Named Entity Recognition for Automated Test Case Generation

Guruvayur Mahalakshmi[1], Vani Vijayan[2], and Betina Antony[1]
[1]Department of Computer Science and Engineering, Anna University, India
[2]Department of Information Technology, Easwari Engineering College, India

Abstract: *Testing is the process of evaluating a software or hardware against its requirement specification. It helps to verify and grade a given system. Recent emphasis on Test Driven Development (TDD) has increased the need for testing from the early stages of software development. System test cases can be obtained from a number of user specifications such as functional requirements; UML diagrams and use case specification. This paper focuses on automating the test process from the early stages of requirement elicitation in the development of software. It describes a semi-supervised technique to generate test cases by identifying named entities in the given set of use cases. The named entities along with flow listing of the use cases serves as the source for scenario matrix from which a number of test cases can be obtained for a given scenario. The Named Entity Recognizer (NER) is trained by a set of features extracted from the use cases. The automated generation of entity list was found to increase the efficiency of the overall system.*

Keywords: *Named entity recognition, test case generation, scenario matrix, decision table december.*

Received July 14, 2014; accepted December 16, 2014

1. Introduction

Software testing plays an important role in estimating reliability of a system, assuring software quality and for verifying and validating the functionalities of software. As the complexity and size of software grow, the time and effort required to do effective testing increase. Studies indicate that more than 50% of the cost of software development is devoted to testing [7]. The main concern in software testing is the generation of test cases. Designing and execution of test cases for any software is highly time consuming and labour intensive. The increasing size of software only escalates the complexity of creating test cases. Hence automation of test cases has become an inevitable process in the course of software testing.

There are essentially two main approaches to automatic design of test cases. One approach attempts to design test cases from requirement and design specification and the other from code. Since generation of test cases from code is cumbersome, the alternate approach is given more importance in research point of view. The process of generating tests from design will often help the test engineer to discover problems with design itself. If this step is done early, the problems can be eliminated early, saving time and resources. Generating tests during design also allows testing activities to be shifted to an earlier part of the development process, allowing for more effective planning of test cases.Another advantage is that the test data is independent of any particular implementation. Generating test cases at early stages is a good supplement to testing. These test cases can be tested at later stages coding. Though testing essentially starts at the design phase, the error in understanding or design can be carried on to consecutive phases. Hence it is essential to commence the examination modules right from the requirement phase. This leads to a more stable system covering aspects of both user specification and developer understanding.

In a software development project, use cases define system software requirements. Use case development begins early on, so real use cases for key product functionality are available in early iterations [21]. A use case fully describes the sequence of actions performed by a system to provide an observable result of value to a person or another system using the product under development. Use cases tell the customer what to expect, the developer what to code, the technical writer what to document, and the tester what to test. Thus use cases can be deployed effectively in the development of test cases. These test cases identify and communicate the conditions that will be implemented in test and are necessary to verify successful and acceptable implementation of the product requirements. They are all about making sure that the product fulfils the requirements of the system.

Named-Entity Recognition (NER) also known as entity identification and entity extraction is a subtask of information extraction that seeks to locate and classify atomic elements in text into predefined categories. The Named Entities refer to one or more rigid designators which includes proper nouns as well as certain kinds of natural terms. The ability of recognizing previously unknown entities is an essential part of NERC systems [36]. These abilities depend on recognizing and classifying based on distinctive features associated with positive and negative examples.

In this paper, the authors provide a semi-supervised method of extracting named entities from use cases. Different features like orthographic and semantic features are extracted from the dataset along with general features such as POS tags and word frequency. These entities along with flow specification can be used to generate test cases from the given set of use cases.

2. Related Work

Data mining has found its application in a wide range of fields [26] such as data modelling like language modelling [9], XML document modelling [32] meta learning [41]; knowledge discovery [48], Knowledge Management [14]; neural networks [54] medical system [24], CRM [37], web education [43] etc. A data mining project has a list of phases such as business understanding, data understanding, data preparation, modelling and deployment. A number of data mining techniques [27] are applied in various applications. Some of these techniques include clustering, classification, pattern matching, data summarization and deviation detection [17].

