International Journal of Engineering & Technology, 7 (2.29) (2018) 501-505

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

An Analysis of Ambiguity Detection Techniques for Software Requirements Specification (SRS)

Khin Hayman Oo[1]*, Azlin Nordin[2], Amelia Ritahani Ismail[3], Suriani Sulaiman[4]

[1]Kulliyyah Of Information and Communication Technology (KICT), International Islamic University Malaysia (IIUM)
[2,3,4]Department Of Computer Science, Kulliyyah Of Information and Communication Technology (KICT)
International Islamic University Malaysia (IIUM)

Abstract

Ambiguity is the major problem in Software Requirements Specification (SRS) documents because most of the SRS documents are written in natural language and natural language is generally ambiguous. There are various types of techniques that have been used to detect ambiguity in SRS documents. Based on an analysis of the existing work, the ambiguity detection techniques can be categorized into three approaches: (1) manual approach, (2) semi-automatic approach using natural language processing, (3) semi-automatic approach using machine learning. Among them, one of the semi-automatic approaches that uses the Naïve Bayes (NB) text classification technique obtained high accuracy and performed effectively in detecting ambiguities in SRS.

Keywords: *Ambiguity; SRS; Techniques*

1. Introduction

Requirements Engineering (RE) is a process of the production and refinement of a Software Requirements Specification (SRS). It executes an important role in software development life cycle (1) because SRS produced artifacts such as system design, coding and testing for the software development and the successful result of software project is mainly based on the quality of SRS documents. Hence, SRS is significant in software projects (2). SRS assists as a bond in the beginning of the development until the main point of quality control. Furthermore, formal languages are not understandable for most of the stakeholders to use in SRS. Thus, SRS are mostly written in natural language (3). However, natural language is basically ambiguous(3). Ambiguity means a word can be interpreted in more than one meaning (4). According to (5), the four most common types of ambiguity in SRS are (i) lexical, (ii) syntactic, (iii) semantic and (iv) pragmatic. *Lexical ambiguity* exits when a word has two or more possible meanings. *Syntactic ambiguity* is also known as structure ambiguity, appears when a sequence of words can be translated into more than one ways due to vague grammatical structure. On the other hand, *semantic ambiguity* is a sentence, which can be translated into more than one way within its context. Besides, *pragmatic ambiguity* arises when a sentence is not specific and the given context is lacking or missing the needed information to clarify its meaning.

1.1. Motivation

Ambiguity in requirements sentence is a major problem because it leads to the poor quality of SRS documents (6). Additionally, the software errors increase during requirements phase because of requirements specification are ambiguous and errors in requirements are widespread, harmful and costly such that it can lead to lower quality of SRS documents(7). Several techniques have been used to detect ambiguity in SRS. (8-10) suggested using human expertise in detecting ambiguity. However, (2, 11-17) used semi-automatic approaches using natural language processing techniques and (7, 13, 18-23) used semi- automatic approaches using machine learning techniques in detecting ambiguities. Semi-automatic means both human expertise and automatic techniques (natural language processing and machine learning) are used in detecting ambiguities. Based on the above, we can conclude that ambiguity detection approaches can be categorized into three groups and they are manual, semi-automatic using natural language processing techniques and semi-automatic using machine learning techniques. The detailed explanation is discussed in the Literature Review section.

2. Literature Review

Based on the previous work analysis, we can conclude that there are three approaches to detect ambiguity in SRS and they are manual, semi-automatic using natural language processing and semi-automatic using machine learning techniques.

2.1. Manual Approaches in Detecting Ambiguity

The manual approach in detecting ambiguity denotes that only requirements engineering expertise is used in detecting ambiguity and no automatic techniques are required. There are two types of manual approaches, which are inspection and review.

