UNIVERSITI PUTRA MALAYSIA

A HYBRID METHOD OF FEATURE EXTRACTION AND NAIVE BAYES CLASSIFICATION FOR SPLITTING IDENTIFIERS

NAHLA ALANEE

FSKTM 2016 32

A HYBRID METHOD OF FEATURE EXTRACTION AND NAÏVE BAYES CLASSIFICATION FOR SPLITTING IDENTIFIERS

NAHLA ALANEE

MASTER OF COMPUTER SCIENCE
UNIVERSITI PUTRA MALAYSIA

DECEMBER 2016

© COPYRIGHT UPM

DEDICATIONS

To my beloved and darling, master of humans, the light of knowledge, the teacher of truth. . . The prophet and the messenger of Allah, Mohammed peace and blessing be upon him.

To the soul of my father and brother who passed away during my study, the grief in our hearts is big, but bigger is the love; they will live with us forever.

To the joy of the soul, balm of life, and presence tune that
I live on . . . my darling mother.

To whom I shelter in the space of their love and delight under their closeness moments . . . my brothers.

To the amiability of difficult days and the moon of dark nights . . . my Precious husband.

To my beloved and my eyes . . . my son Mohamed and my daughter Iceel.

To the idea's companions and the messengers of fraternity and love . . . dear friends.

To each of who has taught me or gave me advice . . . teachers.

To every Muslim in this world.

To everybody in this world who understands the meaning of humanity.

Dedicate this work . . .

© COPYRIGHT UPM

ACKNOWLEDGMENTS

First and foremost, praise be to Allah for granting me patience and strength throughout my journey to complete this study.

I would like to express my sincere gratitude to my supervisor Assoc. Prof. Dr. Masrah Azrifah Azmi Murad for her guidance and support throughout this work. She has been a great source of inspiration to me. No words can express how grateful I am to her.

To the soul of my father and brother who passed away during my study, death took their body, but their spirit will remain with us and in our hearts forever.

To my darling mother and husband, and my two little angles Mohamed and Iceel. This work is dedicated for you.

© COPYRIGHT UPM

ABSTRACT

Nowadays, integrating natural language processing techniques on software systems has caught many researchers' attentions. Such integration can be represented by analyzing the morphology of the source code in order to gain meaningful information. Feature location is the process of identifying specific portions of the source code. One of the most important information lies on such source code is the identifiers (e.g. Student). Unlike the traditional text processing, the identifiers in the source code is formed as multi-word such as 'Employee-Name'. Such multi-words are not divided using white space, instead it can be formed using special characters (e.g. Employee_ID), CamelCase (e.g. EmployeeName) or using abbreviations (e.g. EmpNm). This makes the process of extracting such identifiers more challenging. Several approaches have been performed to resolve the problem of splitting multi-word identifiers. However, there is still room for improvement in terms of accuracy. Such improvement can be represented by utilizing more robust features that have the ability to analyses the morphology of identifiers. Therefore, this study aims to propose a hybrid method of feature extraction and Naïve Bayes classifier in order to separate multi-word identifiers within source code. The dataset that has been used in this study is a benchmark-annotated data that contains large number of Java codes. Multiple experiments have been conducted in order to evaluate the proposed features independently and with combinations. Results shown that the combination of all features have obtained the best accuracy by achieving 64.7% of f-measure. Such finding implies the usefulness of the proposed features in terms of discriminating multi-word identifiers.