2.1. Intelligent Approaches for Test Case Generation

Generating Test cases via machine learning techniques [5] is two-decade old. Applying metaheuristic search techniques and genetic algorithms [53] have been extensively used to automate the process of generating test cases, and thus providing solutions for a more cost-effective testing process. SBST is a branch of Search-Based Software Engineering (SBSE) [18], in which optimisation algorithms are used to automate the search for test data that maximises the achievement of test goals, while minimising testing costs. SBST has been applied to a wide variety of testing goals including structural [19, 30, 31, 33, 47], functional [49], non-functional [50] and state-based properties [11]. Lakhotia *et al.* [25] used a local search to augment the Pex DSE-based testing tool from Microsoft, while [45] augmented 'standard' constraint solving with a Particle Swarm optimiser to improve the performance of Symbolic PathFinder. Ali *et al*. [1] provide a good review of the existing attempts to test case generation. DeSantiago and Vijaykumar [10] presented a methodology, SOLIMVA, which aims at model-based test case generation considering NL requirements deliverables. The methodology is supported by a Semantic Translation Model in which, among other features, a word sense disambiguation method helps in the translation process. Application of Constraint Logic Programming to Test Case Generation is also experimented [34].

2.2. Mining in Software Engineering

Software engineering is a wide domain packed with textual artifacts written in natural language such as requirement specification documents, design documents, code, execution logs, test suites and bug logs. Various sources of software engineering data include documentation, SCM documents, Source code, issues and bugs database and mailing list [22]. Mining of these units is one of the key requisite for automating the activities of software development. Mining activities may include tracing of requirements; retrieval of components from a repository; extracting functional and non functional attributes; conversion of design to action, identify and eradicate bugs etc [20]. Text mining done in software documents have also led to ontology building. Here the software data documents are mined at semantic level and the extracted information is used in automated population of documentation ontology [51].

2.3. Mining Techniques in Software Testing

Testing plays a major role software development process. It is but natural to try and automate this process to optimise the development. Before actually generating test cases, researches were carried out to study software behaviour. An active learning technique was suggested by [8] where a Morkov classifier was built and trained to predict the behaviour of program execution. Data mining techniques such as cluster analysis played a major role in operation-based testing [12]. The filtering of these clusters based on certain metrics improved the efficiency of the system to identify more failures in the execution profile.

Works on automating the process of software testing started as early as in 2000's. Initial works included automated input-output analysis of data-driven software systems where an Info-Fuzzy Network (IFN) was constructed [26]. The network was employed to automatically generate non-redundant set of test cases for execution data with the help of Legacy systems and random test generator. A number of different techniques to generated test cases from functional requirements that are presented in natural language were studied [16]. Test cases can also be generated from state charts drawn using SRS. Here a rule based classifier is used to identify functional and non-functional requirements which are used to produce state diagrams. These in turn are used to produce test suites on which DM techniques such as association, clustering etc are done for optimization [40].

Test cases can be generated from dynamic models such as control flow graphs and sequence diagrams by considering full predicate coverage criteria [46]. The test cases thus formed can be used to identify object interaction and operational faults. The test cases produced can be prioritized by k-means clusters and code complexity metrics [2].

2.4. Named Entity Recognition

Named Entity Recognition (NER) is one of the subtasks of information extraction. NER aims at locating the Named Entities (NE's) in a given context and classifying them into different categories. NER finds its application in text summarization, machine learning, information retrieval etc. [15]. NER is employed in a number of applications some of which are shown in table 1.

Table 1. Applications of NER.

Application	Method used	Author
Newswire	MEM model	Mikheev *et al.* [35]
Multi-lingual	Semi-CRF model	Kim *et al.* [23]
	Gazeetter based classification	Nothman *et al.* [38]
Tweet	LabeledLDA	Ritter *et al.* [42]
	KNN with Linear CRF	Liu *et al.* [28]
Biomedicine	Semi-CRF model	Yang and Zhou [52]
	SVM model	Song *et al.* [44]
	MEM model	Patrick and Wang [39]
	SVM with HMM model	Atkinson and Bull [3]
Diverse domain	General Architecture for Text Engineering (GATE)	Maynard *et al.* [29]
	SVM with HMM model	Etter *et al.* [13]

3. System Description

The process of generating test cases from use cases follows a given set of tasks. Flow analysis and scenario listing are two main components needed for generating test case matrix as shown in Figure 1.

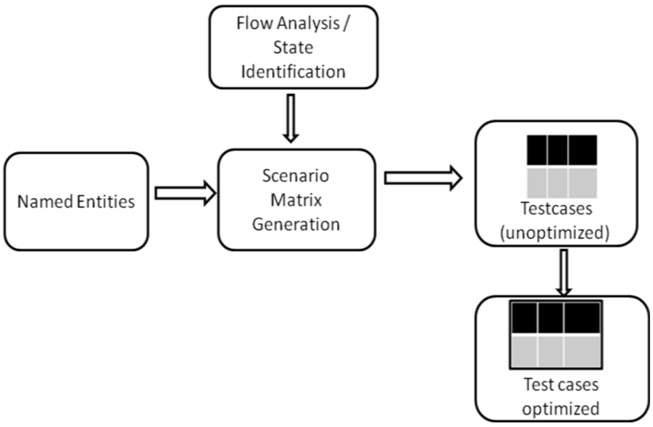
Flow Analysis / State Identification
Named Entities
Scenario Matrix Generation
Testcases (unoptimized)
Test cases optimized

Figure 1. Test case generation.