2.2. Inspection Techniques

(9, 24) mentioned that the inspection technique is performed by distributing the requirements to all stakeholders requesting for an interpretation. The interpretations are then compared across all

stakeholders. If one interpretation is different from the other, then the requirements are ambiguous.
The two methods of inspection techniques in detecting ambiguity are checklist and scenario-based reading techniques. The checklists are mainly used in detecting ambiguity of a single requirements sentence and scenario-based reading techniques are used for detecting passage level requirements sentence(7). Moreover, the checklists are not systematic, which means it does not provide any instructions on how to detect ambiguity; however, checklists techniques identify ambiguity based on previously identified ambiguity types (24, 25). Additionally, the general idea of a scenario-based reading technique is providing effective scenario with an inspector. For example, an inspector needs to create test cases for a requirements document and answer the questions thereafter. If the questions cannot be answered by the inspector then defect is therefore detected (25).

2.3. Review Techniques

The review process involves three steps. Firstly, the reviewers looked for ambiguity in documents manually. Next, the reviewers rated the collection of ambiguities based on their severities. Finally, both collected ambiguity and the review of natural language documents are sent back to author for corrections(15).
According to(9) review is a manual process, which consist of multiples readers who are required to check a document for irregularity and errors. A walkthrough is one type of review process, which is made up of a group of reviewers from requirements engineering background. The common walkthrough consists of at least one reviewer and most of the time reader's task is to present the document to the other group.
Based on the above description for both inspection and review processes, the inspection techniques are mostly used for detection of defects in the context of requirements documents. But, the review process is just for agreement between the reviewers and walkthrough is training for reviewers before reviewers begin the agreements between them. Moreover, the inspection techniques should be done before the process of detecting errors in the requirements and the walkthrough process(12).

2.4. Semi-Automatic Approach Using Natural Language Processing Techniques

The semi-automatic approach using natural language processing techniques are where requirements engineers used both natural language processing techniques with the assistance of human experts while detecting ambiguity in SRS. There are two types of natural language processing techniques and they are ontology and natural language patterns.

2.5. Ontology Classifier

The ontology text classification technique begins by removing less significant words of the documents. The ontology text classification then classified the relations between the concepts and the explanations of the concepts, which helps to clearly understand the potential meanings of an ambiguous terms (11).
According to (15), they developed the ontology based *ConceptNet3* in order to detect ambiguity in "common sense knowledge". Before they begin to use *ConceptNet3*, they used a dictionary to eliminate the typographical errors and spelling mistakes. Hence, if a sentence has many clauses, they will extract a simple sentence from it. Finally, they reported that *ConceptNet3* is able to detect ambiguity of common sense knowledge and compatible with other ontologies such as *WordNet* and *Brandeis Semantic Ontology.*
(8) built Requirements Engineering Specification Improver (RESI) to detect ambiguity in SRS. The RESI is semi-automatic tool in which ontology is the main techniques used to detect ambiguity. This paper discusses the four most popular types of ontology, which are *ResearchCyc, WordNet, ConceptNet* and *YAGA* that are used to detect ambiguity. Among them, *ResearchCyc* produces the most number of ambiguous words, *WordNet* is able to detect only synonyms and possible ambiguities, while *ConceptNet* and *YAGA* are not able to detect any ambiguity at all, but able to detect synonyms. In addition, they implemented a new version of RESI by improving the previous version. The new version of RESI represents a graph system to be able to read specification machine to communicate bi-directionally and while the ontologies look for ambiguities. Finally, they reported that they are able to identify semantic and syntactic ambiguities by using the new version of RESI(26) (**.**
On the other hand,(27) suggested that they used ontology based text classification method. At first, they trained data and then selected the most relevant documents for input from the training by using the Application Programming Interface (API). Furthermore, they used Protégé ontology editor to perform the creation, visualization and manipulation of ontologies in different types of representation formats. The purpose of using ontology is to minimize the misclassifications, which leads to ambiguous documents. However, the ontology text classification obtained low accuracy in automatic documents classification system.