© COPYRIGHT UPM

TABLE OF CONTENTS

		Page
DEDICATIONS		ii
ACKNOWLEDGMENTS		iii
ABSTRACT		iv
CONTENTS		v
LIST OF TABLES		vii
LIST OF FIGURES		ix
LIST OF ABBREVIATIONS		x
CHAPTERS		
CHAPTER I	**INTRODUCTION**	
1.1	Introduction	1
1.2	Significance of Study	2
1.3	Problem Statement	2
1.4	Research Objectives	3
1.5	Research Scope	4
1.6	Research Methodology	4
1.7	Thesis Organization	6
1.8	Summary	7
CHAPTER II	**LITERATURE REVIEW**	
2.1	Introduction	8
2.2	Feature Location	8
2.3	The Dimensions Of Feature Location Task	9
	2.3.1 Analysis	9
	2.3.2 Input	10
	2.3.3 Data Sources	10
	2.3.4 Output	10
	2.3.5 Programming Language Support	11

© COPYRIGHT UPM

	2.3.6 Evaluation	11
2.4	Feature Location Techniques	11
	2.4.1 Dynamic Feature Location	12
	2.4.2 Static Feature Location	13
	2.4.3 Textual Feature Location	14
2.5	Splitting Identifiers	15
2.6	Naïve Bayes (NB)	17
2.7	Related Work	18
2.8	Summary	21
CHAPTER III	**RESEARCH METHOD**	
3.1	Introduction	23
3.2	Phase 1: Problem Identification	23
3.3	Phase 2: Design	24
3.4	Phase 3: Implemenation	24
	3.4.1 Dataset	25
	3.4.2 Transformation	26
	3.4.3 Feature Extraction	27
	3.4.4 Classification	30
3.5	Phase 4: Evaluation	31
3.6	Summary	31
CHAPTER IV	**EXPERIMENT SETTING**	
4.1	Introduction	33
4.2	Collecting And Dispalying The Data	33
4.3	Transformation	34
4.4	Developing The Features	35
4.5	Naïve Bayes Classification	37
4.6	Evaluation	43
4.7	Summary	44
CHAPTER V	**EXPERIMENTAL RESULTS**	
5.1	Introduction	45
5.2	Resulsts Of Independent Features	45

© COPYRIGHT UPM

5.3	Results Of Pair Combinaiton Of Features	47
5.4	Results Of Full Combinations Of Features	48
5.5	Summary	50
CHAPTER VI	**CONCLUSION AND FUTURE WORK**	
6.1	Introduction	51
6.2	Research Conclusion	51
6.3	Research Contribution	52
6.4	Future Work	52
6.5	Summary	53
REFERENCES		54

© COPYRIGHT UPM

LIST OF TABLES

Table No. **Page**

Table 2.1 Summary of separation mechanisms 16

Table 2.2 Summary of related work 19

Table 3.1 Sample of count capital feature..... 28

Table 3.2 Sample of count punctuation feature 28

Table 3.3 Sample of digit count feature..... 29

Table 4.1 Sample data for NB computation..... 37

Table 4.2 Dividing the data into training and testing..... 38

Table 4.3 Sample of actual and predicted class labels..... 43

Table 4.4 Validating the predicted class labels..... 44

Table 5.1 Results of independent features 45

Table 5.2 Results of pair combinations..... 47

Table 5.3 Results of full combinations of the features 49

© COPYRIGHT UPM

LIST OF FIGURES

Figure No. **Page**

Figure 1.1 Research methodology **Error! Bookmark not defined.**

Figure 3.1 Research methodology phases 23

Figure 3.2 Implementation phases 25

Figure 3.3 Sample of raw data 26

Figure 3.4 Sample of transformed data 27

Figure 3.5 Sample of Microsoft spell checking 29

Figure 3.6 Training and Testing representation used by NB classifier 30

Figure 4.1 Loading the data 34

Figure 4.2 Transformation task results 35

Figure 4.3 Feature representation 37

Figure 5.1 Performances of each feature using f-measure 46

Figure 5.2 performances of all pair combinations of features 48

Figure 5.3. Performances of all combinations of the features 49

© COPYRIGHT UPM

LIST OF ABBREVIATIONS

ASDG	Abstract System Dependence Graphs
DFT	Dynamic Feature Traces
FL	Feature Location
NB	Naïve Bayes
NLP	Natural Language Processing
SR	Software Reconnaissance
SVD	Singular Value Decomposition