3.1. Scenario Matrix Generation

Scenario matrix plays a major role in test case generation from use cases. The creation of scenario matrix is done in three steps:

- Alternate flow identification
- Decision table construction
- Scenario matrix generation

3.1.1. Alternate Flow Identification

Alternate flows are a conditional set of steps that are an alternative to one or more steps in another flow after which the use case continues to pursue its goal. The alternate flow can be option flow, exception flow or recovery flow. These alternate flows are identified from the given set of use cases. Redundant flows are eliminated and the final set of alternate flows is saved. These form the basis for the scenario matrix.

3.1.2. Decision Table Construction

The decision table is a multidimensional data structure which gives information about the set of characteristics that lead to the success or failure of a scenario. The table contains a list of scenarios and the set of entities used in that application. The flow steps in each scenario are mapped against the entity in it. Finally the success or failure of the usecase is denoted in the result section. The decision table plays an important role in test case generation. That is the nature of input data and system response for each test scenario can be obtained. The decision table however does not gives details about the required sequencing of flows or which data to test for.

3.1.3. Scenario Matrix Generation

The scenario matrix is a deductive method useful for constructing scenarios in volatile and uncertain situations. The matrix describes the alternate flows taken in each scenario when the basic flow fails. The final state of the system is also indicated. The change in flow depends on the input action performed.

3.2. Named Entity Recognition from Use Cases

The domain related named entities that are used in the construction of decision table can be generated from the uses cases by machine learning techniques. The NE's thus found are saved in the NE dictionary in their domain which can be referred in future for other set of use cases. However the Named Entity Recognition (NER) module is domain independent. Named Entity Recognition uses four different features as depicted in Figure 2, namely n-gram frequency, term frequency scoring, gazetteer reference and certain minor features. Using these features, the use cases are trained by machine learning algorithms.

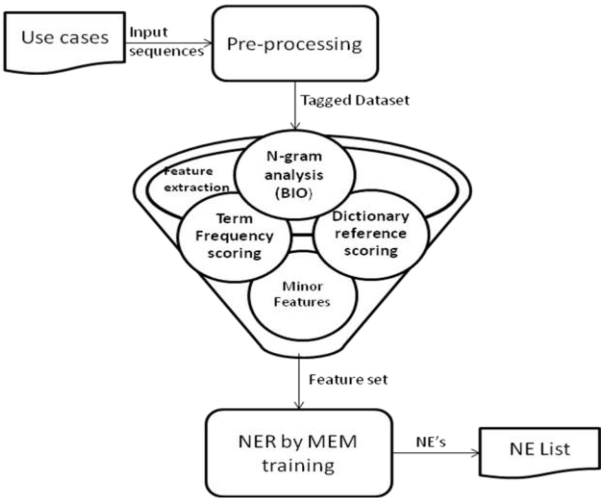

Figure 2. Named entity recognition for use cases.

3.2.1. Feature Set

When it comes to NER, a number of features such as contextual, lexical, morphological and shallow syntactic features play a prominent role [4]. For training the model both orthographic and semantic features are extracted from the data set.

- *N-gram frequency analysis*: N-grams are two or more adjacent elements in a string of tokens that represent a single word. They are used to provide better representation of document than Bag-Of-Words (BOW). N-grams provide conditional probability of a token given its preceding and succeeding token (BIO tags). The analysis done uses Enhances-SVM approach to identify frequency of n-grams where in addition to the frequencies, the positions of the terms are also considered [6].
- *Term frequency scoring*: For term scoring, tf-idf is used which serves as a weighing factor for information retrieval and central tool for scoring and ranking frequently occurring words and a document relevant to a query.
- *Dictionary reference scoring:* A simple way to guess the sense of a particular phrase is to look it up in a local dictionary. Look-up systems with large entity lists work pretty well if the entities are not ambiguous. Princeton's wordnet[1] provides various details such as the sense of a word and their meaning to resolve ambiguity to an extent. It also provides various other details such as synonyms, antonyms, hyper and hyponyms for the given term and also the domain to which they may belong. The dictionary provides an additional grade called familiarity which has values such as very familiar, familiar, common, uncommon, rare and very rare based on the number of forms the word takes for a given sense. Thus the wordnet reference scoring calculates a value based on the number of senses. The terms with a positive score have higher possibility of being a named entity.
- *Minor features:* Other minor features include identifying words with special characters (eg. '_','*' etc.) and capitalized words.

