. Sommerville, Software engineering, 7th Edition. Addison-Wesley, Boston, MA, USA, 2004.
- [25] G. Booch, "Handbook of software architecture," 2006. <http://handbookofsoftwarearchitecture.com/>
- [26] P. Velasco-Elizondo, R. Marin-Pina, S. Vazquez-Reyes, A. Mora-Soto, and J. Mejia, "Knowledge representation and information extraction for analyzing architectural patterns," Science of Computer Programming, vol. 121, pp. 176-189, 2016.
- [27] D. Zoya, "Architectural design decision documentation through reuse of design patterns," The Karlsruhe Series on Software Design and Quality, KIT Scientific Publishor, 2014.
- [28] S. Hasso and C. R. Carlson, "A theoretically-based process for organizing design patterns," Proceedings of 12th Pattern Language of Patterns (PLoP), 2005.
- [29] W. Zimmer, "Relationships between design patterns," Journal of Pattern Languages of Program Design, vol. 1, pp. 345–364, 1995.
- [30] G. J. Kim and J.S. Han, "Clustering algorithm of design pattern using object-oriented relationship," Proceedings of Computational Science and Its Applications, LNCS, Springer, ICCSA, p. 997-1006, 2007.
- [31] S. M. Hasheminejad and S. Jalili, "Design patterns selection : An automatic two-phase method," Journal of System and Software, vol. 85, pp. 408-424, 2012.
- [32] S. Hussain, J. Keung, A.A. Khan, and K.E. Bennin, "A methodology to automate the selection of design patterns," Proceedings of IEEE International Conference on Computers, Software and Application, 2016.
- [33] S. Hussain, J. Keung, and A.A. Khan, "Software design patterns classification and selection using text categorization approach ," Applied Soft Computing, vol. 58, pp. 225-244, 2017.
- [34] D. K. Kim and C.E. Khawand, "An approach to precisely specifying the problem domain of design patterns," Journal of Visual Languages and Computing, vol. 18, pp. 560-591, 2007.
- [35] N. L. Hsueh, J-Y. Kuo, and C-C. Lin, "Object-oriented design : A goal-driven and pattern-based approach," Journal for Software and Systems Modeling, vol. 8, no. 1, pp.1-18, 2007.
- [36] D. K. Kim and W. Shen, "Evaluating pattern conformance of UML models : A divide and conquer approach and case studies," Software Quality Journal, vol. 16, no. 3, pp. 329-359, 2008.
- [37] S. Hasso and C. R. Carlson, "A theoretically-based process for organizing design patterns," Proceedings of 12th Pattern Language of Patterns (PLoP), 2005.
- [38] E. Blomqvist, "Pattern ranking for semiautomatic ontology construction," Proceedings of ACM Symposium on Applied Computing, pp. 2248-2255, 2008.

- [39] P.E. Khoury, A. Mokhtari, E. Coquery, and M.S. Hacid, "An ontological interface for software developers to select security patterns," *Proceedings of 19th International Conference on Database and Expert Systems Application*, (DEXA'08), pp. 297-301, 2008.
- [40] I. Idris and A. Selamat, "Improved email spam detection model with negative selection algorithm and particles warm optimization," *Applied Soft Computing*, vol. 22, pp. 11-27, 2014.
- [41] E. Sarac and S.A. Ozel, "An ant colony optimization based feature selection for web page classification," *The Scientific World Journal*, pp. 1-16, 2014.
- [42] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications : A survey," *Ain Shams Engineering Journal*, vol. 5, pp. 1093-1113, 2014.
- [43] C. Zhang, X. Wu, Z. Niu, and W. Ding, "Authorship identification from unstructured texts," *Knowledge Based Systems*, vol. 66, pp. 99-111, 2014.
- [44] E. Alpaydin, *Introduction to Machine Learning*, 2nd Edition. The MIT Press Cambridge, Massachusetts, London, England, 2010.
- [45] F. Sebastiani, "Machine learning in automated text categorization," *Journal of ACM Computing Survey*, vol. 34, pp. 1-47, 2002.
- [46] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3133-3181, 2014.