2.6. Natural Language Patterns

In the process of detecting ambiguity using natural language pattern, the requirements engineers first look for ambiguity through a manual process. The natural language patterns are then matched based on the ambiguity of the words in the requirements. This process is known as the 'patterns matching process'. Finally, requirements engineers rewrite the requirements(13).
(16) have argued that they developed natural language application technique to detect ambiguity as well as to resolve the ambiguities named as *newspeak*. The *newspeak* is the refining of the Alvey Natural Language Toolkit (ANLT). In *newspeak,* they applied Goal-Structure Analysis (GSA) to build SRS by examining the effect of changing the specific requirements. GSA contains graphs structures of object and object is goal, effect, fact and condition. These association is a frame in GSA. The GSA helps to process all rare statements of a frame, detect ambiguities and translate each casually specified statement, which directs into a logic form.
Jiang et al. (2012) presented that they used active-learning based method to look for entity aliases without string similarity. First and foremost, they removed alias or entities from documents by using extractor. Then, they trained documents into the subset-based pairwise comparator after removing the alias. The purpose of using pairwise comparator is that it narrows down the scope of every details entity. Hence, it can keep every details entity and aliases in the same divisions for the construction an entity document index and extracting the most commonly applied entities in the documents. Moreover, they used a classifier to differentiate between aliases and non-aliases and if aliases are higher than non-aliases, they consider that they are the same. At last, the classifier stimulates active user learning by combining these objects to attain better performance with less training labels in each given entity. The training sample selector informally choses a subset of entity values and labelled for each sample to train in classifier until training is satisfied.

2.7. Semi-Automatic Approach Using Machine Learning Techniques

The four common types of semi-automatic approaches using machine learning techniques in detecting ambiguity are decision-tree, Support Vector Machine (SVM), Naïve Bayes (NB) and N-gram modeling.

2.8. Decision-Tree Text Classification Technique

The process of the decision-tree performs from a general to the specific search of a feature by adding the highest useful features to a tree structure to proceed the search. In learned decision tree process, each feature is selected during the search process, which is signified by a node and each node represents a selective point between number of unlike possible values for a feature. This process continues until all training examples are accounted by a decision tree (Pedersen, 2001).

One of the studies(19) suggested that they applied Test-Based Risk Identification Methodology (TBRIM), which is based on decision assistance for the requirements assessment. TBRIM can be used in both manual and automatic requirements assessment tool. For the TBRIM of manual process, they applied review method by providing instruction on how to detect ambiguity, conflict, complexity and technical errors. For the TBRIM automatic process, they applied Advanced Integrated Requirements Engineering System (AIRES) that is based on ASCII format version of the specification and they reported that AIRES assist the achievement and analysis of software requirements by identifying ambiguity, conflict, complexity and technical errors as compared to the previous.

On the other hand, (18) stated that they used fuzzy decision to detect ambiguity involving four steps. In step 1, they labeled the data manually for training data. In step 2, they trained the training data into the fuzzy decision-tree for building classifier. In step 3, they estimated unlabeled instances by using fuzzy decision-tree classifier to classify ambiguity. In the last step, they selected the highest classification ambiguous instances then, removed those ambiguous instances. Furthermore, if the collected instances were lower than the predefined sizes, they chose the next instance by repeating the step 1 to 4. Finally, they found out that this approach provided the sufficient evidence and assisted to the corresponding theoretical interpretation.

2.9. Support Vector Machine (SVM) text classification

In SVM model, requirements engineers used Part of Speech (POS) tagging to tag every element in the given corpus for representing a word. Next, SVM will assign weight to the corpus. The weight is then compared with threshold value. If the weight is larger or difficult to identify by SVM, then the errors are detected (18).