3.2.2. Training by Machine Learning

The features thus obtained are used to train an NER identification model. The authors employ Maximum Entropy Model (MEM) to train the dataset. MEM is also known as multinominal logistic regression model that assigns conditional probabilities on the hidden structures in the given data. This model assigns to each feature a weight. A positive weight indicates the configuration is likely to be correct whereas the negative weight indicates the configuration is possibly incorrect.

3.3. Test Case Generation

The generation of test cases involves identifying test conditions or data elements for a given functionality, identifying all possible scenarios in the given operation and finally identifying data element states in each scenario and their corresponding output. The generated test case contains information such as the input to provided, expected result and actual result for each scenario in a given functionality. The system uses finite-state automation technique where each input sequence transit to finite number of states.

3.3.1. Identify Data Elements

The data elements or the data conditions are the entities that are used in a use case. They can take up different values which determine the success or failure of the system. These entities are obtained by matching the input data against a domain based NE dictionary.

3.3.2. Identify Possible Scenarios

Dividing the use case into set of scenarios enhances the number of test cases generated. Each scenario is formed by considering different values for the data elements and the corresponding result of the system. The results are listed in the form of a scenario matrix. The different states of data elements are termed as alternate flows. Thus a scenario matrix generated above contains list of scenarios, their set of basic and alternate flows and the nature of the output for that scenario.

3.3.3. Identify State of Data Elements and Corresponding Output

From the data elements and the scenarios identified, the last step in creating use case is providing values for the data elements. As shown in the finite-state diagram, the outcome of a given scenario depends upon the value of these data elements or conditions. The possible values for these elements are found from the given use case sequences as represented in Figure 3.The test cases are written for all possibilities where each case shows the values taken by the data elements and their expected output.

Tabulating the data obtained from the above steps will yield the set of test cases for the given functionality of the system under study. These test cases can further be optimized by reducing redundant data and identifying missing test conditions.

[1] http://wordnet.princeton.edu/

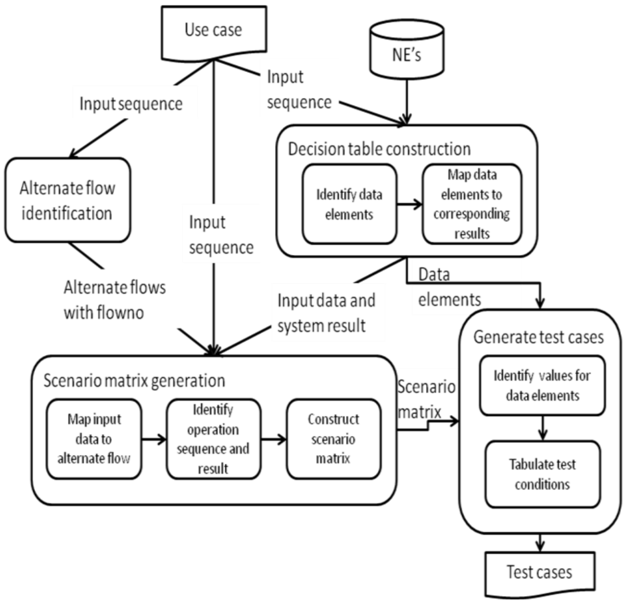
Use case
NE's
Input sequence
Input sequence
Decision table construction
Identify data elements
Map data elements to corresponding results
Alternate flow identification
Input sequence
Alternate flows with flowno
Input data and system result
Data elements
Generate test cases
Identify values for data elements
Scenario matrix generation
Map input data to alternate flow
Identify operation sequence and result
Construct scenario matrix
Scenario matrix
Tabulate test conditions
Test cases

Figure 3. Detailed design of test case generation.

4. Results and Discussion

The experiment was conducted in two stages; Named entity recognition and Scenario matrix construction. The testing and discussion mainly focuses on the NER section that serves as the baseline for scenario matrix and decision table.

4.1. Dataset

The input corpus for training had 80 domain independent use cases. Each use case was described in detail and was expressed in Jacobson template. From these use cases 354 constituent entities were obtained among which 262 were positive and 92 were negative. The test set was divided into 2 sets: set 1 contained 10 domain dependent use cases and 87 constituent terms; set 2 contained 49 domain independent use cases and 237 constituent terms. The details of the datasets used are depicted in table 2.

Table 2. Domain details of dataset for classification.

Dataset	No of use cases	No of Domain	List of Domain
Training data	80	3	Hotel management Stock maintenance Weather forecast
Test set 1 (domain dependent)	10	1	Stock maintenance
Test set 2 (domain independent)	49	8	Hotel management Employee database management Restaurant service Reservation system Logistics management Healthcare Security and maintenance Inventory management

4.2. Feature Extraction

The four features considered here are N-grams frequency count, term frequency scoring, sense-based scoring and minor feature scores. These features are identified separately and are integrated to form the training set. Here the number of terms in N-gram is limited to 2 and their frequency is calculated using E-SVM method [42]. For term frequency scoring, *tf-idf* value was calculated for frequently occurring terms in a document and the value for a term was normalized[2]. For sense-based scoring, the number of senses for a given constituent term was found. A weighted value was given for each sense (noun, verb, adjective and adverb) and a final score was calculated.