- [47] A. K. Uysal, "An improved global feature selection scheme for text classification," *Expert System with Applications*, vol. 43, pp. 82-92, 2016.
- [48] S. Tascı and T. Güngör, "Comparison of text feature selection policies and using an adaptive framework," *Expert Systems with Applications*, vol. 40, pp. 4871-4886, 2013.
- [49] B. B. Mayvan, A. Rasoolzadegan, and Z. G. Yazdi, "The state of the art on design patterns: A systematic mapping of the literature," *The Journal of System and Software*, vol. 125, pp. 93-118, 2016.
- [50] A. Shvets, *Design Pattern Explained Simply*, <https://sourcemaking.com/design-patterns-ebook>, 2009.
- [51] E. Nilsson, "Design patterns for user interface for mobile applications," *Advances in Software Engineering*, vol. 40, no. 12, pp. 1318-1328, 2009.
- [52] K. Lano and S. K. Rahimi, "Model-transformation design patterns," *IEEE Transaction on Software Engineering*, vol. 40, no. 12, pp. 1224-1259, 2014.
- [53] A. Ampatzoglou, S. Charalampidou, and I. Stamelos, "Research sate of the art on GoF design patterns : A mapping study," *Journal of System and Software*, vol. 86, pp. 1945-1964, 2013.
- [54] C. Zhang and D. Budgen, "A survey of experienced user perceptions about software design patterns. *Information and Software Technology*," vol. 55, no. 5, pp. 822-835, 2013.
- [55] M. O. Elish and M. A. Mohammed, "Quantitative analysis of fault density in design patterns," *Information and Software Technology*, vol. 66, pp. 58-72, 2015.
- [56] A. Birukou, "A survey of existing approaches for pattern search and selection," *Proceeding of Pattern Languages of Programs (PLoP)*, 2010.

- [57] T-H. Chen, S. W. Thomas, and A. E. Hassan, "A survey on the use of topic models when mining software repositories," *Empirical Software Engineering*, vol. 21, pp. 1843-1919, 2016.
- [58] T. P. Williams and J. Gong, "Predicting construction cost overruns using text mining, numerical data and ensemble classifiers," *Automation in Construction*, vol. 43, pp. 23-29, 2014.
- [59] W. Hu, O. Wu, Z. Chen, Z. Fu, and S. Maybank, "Recognition of pornographic web pages by classifying texts and images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1019-1034, 2007.
- [60] J. Sivic and A. Zisserman, "Efficient visual search of videos cast as text retrieval," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 21, no. 4, pp. 591-605, 2009.
- [61] Y. Chen, Y. Zhou, S. Zhu, and H. Xu, "Detecting offensive language in social media to protect adolescent online safety," *IEEE International Conference on Social Computing (SocialCom)*, pp. 71-80, 2012.
- [62] W.B. Cavnar and J. M. Trenkle, "N-Gram-Based text categorization," *Proceedings of 3rd Annual Symposium on Document Analysis and Information Retrieval*, pp. 161-175, 1994.
- [63] R. Feldman and I. Dagan, "Knowledge discovery in texts," *Proceedings of the 1st International Conference on Knowledge Discovery (KDD)*, pp. 112-117, 1995.
- [64] A. Hotho, A. Nurnberger, and G. Paab, "A brief survey of text mining," *Journal for Computational Linguistics and Language Technology*, vol. 20, pp. 19-62, 2005.
- [65] C. J. V. Rijsbergen, *Information Retrieval*, 2nd Edition, Butterworth-Heinemann, Newton, MA, USA, 1979.
- [66] C. Fox, "Lexical analysis and stoplists. *Information Retrieval - Data Structures & Algorithms*," Prentice-Hall, pp. 102-130, 1992.
- [67] G. K. Zipf, "Human behavior and the principle of least effort," Addison-Wesley 1949.
- [68] R. T. W. Lo, B. He, and L. Ounis, "Automatically building a stopword list for an information retrieval system," *Journal on Digital Information Management: Special Issue on the 5th Dutch-Belgian Information Retrieval Workshop (DIR)*, 2005.