© COPYRIGHT UPM

CHAPTER I

INTRODUCTION

1.1 INTRODUCTION

Software engineering is the process of analyzing software systems in order to improve the efficiency (Sjøberg et al. 2005). This process can be explained as supplying recommendation, illustration and providing reports for enhancing the performance of a particular system. To do so, a comprehensive analysis should be concentrated on the significant features shown in the source code of the system (Thomas 2011). Analyzing these features within the code provides valuable understanding of the intention of the code which facilitate the process of re-use and modification that would be performed on such code. One of the common concepts that are frequently used in any source code is the identifiers (e.g. string Name) (Enslen et al. 2009). Extracting such identifiers would offer a good opportunity to understand the headlines of the source code where the programmer declares all the objects that will be used in the system (e.g. student, employee, etc.) (Lawrie et al. 2007). In addition, the process of extracting identifiers has a significant impact on improving feature locations. Feature location aims to extract specific portion of the source code that typically correspond to the developer's query (Chen & Rajlich 2000).

Since the source code is written by the natural language, analyzing the source code can be done by using Natural Language Processing techniques. However, there are multiple differences between the regular text and the source code. In the source code, the multi-word identifiers are written without a space between them, instead several strategies can be used. First, it may be divided using special characters such as 'Employee-Name'

© COPYRIGHT UPM

or ‘Employee_Name’ (Enslen et al. 2009). Second, it may be written using ‘CamelCase’ approach, this approach aims to capitalize the first letter of the first words and the first letter of the second word without spacing (e.g. EmployeeName) (Lawrie & Binkley 2011a). Apart from the multi-word splitting problem, the identifiers in the source code may be written using abbreviations such as ‘Emp’ for ‘Employee’ (Feild et al. 2006). Hence, there is no unified or agreement mechanism to write the identifiers in the source code. This can make extracting such identifiers from the source code is a challenging task. Therefore, this study aims to propose an automatic approach for extracting identifiers from the source code. In particular, domain specific features will be developed that have the ability to utilize the characteristics of identifiers in the source code. After that, supervised machine learning technique will be used in order to classify the identifiers.

1.2 SIGNIFICANCE OF STUDY

With the dramatic evolution of software engineering, tremendous amount of software nowadays is being modified, changed and improved chronically. This continuous changing requires understandable developer who can treat the source code. The developer should know what the source code is intended to do by each included function. However, dealing with a large-scale source code would significantly hinder the process of modification by the user. Therefore, developer will tend to search manually on the desired portions that wanted to be modified. Meanwhile, the manual searching would be tedious and time consuming especially when there are thousands of lines. Therefore, feature location has been proposed for this purpose in order to facilitate the process of identifying the location of specific feature or identifier.

1.3 PROBLEM STATEMENT

Recently, information retrieval has been applied on software engineering applications in order to enhance the productivity of systems. This can be shown by allowing the developer to search within the source code for specific portions. This process called

© COPYRIGHT UPM

Feature Location. Such process depends mainly on the natural language processing techniques by utilizing the syntax and semantic of the words. The most important semantics that have been addressed is the identifiers. Such identifiers can provide a big picture of the whole source code. This can facilitate the process of feature location. However, unlike the traditional text processing, source code contains complex morphology where the multi-word identifiers are not divided using white space. Instead, the splitting depends on several mechanisms. It could be divided using special characters such as 'dash' (e.g. Student-ID) or 'underscore' (e.g. Student_ID). In addition, it could be divided using CamelCase where the first letters of both words are being capitalized such as 'StudentId'. The most complicated splitting mechanism is the multi-word that are separated neither by special characters nor by CamelCase. In such case, both words are in lowercase and attached to each other without a white space such as 'studentid'. Several approaches have been proposed to resolve such problem (Enslen et al. 2009; Feild et al. 2006; Lawrie & Binkley 2011a; Lawrie et al. 2007). Yet, there is a need for enhancement in terms of recall and precision. Such requirement of improvement is represented by using more robust features that have the ability to recognize the splitting words. Therefore, this study aims to identify an extension of features with machine learning techniques in order to separate the multi-word identifiers.