4.3. NER Classifier

The scores thus calculated were used to train a MEM classifier. In the Figure 4, the classifier was trained using 354 constituent terms. The model built had an f-measure of 70.4. The model had high TP-rate (recall) of about 94.5% for positive classification. This model was then test with the 2 different test sets. The classifier produced good results for both the sets. The results of the classifier on various dataset are shown in Table 3.

Table 3. Evaluation of NER classifer.

Dataset	Precision	Recall	F-score
Training data	0.716	0.751	0.704
Test set 1	0.714	0.968	0.822
Test set 2	0.675	0.934	0.783

Two minor challenges were encountered during the testing of this model. Firstly only noun and noun phrases were considered constituent terms for classification. Here the incorrect tagging of POS tagger is carried on to the next stages. For example, “displays_NNS” is a verb tagged as noun. Secondly sense based scoring was mainly used to avoid ambiguity of the sense of the word in the given context. This however was not completely successful. For instance, the word “case” has a familiarity rating of very familiar for noun sense and rare familiarity rate in verb sense. Based on the sense-score, the word falls into the category of named entity but the actual word is not an entity. Hence additional features may have to be included to improve the efficiency of the system. The minor features considered in the model are capitalization and specific set of special characters. Being positive in most of the cases, they also contributed to incorrect tagging. For example, capitalization at the beginning of a sentence need not necessarily be an entity. The greatest challenge of this model was high specificity rate (nearly 0.3 for test set 2). The model lacked few discriminating features to correctly reject non-entity terms.

[2] Average value of the score for a term in all possible documents was calculated.

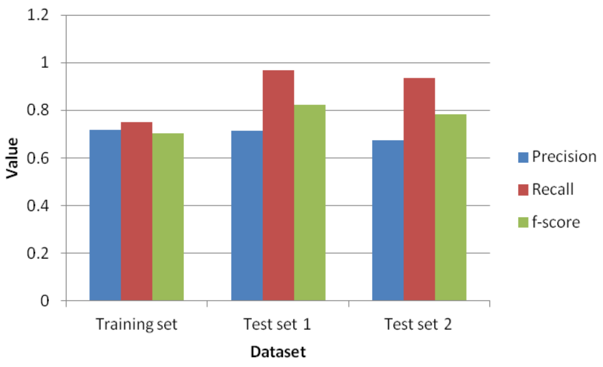

Figure 4. Evaluation of NER classifier.

4.4. Test Case Using Named Entities

The named entities thus obtained are used by the test case generation module. In this module, the list of entities indicates the data elements of the system. When an entity is identified in the use case, possible states taken by the entity is obtained. For example, an entity ‘password’ can be valid, invalid, correct, incorrect, empty etc. Hence a test statement is formed to check each of these different states. Also the scenario matrix is obtained which gives flow of events for each scenario. Test condition is identified for each of these events. Finally all the test conditions for a scenario are tabulated to form a test suit for the given scenario.

The steps can be explained using the example of login use case (Algorithm 1). It is to be noted that the list of named entities form the basis for the artifacts created. Without them, the data elements cannot be identified. Hence automation of this step has a greater impact on the test case generation. The states of the data element are obtained from the given set of use cases. Hence it is very essential to identify all possible alternate flows for a given module. Thus this method of testing from the requirement elicitation phase ensures that a proper system is built from the first phase of development. The test cases thus produced for login use case are shown in Table 4. This method is extended to different use cases from a wide range of domain.

Table 4. Test cases for Login use case.

S no	Username	Password	Verification code	Expected output
1	Correct	Correct	N/A	Success
2	Incorrect	N/A	N/A	Re-enter username
3	Correct	Incorrect	N/A	Re-enter password
4	Correct	Forgot	Correct	Success
5	Correct	Forgot	Incorrect	Failure
6	Null	Null	N/A	New user

Example 1. Test Condition from Use Case

```
Named Entities:
        Username, Password, Verification code, Register,
Result
Flow of events:
        Correct user name
        Correct password
        Incorrect user name
        Incorrect password
        Forgot password
        Correct verification code
        Incorrect verification code
Result: Success, Failure.
No of Scenarios: 6
Scenario 1:
        Correct Username -> Correct Password -> Success
Test condition for scenario 1:
        if(username == correct)
        then
                if(password == correct)
                then
                        success;
                else
                then
                        error(not expected value);
                end if
        else
        then
                error(not expected value);
        end if
```

5. Conclusions

This paper provides a break through attempt in employing a very effective data mining technique to automate the process of generating test cases during the initial stage of software development life cycle. A domain independent NER system was built to optimize a test case generation system. Though the system had to surf through a voluminous list of entities when scanning individual use cases, the effort is less when compared to manually identifying tagging entity names from individual use cases. Also error in tagging shall be avoided. The system however can be expanded with different sets of feature that may overcome the drawbacks of ambiguity and improve the efficiency of classification. The NER system can also be used in different stages of software development.