- [69] T. M. Cover and J. A. Thomas, "Elements of information theory," John Wiley & Sons, 2nd Edition, 2006.
- [70] M. F. Porter, "An Algorithm for suffix stripping," *Journal of Program-Electronic Library and Information Systems*, vol. 40, pp. 211-218, 2006.
- [71] D. C. Paice, "Another stemmer," *ACM SIGIR Forum*, vol. 24, no. 3, pp. 56-61, 1990.
- [72] J. B. Lovins, "Development of a stemming algorithm," *Mechanical Translation and Computational Linguistics*, vol. 11, nos. 1 and 2, pp. 22-31, 1968.
- [73] G. Adamson and J. Boreham, "The use of an association measure based on character Structure to identify semantically related pairs of words and document titles," *Information Storage and Retrieval*, vol. 10, pp. 253-260, 1974.
- [74] J. H. Paik, M. Mitra, S. K. Parui, and K. Jarvelin, "GRAS : An effective and efficient stemming algorithm for information retrieval," *ACM Transaction on Information System*, vol. 29, no. 4, pp. 19:1-19:24, 2011.
- [75] M. Massimo and O. Nicola, "A novel method for stemmer generation based on hidden markov models," *Proceedings of the 12th international conference on Information and Knowledge Management*, pp. 131-138, 2003.

- [76] P. Majumder, M. Mitra, S. K. Parui, G. Koley, P. Mitra, and K. Datta, "YASS : Yet another suffix stripper," *ACM Transactions on Information Systems*, vol. 25, no. 4, 2007.
- [77] K. Robert, "Viewing morphology as an inference process," *Proceedings of 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 191-202, 1993.
- [78] D. A. Hull and G. Grefenstette, "A detailed analysis of english stemming algorithms, Xerox Research and Technology, Technical Report, <http://www.xrce.xerox.com>, 1996.
- [79] X. Jinxi and W. B. Croft, "Corpus-based stemming using co-occurrence of word variants," *ACM Transactions on Information Systems*, vol. 16, no. 1, pp. 61-81, 1998.
- [80] F. Peng, N. Ahmed, X. Li, and Y. Lu, "Context sensitive stemming for web search," *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 639-646, 2007.
- [81] G. Salton, "The SMART retrieval system : Experiments in automatic document processing," Prentice-Hall, Upper Saddle River, NJ, 1971.
- [82] G. Salton, A. Wong, and C-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613-620, 1975.
- [83] I. H. Witten and E. Frank, "Data Mining : Practical Machine Learning Tools and Techniques with Java Implementations," Morgan Kaufmann, San Francisco, 2005.
- [84] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman., "Indexing by latent semantic analysis," *Journal of the American Society for Information Science (JASIS)*, vol. 41, no. 6, pp. 391-407, 1990.
- [85] D. Pado and M. Lapata, "Dependency-based construction of semantic space models," *Computational Linguistics*, vol. 33, no. 2, pp. 161-199, 2007.
- [86] M. Sahlgren, "The word-space model: using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces," Ph.D. thesis, Department of Linguistics, Stockholm University, 2006.
- [87] D. Lin and P. Pantel, "DIRT: Discovery of Inference Rules from Text," *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 323-328, 2001.
- [88] P. D. Turney and P. Pantel, "From frequency to meaning: Vector Space Models of semantics," *Journal of Artificial Intelligence Research*, vol. 37, pp. 141-188, 2010.
- [89] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing and Management*, vol. 24, no. 5, pp. 513-523, 1988.
- [90] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *Journal of Machine Learning Research*, vol. 3, pp. 1289-1305, 2003.
- [91] K. Aas and L. Eikvil, "Text categorization : A survey," Technical report, Norwegian Computing Center, 1999.
- [92] Y. Yang and J.O. Pedersen, "A comparative study on feature selection in text categorization," *Proceedings of the 14th International Conference on Machine Learning*, pp. 412-420. 1997.

- [93] H. Uguz, "A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm," *Knowledge-Based Systems*, vol. 24, no. 7, pp. 1024-1032, 2011.
- [94] C. Lee, and G.G. Lee, "Information gain and divergence-based feature selection for machine learning-based text categorization," *Information Processing and Management*, vol. 42, no. 1, pp. 155-165, 2006.