Moschitti, Riccardi and Raymond (2007) applied SVM, to detect the concept disambiguation for both syntactic and semantic structures on the Airlines Travel Information System (ATIS). Hence, they used multiclassification approach such as SVM-light-TK software[2], Poly-SVM (polynomial kernel) and Maximum Entropy Markov Models (MEMMS) or Conditional Random Fields (CRFs) to be used in the disambiguation classification concept. SVM-light-TK software[2] is used for encoding from all the tree kernels functions like Simple Parse Tree(SPT), Concept Marker Tree (CMT), All Concept Market Tree (ACMT) and Percolating Concept Marked Tree (PACMIT), which assisted in classifying disambiguation process. The Poly-SVM depends on the SVM's a polynomial kernel for supporting the disambiguation concepts in documents. Maximum Entropy Markov Models (MEMMS) or Conditional Random Fields (CRFs) is used for discovering sequence due to the structure of the graph, which displayed individuality label sequence model. CRFs require a single combined probability distribution from the complete label sequence. Maximum Entropy Markov Models (MEMMS) or Conditional Random Fields (CRFs) is a type of undirected graphical model for discovering the sequences because the structure of the graph shows the independence label sequence model within the sentence. Thus, CRFs will specify a single joint probability distribution from the whole label sequence of the sentence. Finally, they reported that they combined Poly-SVM and PAMCT called (Combine kernel Model (CKM)) for achieving high accuracy. Regardless of the combination techniques for the disambiguation process, some techniques however, did not perform well in disambiguation process.

(17) suggested that they built pattern recognition system to detect ambiguous patterns. They used SVM by specifying every elements into unlike features with the independent classification of the elements. By using SVM, they are able to detect ambigutiy in handwritten digit recognition. Likewise, they also used SVM based kernel functions to map between different spaces of hyper plane (splitting classes of the table from positive element to negative and maximizing the separation of classes) by dividing two levels of the class. Finally, they look for ambigutiy by looking for classses that have more features in general.

2.10. Naïve Bayes (NB) text classification

The NB text classifier trained the text based on word probability and word count method. It classifies a text through a class based on the words, which appears in the text content by dealing with the probabilities of the words learned from the training data(28).

Mishra & Shukla (29) stated that they implemented a hierarchical text classification, which has two types of text classification and they are NB text classification and Euclidean distance text filtering to detect ambiguities in requirements. The NB text classifier classified the text by estimating the class label based on the training document. They used text filtering classifier to re-classify the process from the NB classifier by narrowing down the number of requirements to reduce errors. They reported that this method supported requirements engineers to save time during the process of software requirements.

Sharma *et al.* (21) used NB classifier to detect ambiguity in requirements. Before they began with NB classifier, they tokenized the training data using bag of words (BOW) to get the sequences of words from documents and they removed stops words and unique words. Then, they also used *n-gram* model to obtain consecutive sequence of words in which they focused on the use of unigrams and bigrams with Part of Speech (POS) tagging. Finally, NB classifier classified the documents of *n-gram* based on POS tagging to detect ambiguity in requirements documents. Another study(20) also suggested that they used NB classifier to classify the antecedents and anaphoric ambiguity, which occurs when a sentence has more than one antecedent. Anaphor are nouns or noun phrases, which is also called antecedent of personal pronoun or anaphor because antecedent can be the expression of the anaphor.

According to Maroulis, (30), they indicated that they developed NB classifier to predict the defect proneness of software module because it helped to reduce the testing efforts and software modules time. The feature selection method is very significant for developing defect prediction model. This is because if the task of attributes is larger, it is difficult for normal classifier to resolve this problem. Hence, this paper used NB classifier for defect prediction based on Eclipse and KC1 running bug databases. NB is able to handle for both continue and discrete attributes of Eclipse and KC1 for each attributes of every files in predicting defects.

Brown, et al. (31) pointed out that they applied NB text classifier to detect coordinating ambiguity in natural language. The coordinating ambiguity means more than one "and" and "or" is used a sentence. They also compared with other machine learning techniques like K-NN, Random tree and Random Forest in detecting coordinating ambiguity. Finally, they reported that NB achieved higher accuracy in detecting coordinating ambiguity as compared to K-NN, Random Forest and Random tree.