References

[1] Ali S., Briand L., Hemmati H., and Panesar-Walawege R., “A Systematic Review of the Application and Empirical Investigation of Search-Based Test Case Generation,” *IEEE Transactions* on Software Engineering, vol. 36, no. 6, pp. 742-762, 2010.

[2] Arafeen M. and Do H., “Test Case Prioritization Using Requirements-Based Clustering,” *in Proceedings of IEEE Sixth International Conference on Software Testing, Verification and Validation*, Luxembourg, pp. 312-321, 2013.

[3] Atkinson J. and Bull V. “A Multi-Strategy Approach to Biological Named Entity Recognition,” *Expert Systems with Applications*, vol. 39, no. 17, pp. 12968-12974, 2012.

[4] Benajiba Y., Diab M., and Rosso P., “Using Language Independent and Language Specific Features to Enhance Arabic Named Entity Recognition,” *The International Arab journal of*

Information Technology, vol. 6, no. 5, pp. 463-471, 2009.

[5] Bergadano F., "Test Case Generation by Means of Learning Techniques," *in Proceedings of the 1st ACM SIGSOFT symposium on Foundations of Software Engineering*, Los Angeles, pp. 149-162, 1993.

[6] Bhakkad A., Dharamadhikari S-C., and Kulkarni P., "Efficient Approach to Find Bigram Frequency in Text Document Using EVSM," *International Journal of Computer Applications*, vol. 68, no. 19, pp. 9-11, 2013.

[7] Binder R., *Testing Object-Oriented Systems: Models, Patterns, and Tools*, Addison-Wesley Professional, 2000.

[8] Bowring J., Rehg J., and Harrold M., "Active Learning for Automatic Classification of Software Behavior," *ACM SIGSOFT Software Engineering Notes*, vol. 29, no. 4, pp. 195-205, 2004.

[9] Chen Y., Constructing Language Model by Using Data Mining Technique, Theses, The University of Hong Kong, 2004.

[10] DeSantiago V. and Vijaykumar N., "Generating Model-Based Test Cases from Natural Language Requirements for Space application software," *Software Quality Journal*, vol. 20, no. 1, pp. 77-143, 2012.

[11] Derderian K., Hierons R., Harman M., and Guo Q., "Automated Unique Input Output Sequence Generation for Conformance Testing of FSMs," *The computer Journal*, vol. 49, no. 3, pp. 331-344, 2006.

[12] Dickinson W., Leon D., and Fodgurski A., "Finding Failures by Cluster Analysis of Execution Profiles," *in Proceedings of Software Engineering*, Toronto, pp. 339-348, 2001.

[13] Etter D., Ferraro F., Cotterell R., and Buzek O., "Nerit: Named Entity Recognition for Informal Text," *Technical Report 11, Human Language Technology Center of Excellence*, 2013.

[14] Fesharaki M., Shirazi H., and Bakhshi A., "A Knowledge Acquisition from Database of Information Management and Documentation Softwares by Data Mining Techniques," *Information Sciences and Technology*, vol. 26, no. 2, pp. 259-283, 2011.

[15] Grishman R. Information Extraction: Techniques and Challenges, *in Proceedings of International Summer School on Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*, Frascati, pp. 10-27 1997.

[16] Gutiérrez J., Escalona M., Mejías M., and Torres J., "Generation of Test Cases from Functional Requirements," *A survey in 4ş Workshop on System Testing and Validation*, 2006.

[17] Halkidi M., Spinellis D., Tsatsaronis G., and Vazirgiannis M., "Data Mining in Software Engineering," *Intelligent Data Analysis*, vol. 15, no. 3, pp. 413-441, 2011.

[18] Harman M., Mansouri S., and Zhang Y., "Search-Based Software Engineering: Trends, Techniques and Applications," *ACM Computing Surveys*, vol. 45, no. 1, pp. 1-64, 2012.

[19] Harman M. and McMinn P., "A Theoretical and Empirical Study of Search Based Testing: Local, Global and Hybrid Search," *IEEE Transactions on Software Engineering*, vol. 36, no. 2, pp. 226-247, 2010.