- [95] C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.
- [96] W. Shang, H. Huang, H. Zhu, Y. Lin, Y. Qu, and Z. Wang, "A novel feature selection algorithm for text categorization," *Expert Systems with Applications*, vol. 33, no. 1, pp. 1-5, 2007.
- [97] S.S.R. Mengle and N. Goharian, "Ambiguity measure feature-selection algorithm," *Journal of the American Society for Information Science and Technology*, vol. 60, pp. 1037-1050, 2009.
- [98] A. K. Uysal and S. Gunal, "A novel probabilistic feature selection method for text classification," *Knowledge-Based Systems*, vol. 36, pp. 226-235, 2012.
- [99] S. Deerwester, S.T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of American Society Information Science*, vol. 41, no. 6, pp. 391-407, 1990.
- [100] R. B. Yates and B. R. Neto, "Modern information retrieval," ACM Press, New York, 1999.
- [101] G. Salton and M. J. McGill, "Introduction to modern information retrieval," McGraw-Hill, New York, 1983.
- [102] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993-1022, 2003.
- [103] X. Hu, Z. Cai, D. Franceschetti, P. Penumatsa, and A. C. Graesser, "LSA : the first dimension and dimensional weighting," *Proceedings of the 25th meeting of the Cognitive Science Society*, Boston. Cognitive Science Society, pp. 1-6, 2003.
- [104] P. Comon, "Independent component analysis, a new concept," *Signal Processing*, vol. 36, no. 3, pp. 287-314, 1994.
- [105] S. Grant and J. R. Cordy, "Vector space analysis of software clones," *Proceedings of the 17th international conference on program comprehension*, pp. 233-237, 2009.
- [106] T. Hofmann, "Probabilistic latent semantic indexing," *Proceedings of the 22nd International Conference on research and development in information retrieval*, pp. 50-57, 1999.
- [107] T. Hofmann, "Unsupervised learning by probabilistic latent semantic analysis," *Machine Learning*, vol. 42, no. 1, pp. 177-196, 2001.
- [108] C. X. Zhai, "Statistical language models for information retrieval," *Synthesis Lectures on Human Language Technologies*, vol. 1, no. 1, pp. 1-141, 2008.
- [109] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993-1022, 2003.
- [110] D. M. Blei and J. D. Lafferty, "Topic models. *Journal of Text mining : classification, clustering, and applications*," Chapman & Hall, London, pp. 71-94, 2009.
- [111] M. Steyvers and T. Griffiths, "Probabilistic topic models, latent semantic analysis : A road to meaning," *Laurence Erlbaum*, pp. 1-15, 2007.

- [112] D. M. Blei, T. L. Griffiths, and M. I. Jordan, "The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies," *Journal of the ACM (JACM)*, vol. 57, no. 2, pp. 1-30, 2010.
- [113] L. Jiang, Z. Cai, H. Zhang, and D. Wang, "Naïve Bayes text classifiers : A locally weighted learning approach," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 25, pp. 273-286, 2013.
- [114] R. O. Duda and P. E. Hart, "Pattern classification and scene analysis," J. Wiley & Sons, New York, NY, USA, 1973.
- [115] C. D. Manning and H. Schütze, "Foundations of statistical natural language processing," MIT Press, Cambridge, MA, 2001.
- [116] M. Steinbach, G. Karypis, and V. Kumara., "A comparison of document clustering techniques," *Proceedings of KDD Workshop on Text Mining*, 2000.
- [117] J. Hartigan, "Clustering algorithms," John Wiley and Sons, New York, 1975.
- [118] T. Kohonen, "Self-organization and associative memory," SpringerVerlag, Berlin, 1984.
- [119] T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning," Springer, 2001.
- [120] J. C. Bezdek, R. Ehrlich, W. Full, "FCM : The Fuzzy c-means clustering algorithm," *Journal of Computers and Geosciences*, vol. 10, no. 2, pp. 191-203, 1984.
- [121] A. Rosenberg and J. Hirschberg, "V-Measure : A conditional entropy-based external cluster evaluation measure," *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 410-420, 2007.