2.11. Statistical Machine translation using *n-gram* model

The *n-gram* model can be used to detect ambiguity in two ways, which are using probabilistic language models and using words only for prediction. In probabilistic language model, the counting words in corpus is for assigning the probability distribution across words(32). However, the word prediction method is predicting the next word in a sentence or getting consecutive sequence of the words. The other machine learning techniques of text classification will be then used to calculate the probability of the word(20).

Allahyari-Abhari *et al.,* (22) stated that they used *n-gram* model for predicting a word from previous words in a text based on the classes of words. Some have similar word classes with other words in relation with their meaning and syntactic purposes. Hence, they used unigrams, bigrams and trigrams with the same training text. They then measured the perplexity(ambiguity) of the Brown corpus by using class-based estimators with the word-based estimators. Their finding showed that trigram produced more significant perplexity value critical of the classes for their 236 out of 244 tested data.

(33) applied Terms Ambiguity Detection (TAD) approaches, which involves language modeling (*n-gram*), ontologies and clustering for detecting terms ambiguities. They used *n-gram* model to detect non-referential ambiguity based on the understanding of the terms and they determined the terms, which are referential. For example, terms, which has no-named entity contexts are considered non-referential as well as ambiguous. Then, they used ontologies to detect cross domain ambiguity and used clustering to detect for both non-referential and across domain ambiguities. At the end of the ambiguity detection process, they labeled the term as ambiguous if any one of these approaches predict an ambiguity, but they labeled the term as non-ambiguous if each of these approaches predict non-ambiguity.

The recent study of (34) pointed out that they used *n-gram* model to look for several types of grammatical passive sentences in English. They applied *n-gram* model to record the log probabilities (logprobs) for dealing with scoring function. The scoring functions are based on the properties of the string, which has the parameters of the model. In their experiment, they tested with many different types of passive based sentences classification using *n-gram* model and they had tested with two different types of score, i) a normalized the log probability given by *n-gram* model to a string for eliminating the significant parts that do not effect on the grammatical sentence such that sentence length and frequency of word, ii) the second score is depended on the ungrammaticality of a sentence, which is considered for the problematic modules. Moreover, these two scores are function of the least scoring *n-gram* in the sentence. At last, they demonstrated that n-gram achieved high accuracy in classifying different types of tasks for passive sentences.

3. Conclusion

As mentioned in literature review section, three approaches to detect ambiguity in Software Requirements Specification (SRS) are manual techniques, semi-automatic approach using natural language processing techniques and semi-automatic approach using machine learning techniques. The strengths and the weaknesses of these three approaches are discussed in below.

The advantage of using the manual approach in detecting ambiguity is that if the inspection and reviewing techniques are combined, it provides instruction and guides on what should be looked for by the inspectors and reviewers. In general, the manual approach is better for those who is less familiar with the software domain (15). The disadvantage of manual approach however, is that some of the inspectors assume that they can detect ambiguity by looking at requirements sentences without instruction. Furthermore, one of the question in the checklist item stated: *"Is the requirements ambiguous?"*. Such question requires only a 'yes' and 'no' answer without any supporting details, which may lead to more critical problems in ambiguity detection considering that readers may not even be aware of the ambiguity itself (8). Additionally, reviewers naturally disambiguate the ambiguous documents and they are not aware of other possible interpretations. Instead, they just interpret the document based on first come to their mind known as intended interpretation (16).

The interests in using ontology classifier is that it supported the document understandable by providing the possible subjects or documents even though the input document is ambiguous suggested by the ontology classifier, but still need to increase the accuracy in detecting ambiguity (21). Additionally, the pattern matching process for detecting ambiguity in SRS stated that requirements engineer looks for ambiguity using natural language patterns. However, it still needs to rewrite the requirements after patterns matching process (17). Both ontology classifier and natural language patterns processes are in the category of the semi-automatic approach using natural language processing.