1.4 RESEARCH OBJECTIVES

The research objectives of this study are illustrated as follows:

1. To develop a set of features that have the ability to describe the cases of multi-word identifiers.

2. To implement a Naïve Bayes classifier to accommodate the splitting process based on the developed features.

© COPYRIGHT UPM

1.5 RESEARCH SCOPE

This study aims to propose a splitting method for the identifiers located in the source code. The type of analysis used to perform the separation is textual (more specifically lexical) in which multiple morphological features are being developed, then a Naïve Bayes classifier will utilize these features in order to conduct the separation. The source code that will be used in this study collected from the study of Enslen et al. (2009). Such data contains 9000 open source Java programs from SourceForge (https://sourceforge.net) which is a website consists of numerous Java projects with its source code.

1.6 RESEARCH METHODOLOGY

The research architecture contains four main stages as shown in Figure 1.1 including (i) problem identification, (ii) design, (iii) implementation and (iv) evaluation. The first stage which is problem identification aims to conduct a literature review for the field of feature location and narrowing the search into splitting identifier problem. This stage aims to identify ongoing and endure gaps in order to formulate the problem statement of this study. Design stage aims to investigate the existing techniques that have been used for the feature location and splitting identifiers. Such investigation aims to select an appropriate technique as an objective of this study. Implementation stage aims to carry out the proposed technique in the objective. This can be represented by collecting a dataset, applying the proposed method and observing the results. Evaluation stage aims to identify an appropriate evaluation metrics in order to assess the performance of the proposed method.

© COPYRIGHT UPM

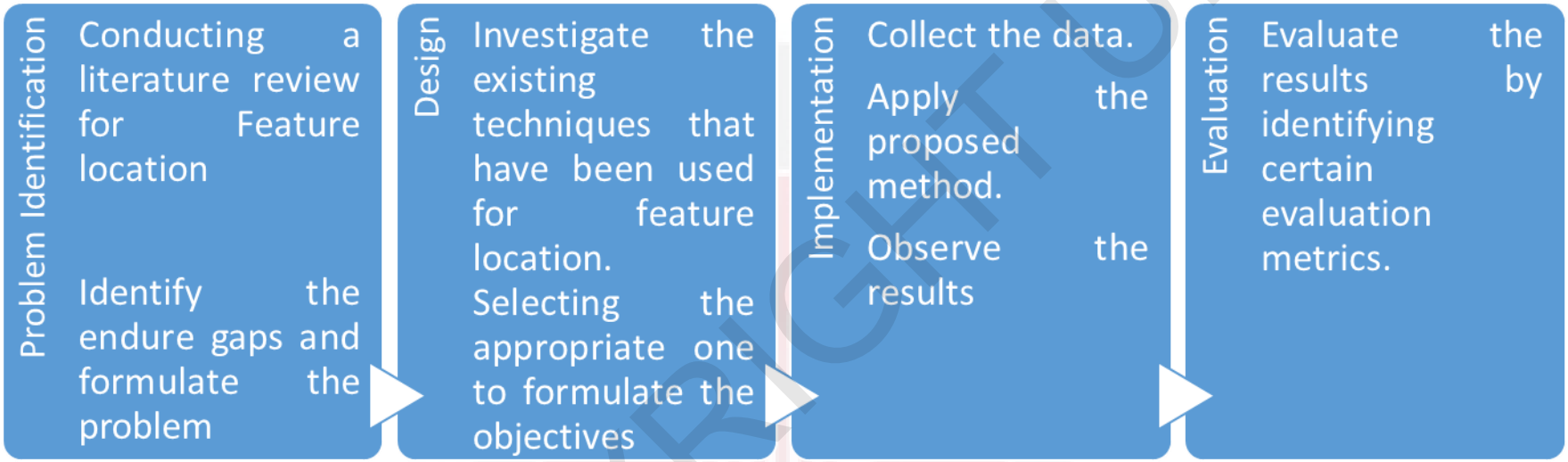

Figure 1.1 Research methodology

© COPYRIGHT UPM

1.7 THESIS ORGANIZATION

This research composed of five chapters that are being described as follows:

Chapter I provides the headlines of this research where the background of the study, problem statement and the objectives are being discussed.