- [122] A. Bartilla and C. Koppe, "Awareness seeds for more gender diversity in computer science education," *Proceedings of EuroPLoP*, 2015.
- [123] Y.-G. Guéhéneuc and G. Antoniol, "DeMIMA : a multilayered approach for design pattern identification," *IEEE Transaction on Software Engineering*, vol. 34, no. 5, pp. 667-684, 2008.
- [124] N. Tsantalis, A. Chatzigeorgiou, G. Stephanides, and S.T. Halkidis, "Design pattern detection using similarity scoring," *IEEE Transaction on Software Engineering*, vol. 32, no. 11, pp. 896-909, 2006.
- [125] F. A. Fontana and M. Zanoni, "A tool for design pattern detection and software architecture reconstruction," *Information Sciences*, vol. 181, no. 7, pp.1306-1324, 2011.
- [126] D. Jing, D.S. Lad, and Z. Yajing, "DP-Miner: design pattern discovery using matrix," *Proceedings of 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, pp. 371-380, 2007.
- [127] J. Niere, W. Schafer, J. P. Wadsack, L. Wendehals, and J. Welsh, "Towards pattern based design recovery," *Proceedings of the 24th International Conference on Software Engineering, (ICSE)*, pp. 338-348, 2002.
- [128] Y.-G. Guéhéneuc, "Ptidej : Promoting patterns with patterns," *Proceedings of the First ECOOP Workshop on Building a System using Patterns*, Glasgow, UK, 2005.
- [129] S. Nija and R.A. Olsson, "Reverse engineering of design patterns from java source code," *Proceedings of 21st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 123-134, 2006.
- [130] J. Dietrich and C. Elgar, "Towards a web of patterns," *Web Semantics : Science, Service and Agents on the World Wide Web*, vol. 5, no. 2, pp. 108-116, 2007.

- [131] A. Binum and G. Kniesel, "Joining forces for higher precision and recall of design pattern detection," Technical Report, Proceedings of Conference on Software Maintenance and Reengineering (CSMR), 2012.
- [132] H. Dabain, A. Manzer, and V. Tzerpos, "Design pattern detection using FINDER," Proceedings of SAC2015, 2015.
- [133] 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," Proceedings of 6th European Conference, Aberdeen, Scotland, UK, 2002.
- [134] A. Birukou, "Implicit culture framework for behavior transfer : Definition, implementation and applications," PhD Thesis, 2009.
- [135] G. S. Hang, L. Y. Qing, J. M. Zhong, G. Jing, and H. Juan, "A requirement analysis pattern selection method for E-Business project situation," Proceeding of IEEE International Conference on e-Business Engineering, 2007.
- [136] Q. Wu, Y. Ye, H. Zhang, M. K. Ng, and S-S. Ho, "FORESTEXTER : An efficient random forest algorithm for imbalanced text categorization," Knowledge-Based Systems, vol. 67, pp. 105-116, 2014.
- [137] H. Ogura, H. Amano, and M. Kondo, "Comparison of metrics for feature selection in imbalanced text classification," Expert Systems with Applications, vol. 38, pp. 4978-4989, 2011.
- [138] S. Guna.l, "Hybrid feature selection for text classification", Turkish Journal of Electrical Engineering and Computer Sciences, vol. 20, pp. 1296-1311, 2012.
- [139] B. Walter and T. Alkhaeir, "The relationship between design patterns and code smells : An exploratory study," Information and Software Technology, vol. 74, pp. 127-142, 2016.
- [140] S. Hussain, J. Keung, and A.A. Khan, "The effect of Gang-of-Four design patterns usage on design quality attributes," Proceedings of IEEE International Conference on Software Quality, Reliability and Security, 2017.
- [141] A. Torres, R. Galante, M. S. Pimentam, and A. J. B. Martins, "Twenty years of object-relational mapping : A survey on patterns, solutions, and their implications on application design," Information and Software Technology , vol. 82, pp. 1-18, 2017.