The most benefit of using semi-automatic using machine learning techniques to detect ambiguity in SRS are that decision-trees technique is used to detect ambiguity and also performed well in a lot of relative studies (Pedersen, 2001). The Support Vector Machine (SVM) is able to apply many natural language processing jobs in order to detect errors effectively (18). Naïve Bayes (NB) classifier performed well in detecting ambiguity (21) and also achieved high accuracy in detecting ambiguity (22, 28, 30, 35). The *n-gram* model is able to help in detecting ambiguity by creating language model and extracting features from a text corpus that can count the frequencies from the appearance of n-gram in a text or word sequence (30).

However, NB classifier is more consistent in detecting ambiguity as compared to the decision-tree (22). NB classifier performed better than Logistic Regression followed by SVM in classification between informational and ambiguous queries based on the use entropy that is known as maximum entropy model to classify some natural language (35). Although *n-gram* model able to detect ambiguity based on assumption, which is also called probability generalized model, nonetheless, the output of the *n-gram* model may produce misleading result(30).

Based on our conclusion of strengths and weakness analysis, we can conclude that Naïve Bayes (NB) classifier has achieved the best performance in accuracy as compared to other machine learning techniques and performed well in detecting ambiguity as compared to natural language processing techniques as well as manual techniques.

References

[1] Carlson N, Laplante P. The NASA automated requirements measurement tool: a reconstruction. Innovations Syst Softw Eng. 2014;10:77-91.

[2] Hussain I, Ormandjieva O, Kosseim L. Automatic quality assessment of SRS text by means of a decision-tree-based text classifier. Proceedings - International Conference on Quality Software. 2007(Qsic):209-18.

[3] De Bruijn F, Dekkers HL. Ambiguity in natural language software requirements: A case study. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2010;6182 LNCS:233-47.

[4] Bussel DV. Detecting ambiguity in requirements specifications. 2009(09).

[5] Kiyavitskaya N, Zeni N, Mich L, Berry DM. Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Requirements Engineering. 2008;13(3):207-39.

[6] Yang H, Willis A, Roeck AD, Nuseibeh B, De Roeck A. Automatic detection of nocuous coordination ambiguities in natural language requirements. Proceedings of the IEEE/ACM international conference on Automated software engineering SE - ASE '10. 2010:53-62.

[7] Kamsties E, Berry DM, Paech B, Kaiserslautern D. Detecting Ambiguities in Requirements Documents Using Inspections. 2001:1-13.

[8] Denger C, Berry DM, Kamsties E. Higher quality requirements specifications through natural language patterns. … : Science, Technology and …. 2003:1-11.

[9] Havasi C, Speer R, Alonso JB. ConceptNet 3: a Flexible, Multilingual Semantic Network for Common Sense Knowledge. In Proceedings of Recent Advances in Natural Languges Processing 2007. 2007:1-7.

[10] Hill E, Fry ZP, Boyd H, Sridhara G, Novikova Y, Pollock L, et al. AMAP: Automatically mining abbreviation expansions in programs to enhance software maintenance tools. Proceedings - International Conference on Software Engineering. 2008:79-88.

[11] Körner SJ, Brumm T. RESI - A natural language specification improver. ICSC 2009 - 2009 IEEE International Conference on Semantic Computing. 2009:1-8.

[12] Körner SJ, Brumm T. Improving natural language specifications with ontologies. Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering, SEKE 2009. 2009:552-7.

[13] Wang X-Z, Dong L-C, Yan J-H. Maximum Ambiguity-Based Sample Selection in Fuzzy Decision Tree Induction. IEEE Transactions on Knowledge and Data Engineering. 2012;24(8):1491-505.

[14] Misra J, Das S. Entity Disambiguation in Natural Language Text Requirements. 2013 20th Asia-Pacific Software Engineering Conference (APSEC). 2013:239-46.