Chapter II conducts a comprehensive literature review for the field of feature location by identifying the problem, impacted factors and techniques used for this problem. In addition, this chapter aims to concentrate on the splitting identifiers problem as a sub-task of feature location. Finally, this chapter will provide a critical analysis of the state of the art approaches.

Chapter III discusses the implementation of the proposed method by discussing the dataset used in the experiments, the transformation tasks, feature extraction, classification and evaluation.

Chapter IV discusses experiment setting in which the mechanism of attaining the results will be identified. This can be performed by determining all the parameters of each phase including transformation, feature extraction, classification and evaluation.

Chapter V analyzes the experimental results that have been obtained by the proposed method. Consequentially, the results will be analyzed technically in terms of the effectiveness.

Chapter VI provides the final conclusion of the research in which a summary of the whole thesis is being described. In addition, an emphasis of the research contribution is also provided. Finally, the future directions that could be inspired by this research will be described.

© COPYRIGHT UPM

1.8 SUMMARY

This chapter has provided the outline of the research in which the background of the study, problem statement, research objectives, research architecture are being described properly. Next chapter will discuss the literature review by describing the related work and their techniques in more details.

© COPYRIGHT UPM

REFERENCES

Alhindawi, N., Dragan, N., Collard, M. L. & Maletic, J. 2013. Improving feature location by enhancing source code with stereotypes. *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*, 300-309.

Amor, N. B., Benferhat, S. & Elouedi, Z. 2004. Naive bayes vs decision trees in intrusion detection systems. *Proceedings of the 2004 ACM symposium on Applied computing*, 420-424.

Bohnet, J., Voigt, S. & Doellner, J. 2008. Locating and understanding features of complex software systems by synchronizing time-, collaboration-and code-focused views on execution traces. *Program Comprehension, 2008. ICPC 2008. The 16th IEEE International Conference on*, 268-271.

Chen, K. & Rajlich, V. 2000. Case Study of Feature Location Using Dependence Graph. *IWPC*, 241-247.

Chen, K. & Rajlich, V. 2001. RIPPLES: tool for change in legacy software. *Proceedings of the IEEE International Conference on Software Maintenance (ICSM'01)*, 230.

Dit, B., Revelle, M., Gethers, M. & Poshyvanyk, D. 2013. Feature location in source code: a taxonomy and survey. *Journal of Software: Evolution and Process* 25(1). 53-95.

Eisenbarth, T., Koschke, R. & Simon, D. 2001a. Derivation of feature component maps by means of concept analysis. *Software Maintenance and Reengineering, 2001. Fifth European Conference on*, 176-179.

Eisenbarth, T., Koschke, R. & Simon, D. 2001b. Feature-driven program understanding using concept analysis of execution traces. *Program Comprehension, 2001. IWPC 2001. Proceedings. 9th International Workshop on*, 300-309.

Eisenberg, A. D. & De Volder, K. 2005. Dynamic feature traces: Finding features in unfamiliar code. *21st IEEE International Conference on Software Maintenance (ICSM'05)*, 337-346.

Enslen, E., Hill, E., Pollock, L. & Vijay-Shanker, K. 2009. Mining source code to automatically split identifiers for software analysis. *Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on*, 71-80.

Feild, H., Binkley, D. & Lawrie, D. 2006. An empirical comparison of techniques for extracting concept abbreviations from identifiers. *Proceedings of IASTED International Conference on Software Engineering and Applications (SEA'06)*,

© COPYRIGHT UPM

Freitag, D. 2000. Machine learning for information extraction in informal domains. *Machine learning* 39(2-3). 169-202.

Gay, G., Haiduc, S., Marcus, A. & Menzies, T. 2009. On the use of relevance feedback in IR-based concept location. *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*, 351-360.