- [142] R. M. Maracini and S. O. Rezende, "PreText : A simple text preprocessing tool," LABIC Lab, 1995. <http://sites.labic.icmc.usp.br/torch/msd2011/jpretext/>
- [143] L. Rokach and O. Maimon, "Clustering methods," in Data Mining and knowledge Discovery Handbook, Springer, 2010.
- [144] A. Huang, "Similarity measures for text document clustering," Proceedings of New Zealand Computer Science Research Student Conference (NZCSRSC), 2008.
- [145] R. E. Bonner, "On some clustering techniques," IBM journal of research and development, pp. 22-32, 1964.
- [146] R. O. Duda, P. E. Hart and D. G. Stork, "Pattern classification," Wiley, New York, 2001.
- [147] J. F. Marcotorchino, "Optimization ordinal data analysis," 1979.
- [148] J. J. Fortier and H. Solomon, "Clustering procedures," Proceedings of the Multivariate Analysis, pp. 493-506, 1996.
- [149] M. A. Gluck and J. E. Corter, "Information, uncertainty, and the utility of categories," Proceedings of the 7th Annual Conference of the Cognitive Science Society, pp. 283-287, 1985.
- [150] C. C. Aggarwal and C.K. Reddy, "Data Clustering : Algorithms and Applications," CRC Press, 2013.

- [151] S. Romano, N. X. Vinh, J. Bailey and K. Verspoor, "Adjusting for chance clustering comparison measures," *Journal of Machine Learning Research*, vol. 17, pp. 1-32, 2016.
- [152] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846-850, 1971.
- [153] A. Ben-Hur, A. Elisseeff, and I. Guyon, "A stability based method for discovering structure in clustered data," *Proceedings of Pacific Symposium on bio computing*, vol. 7, pp. 6-17, 2001.
- [154] A. Strehl and J. Ghosh, "Cluster ensembles - a knowledge reuse framework for combining multiple partitions," *The Journal of Machine Learning Research*, vol. 3, pp. 583-617, 2003.
- [155] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison : Variants, properties, normalization and correction for chance," *Journal of Machine Learning Research*, vol. 11, pp. 2837-2854, 2010.
- [156] J. Wu, H. Xiong, and J. Chen, "Adapting the right measures for k-means clustering," *Proceedings of Knowledge Discovery and Data Mining*, pp. 877-886, 2009.
- [157] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, pp. 193-218, 1985.
- [158] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison : is a correction for chance," *Journal of Machine Learning Research*, vol. 11, pp. 2837-2854, 2010.
- [159] C. Edwards, "Growing pains for deep learning," *Communication of The ACM*, vol. 58, no. 7, 2015.
- [160] X. Yang, D. Lo, X. xia, Y. Zhang, and J. Sun, "Deep learning for just-in-time defect prediction," *Proceedings of IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pp. 17-26, 2015.
- [161] S. Wang, T. Liu, and L. Tan, "Automatically Learning Semantic Features for Defect Prediction," *Proceeding of IEEE International Conference on Software Engineering (ICSE)*, 2016.
- [162] A. Lam, A. Nguyen, H. Nguyen, and T. Nguyen, "Combining deep learning with information retrieval to localize buggy files for bug reports," *Proceedings of 30th IEEE Automated Software Engineering (ASE)*, pp. 476-481, 2015.
- [163] M. White, C. Vendome, M. L. Vasquez, and D. Poshyvanyk, "Toward deep learning software repositories," *Proceedings of MSR'15*, pp. 334-345, 2015.
- [164] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," *Proceeding of ICASSP*, pp. 1916-1920, 2015.
- [165] T. Bengio, A. Courville, and P. Vincent, "Representation learning : A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798-1828, 2013.
- [166] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *PMD*, vol. 313, pp. 504-507, 2006.
- [167] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vl. 18, no. 7, pp. 1527-1554, 2006.
- [168] S. Hussain, J. Keung, and A. A., Khan, "A framework for ranking of software design patterns," *Proceedings of International Conference on Comple, Intelligent and Software Intensive Systems (CISIS)*, 2017.

[169] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, “An empirical comparison of model validation techniques for defect prediction models,” *IEEE Transactions on Software Engineering*, vol. PP(99), 2016.