[15] Wijewickrema CM. Impact of an ontology for automatic text classification. Annals of Library and Information Studies. 2014;61(4):263-72.

[16] Nakagawa T, Matsumoto Y. Detecting errors in corpora using support vector machines. Proceedings of the 19th international conference on Computational linguistics-Volume 1. 2002:709-15.

[17] Polpinij J. An ontology-based text processing approach for simplifying ambiguity of requirement specifications. 2009 IEEE Asia-Pacific Services Computing Conference, APSCC 2009. 2009:219-26.

[18] Polpinij J, Ghose A. An automatic elaborate requirement specification by using hierarchical text classification. Proceedings - International Conference on Computer Science and Software Engineering, CSSE 2008. 2008;1:706-9.

[19] Seijas L, Segura E. Detection of ambiguous patterns using SVMs: Application to handwritten numeral recognition. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2009;5702 LNCS:840-7.

[20] Clark A, Giorgolo G, Lappin S. Statistical Representation of Grammaticality Judgements: the Limits of N-Gram Models. Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics. 2013:28-36.

[21] Sharma R, Bhatia J, Biswas KK. Machine learning for constituency test of coordinating conjunctions in requirements specifications. Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering - RAISE 2014. 2014:25-31.

[22] Allahyari-Abhari A, Soeken M, Drechsler R. Requirement Phrasing Assistance Using Automatic Quality Assessment. Proceedings - 2015 IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits and Systems, DDECS 2015. 2015:183-8.

[23] Gause DC, Gause DC, Weinberg GM, Weinberg GM. Exploring requirement. Quality before design. 3. 1989.

[24] Popescu D, Rugaber S, Medvidovic N, Berry DM. Reducing ambiguities in requirements specifications via automatically created object-oriented models. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2008;5320 LNCS:103-24.

[25] Anda B, Sjøberg DIK. Towards an Inspection Technique for Use Cases Models. Proceedings of the 14th international conference on Software engineering and knowledge engineering - SEKE '02. 2002(1325):127-34.

[26] Osborne M, MacNish CK. Processing natural language software requirement specifications. Proceedings of the Second International Conference on Requirements Engineering. 1996:229-36.

[27] Romano JJ, Palmer JD. TBRIM: decision support for validation/verification of requirements. Systems, Man, and Cybernetics, 1998 1998 IEEE International Conference on. 1998;3:2489-94 vol.3.

[28] Subha R, Palaniswami S. Ontology extraction and semantic ranking of unambiguous requirements. Life Science Journal. 2013;10(2):131-8.

[29] Mishra B, Shukla KK. Impact of attribute selection on defect proneness prediction in OO software. 2011 2nd International Conference on Computer and Communication Technology, ICCCT-2011. 2011:367-72.

[30] Maroulis G. Comparison between Maximum Entropy and Naïve Bayes classifiers : Case study ; Appliance of Machine Learning Algorithms to an Odesk ' s Corporation Dataset Georgios Maroulis Submitted in partial fulfilment of the requirements of Edinburgh Napier University. 2014(January).

[31] Brown PF, DeSouza PV, Mercer RL, Della Pietra VJ, Lai JC. Class-Based n-gram Models of Natural Language. Computational Linqustics. 1992;18(1950):467-79.

[32] Tyler Baldwin YLBAIRS. Automatic Term Ambiguity Detection. Acl. 2013;2:804-9.

[33] Singh S. International Journal of Advanced Research in Computer Science and Software Engineering Ambiguity in Requirement Engineering Documents : Importance , Approaches to Measure and Detect , Challenges and Future Scope. 2015;5(10):791-8.

[34] Popescu D. Reducing Ambiguities in Requirements Specifications via Automatically Created Object-Oriented Models. 2007;1.

[35] Wang Y, Agichtein E. Query Ambiguity Revisited : Clickthrough Measures for Distinguishing Informational and Ambiguous Queries. Computational Linguistics. 2010(June):361-4.