[20] Hayes J., Dekhtyar A., and Sundaram S., "Text Mining for Software Engineering: How Analyst Feedback Impacts Final Results," *In ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4, pp. 1-5, 2005.

[21] Heumann J., *Generating Test Cases from Use Cases*, the Rational Edge, 2001.

[22] Ismail N., Ibrahim, R., and Ibrahi N., "Automatic Generation of Test Cases from Use-Case Diagram," *in International Conference on Electrical Engineering and Informatics*, Teknologi, pp. 17-19, 2007.

[23] Kim S., Toutanova K., and Yu H., "Multilingual Named Entity Recognition Using Parallel Data and Metadata from Wikipedia," *in Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, JejuIslan, pp. 694-702, 2012.

[24] Kusiak A., Kernstine K., Kern J., Melaughlin K., and Tseng T., "Data Mining: Medical and Engineering Case Studies," *in Proceedings of the Industrial Engineering Research Conference*, Cleveland, pp. 1-7, 2000.

[25] Lakhotia K., Tillmann N., Harman M., and de Halleux J., "Flopsy-Search-Based Floating Point Constraint Solving for Symbolic Execution," *in Proceedings of the 23rd IFIP International Conference on Testing Software and Systems*, Natal, pp. 142-157, 2010.

[26] Last M., Friedman M., and Kandel A., "The Data Mining Approach to Automated Software Testing," *in Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, pp. 388-396, 2003.

[27] Liao S., Chu P., and Hsiao P., "Data Mining Techniques and Applications-A Decade Review from 2000 to 2011," *Expert Systems with Applications*, vol. 39, no.12, pp. 11303-11311, 2012.

[28] Liu X., Zhang S., Wei F., and Zhou M., "Recognizing Named Entities in Tweets," *in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics:*

Human Language Technologies, Portland, pp. 359-367, 2011.

[29] Maynard D., Tablan V., and Ursu C., "Named Entity Recognition from Diverse Text Types," *in Proceedings of the Recent Advances in Natural Language Processing*, TzigovChark, pp. 257-274, 2001.

[30] McMinn P., Harman M., Hassoun Y., Lakhotia K., and Wegener J., "Input Domain Reduction Through Irrelevant Variable Removal and its Effect on Local, Global and Hybrid search-based Structural Test Data Generation," *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 453-477, 2012.

[31] McMinn P., Shahbaz M., and Stevenson M., "Search-Based Test Input Generation for String Data Types Using the Results of Web Queries," *in Proceedings of the 5th International Conference on Software Testing Verification and Validation*, Montreal, 2012.

[32] Mei D. and ZhangX., "Data Mining Sechniques for Structure of Single XML Document," *Journal of Petrochemical Universities*, vol. 20, no. 1, pp. 94-98, 2007.

[33] Michael C., McGraw G., and Schatz M., "Generating Software Test Data by Evolution," *IEEE Transactions on Software Engineering*, vol. 27, no. 12, pp. 1085-1110, 2001.

[34] Miguel G., Albert E., and Puebla G., "Test Case Generation for Object-Oriented Imperative Languages in Clp," *Theory and Practice of Logic Programming*, vol. 10, no. 4-6, pp. 659-674, 2010.

[35] Mikheev A., Moens M., and Grover C., "Named Entity Recognition Without Gazetteers," *in Proceedings of the 9th Conference on European Chapter of the Association for Computational Linguistics*, Bergen, pp. 1-8, 1999.

[36] Nadeau D. and Sekine S., "A Survey of Named Entity Recognition and Classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3-26, 2007.

[37] Ngai E., Xiu L., and Chau D., "Application of Data Mining Techniques in Customer Relationship Management: A Literature Review and Classification," *Expert Systems With Applications*, vol. 36, no. 2, pp. 2592-2602, 2009.

[38] Nothman J., Ringland N., Radford W., Murphy T., and Curran J., "Learning Multilingual Named Entity Recognition from Wikipedia," *Artificial Intelligence*, vol. 194, pp. 151-175, 2013.

[39] Patrick J. and Wang Y., "Biomedical Named Entity Recognition System," *in Proceedings of the 10th Australasian Document Computing Symposium*, Sydney, 2005.

[40] Raamesh L. and Uma G., "Knowledge Mining of Test Case System," *International Journal on Computer Science and Engineering*, vol. 2, no. 1, pp. 69-73, 2010.

[41] Radosavljevic V., Vucetic S., and Obradovic Z., "A Data-Mining Technique for Aerosol Retrieval Across Multiple Accuracy Measures," *IEEE Geoscience and Remote Sensing Letters*, vol. 7, no. 2, pp. 411-415, 2010.

[42] Ritteri A., Clarki S., and Etzioni O., "Named Entity Recognition in Tweets: An Experimental Study," *in Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Edinburgh, pp. 1524-1534, 2011.