Hill, E., Pollock, L. & Vijay-Shanker, K. 2009. Automatically capturing source code context of NL-queries for software maintenance and reuse. *Proceedings of the 31st International Conference on Software Engineering*, 232-242.

Huang, J., Lu, J. & Ling, C. X. 2003. Comparing naive Bayes, decision trees, and SVM with AUC and accuracy. *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, 553-556.

Koschke, R. & Quante, J. 2005. On dynamic feature location. *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, 86-95.

Lawrie, D. & Binkley, D. 2011a. Expanding identifiers to normalize source code vocabulary. *Software Maintenance (ICSM), 2011 27th IEEE International Conference on*, 113-122.

Lawrie, D. & Binkley, D. 2011b. Expanding identifiers to normalize source code vocabulary. *2011 27th IEEE International Conference on Software Maintenance (ICSM)*, 113-122.

Lawrie, D., Feild, H. & Binkley, D. 2007. Quantifying identifier quality: an analysis of trends. *Empirical Software Engineering* 12(4). 359-388.

Maletic, J. I. & Marcus, A. 2001. Supporting program comprehension using semantic and structural information. *Proceedings of the 23rd International Conference on Software Engineering*, 103-112.

Marcus, A., Sergeyev, A., Rajlich, V. & Maletic, J. I. 2004. An information retrieval approach to concept location in source code. *Reverse Engineering, 2004. Proceedings. 11th Working Conference on*, 214-223.

McCallum, A. & Nigam, K. 1998. A comparison of event models for naive bayes text classification. *AAAI-98 workshop on learning for text categorization*, 41-48.

Petrenko, M., Rajlich, V. & Vanciu, R. 2008. Partial domain comprehension in software evolution and maintenance. *Program Comprehension, 2008. ICPC 2008. The 16th IEEE International Conference on*, 13-22.

Poshyvanyk, D., Gueheneuc, Y.-G., Marcus, A., Antoniol, G. & Rajlich, V. 2007. Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval. *IEEE Transactions on Software Engineering* 33(6). 420-432.

© COPYRIGHT UPM

Poshyvanyk, D., Marcus, A., Dong, Y. & Sergeyev, A. 2005. IRiSS-A Source Code Exploration Tool. *ICSM (Industrial and Tool Volume)*, 69-72.

Robillard, M. P. & Murphy, G. C. 2002. Concern graphs: finding and describing concerns using structural program dependencies. *Proceedings of the 24th international conference on Software engineering*, 406-416.

Robillard, M. P. & Murphy, G. C. 2007. Representing concerns in source code. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 16(1). 3.

Shepherd, D., Fry, Z. P., Hill, E., Pollock, L. & Vijay-Shanker, K. 2007. Using natural language program analysis to locate and understand action-oriented concerns. *Proceedings of the 6th international conference on Aspect-oriented software development*, 212-224.

Sjøberg, D. I., Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanovic, A., Liborg, N.-K. & Rekdal, A. C. 2005. A survey of controlled experiments in software engineering. *Software Engineering, IEEE Transactions on* 31(9). 733-753.

Tan, S., Cheng, X., Wang, Y. & Xu, H. 2009. Adapting naive bayes to domain adaptation for sentiment analysis. *Advances in Information Retrieval*. 337-349. Springer.

Thomas, S. W. 2011. Mining software repositories using topic models. *Proceedings of the 33rd International Conference on Software Engineering*, 1138-1139.

Venkatapathy, S. & Joshi, A. K. 2005. Measuring the relative compositionality of verb-noun (VN) collocations by integrating features. *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 899-906.

Wilde, N., Gomez, J. A., Gust, T. & Strasburg, D. 1992. Locating user functionality in old code. *Proceedings Conference on Software Maintenance 1992*, 200-205.

Wilde, N. & Scully, M. C. 1995. Software reconnaissance: mapping program features to code. *Journal of Software Maintenance: Research and Practice* 7(1). 49-62.

Wilson, L. A. 2010. Using ontology fragments in concept location. *Software Maintenance (ICSM), 2010 IEEE International Conference on*, 1-2.

© COPYRIGHT UPM