[43] Romero C. and Ventura S., "Educational Data Mining: a Review of the State of the Art, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews*), vol. 40, no. 6, pp. 601-618, 2010.

[44] Song Y., Yi E., Kim E., Lee G., and Park S., "POSBIOTM-NER: a Machine Learning Approach for Bio-Named Entity Recognition," *in Proceedings the EMBO Workshop on Critical Assessment of Text Mining Methods in Molecular Biology*, 2004.

[45] Souza M., Borge M., D'Amorim M., and Pasareanu C., "CORAL: Solving Complex Constraints for Symbolic Pathfinder," *in Proceedings of Third International Symposium NASA Formal Methods*, Pasadena, pp. 359-374, 2011.

[46] Swain S., Mohapatra D., and Mall R., "Test Case Generation Based on Use Case and Sequence Diagram," *International Journal of Software Engineering*, vol. 3, no. 2, pp. 21-52, 2010.

[47] Tonella P., "Evolutionary Testing of Classes," *in Proceedings of ACM SIGSOFT International Symposium on Software Testing and Analysis*, Boston, pp. 119-128, 2004.

[48] Wasan S., Bhatnagar V., and Kaur H., "The Impact of Data Mining Techniques on Medical Diagnostics," *Data Science Journal*, vol. 5, pp. 119-126, 2006.

[49] Wegener J. and Buhler O., "Evaluation of Different Fitness Functions for the Evolutionary Testing of an Autonomous Parking System," *in Proceedings of Genetic and Evolutionary Computation Conference*, Seattle, pp. 1400-1412, 2004.

[50] Wegener J. and Grochtmann M., "Verifying Timing Constraints of Real-Time Systems by Means of Evolutionary Testing," *Real-Time Systems*, vol. 15, no. 3, pp. 275 -298, 1998.

[51] Witte R., Li Q., Zhang Y., and Rilling J., "Ontological Text Mining of Software Documents," *in Proceedings of International Conference on Application of Natural Language to Information Systems*, Paris, pp. 168-180, 2007.

[52] Yang L. and Zhou Y., "Exploring Feature Sets for Two-Phase Biomedical Named Entity Recognition Using Semi-CRFs," *Knowledge and Information Systems*, vol. 40, no. 2, pp. 1-15, 2014.

[53] Yuehua D. and Jidong P., "Automatic Generation of Software Test Cases Based on Improved Genetic Algorithm," *in Proceedings of International Conference on Multimedia Technology*, Hangzhou, pp. 227-230, 2011.

[54] Zhang C. and Marquez J., "Approximation of Minimal Cut sets for a Flow Network Via Evolutionary Optimization and Data Mining Techniques," *International Journal of Performability Engineering*, vol. 7, no. 1, pp. 21-31, 2011.

Guruvayur Mahalakshmi is an Assistant Professor (Senior Grade) in the Department of Computer Science and Engineering, College of Engineering, Anna University, Chennai. She completed her B.E. (Computer Science and Engineering) from R.V.S. College of Engineering and Technology, Dindigul and M.E. (Computer Science and Engineering) and Ph.D. from College of Engineering, Anna University, Chennai. She has numerous international journal and conference publications to her credit. She is also the author of Tamil Edition of B.E. course - text books - Fundamentals of Computing and Computer Practice of Anna University. She has authored many book chapters and derives 100+ citations to her credit. Her research interests include Reasoning, Knowledge Sharing and representation, Text Mining, Social Network Analysis, bibliometrics, and Natural Language Computing.

Vani Vijayan is a Senior Assistant Professor in Department of Information Technology in Easwari Engineering College, Anna University, Chennai, Tamilnadu. She completed her Masters from Anna University in 2009 in Computer Science and Engineering and Bachelors from Bharathiar University, Coimbatore, TamilNadu in 2002 in Information Technology. She is currently working towards pursuing Ph.D.degree from Anna University, Chennai in Faculty of Information and Communication Engineering, registered in July 2010. She has experience of 10 years in the field of teaching. She has guided around 20 UG/PG projects and has published few papers in National and International Conferences. Her primary research interest is Natural Language Processing, Text mining and Software Engineering.

Betina Antony is a Research Scholar in the Department of Computer Science and Engineering in Anna University, Chennai, Tamilnadu, India. She finished her Bachelors (Computer Science and Engineering) in Sri Sivasubramania Nadar College of Engineering and her Post graduation (Software Engineering) in College of Engineering, Guindy, Anna University, in which she secured gold medal for being the first rank holder. She has presented many papers in national and international conferences. She is currently working on Named Entity Recognition for Tamil Biomedical texts. Her research interests are Natural Language Processing, Text and Data mining.