International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-5, January 2020

Actionable Analytics on Software Requirement Specifications

Lida Bamizadeh, Binod Kumar, Ajay Kumar, Shailaja Shirwaikar

Abstract: *The volume of data and need for churning this data to provide useful information has increased the scope of data mining and made it promising in recent years. Software intelligence (SI) (as the future of the mining software engineering data) presents theories and techniques to augment software decision making by using fact-based support systems. SI exposes software practitioners to up-to-date and relevant information to support their daily decision activities over the complete software development life cycle. Software documents contain important information for a plenty of software engineering tasks and one such important document is Software requirement specification (SRS) which details the system and user requirements. Inexplicit, ambiguous or imperfect requirements guide leads to a non-acceptable product by users. Constructing of a strong software specification can be supported by building a semantic space, validating new specification for completeness, categorization of software requirement specification and identification of significant concepts and related keywords. This paper proposes a knowledge management system for software document repositories using data analytics and demonstrates its creation and usage for a document set of software requirement specifications.*

Keywords: clustering, semantic analysis, Software intelligence, software requirements specification.

I. INTRODUCTION

Software has a wide impact on all aspects of our life as most systems are controlled by software, thus software plays an important role in business, societies and governments [1]. As Software become complex, it need to be constructed in a systematic manner and thus Software development is spread over several phases such as requirements, design, implementation, testing and maintenance [2]. Software development is a data intensive process where large amount of data gets generated through different phases [3]. Software analytics gives power to practitioners to manage data discovery and analysis for extracting actionable and insightful information for performing data-driven tasks of software systems [4] [5]. Methodology of mining software engineering data includes five main steps:

1- Collecting SE data or determining SE tasks: Software engineers can start with either a data-driven approach of collecting/investigating SE data to mine or problem-driven approach of determining SE task to act upon. In practice it is a hybrid approach where these two approaches can be combined.

2- Pre-processing: Low quality of SE data gives low-quality of SE mining results. When data pre-processing techniques are applied before using mining algorithms, they can implicitly improve the quality of the patterns mined and time expected for mining [6].

3- Adapt/ Adopt/ Develop mining algorithm: Design an algorithm or design a method that supports making decisions for particular SE task. Data will be collected and performance data will be evaluated.

4- Knowledge extraction and storage: The knowledge extracted by mining algorithms is stored in a knowledge base so that it is available for future tasks.

5- Post-processing: Discovered knowledge should be presented in high-level languages, obvious explanation, or other meaningful forms so that the information can be easily recognized and directly usable by people.

In mining software engineering data, availability of rich data and analysis of the data is very important [7]. Types of software repositories are:

1- Historical repositories: control repositories, bug repositories and archived communications about evolution and progress of a project [8].

2- Run time repositories: deployment logs contain information about the execution and the usage of a software system at a single or multiple deployment sites [9].

3- Code repositories: source codes of various software systems.

4- Documentation repositories: Software requirement specification, software acceptance testing, software deliverable [10].

Software documents contain important information for a plenty of software engineering tasks (software maintenance, requirements engineering, etc). Software documentation includes rich information of both functional and non-functional requirements, as well as information relevant to the application area [11]. Software requirement specification (SRS) is a documentation of the system and user requirements [12].

Manuscript published on January 30, 2020.
* Correspondence Author
Lida Bamizadeh*, Department of Computer Science , Savitrabia Phule Pune University, Pune, India. E-mail: lida_bamizadeh@yahoo.com
Dr. Binod Kumar, Department of Computer Applications, JSPM Jayawant, Pune, India. E-mail: binod.istar.1970@gmail.com
Dr. Ajay Kumar, Technical Campus, JSPM Jayawant, Pune, India. E-mail: ajay19_61@rediffmail.com
Dr. Shailaja Shirwaikar, Department of Computer Science, Savitrabia Phule Pune University, Pune, India. E-mail: scshirwaikar@gmail.com

It is a complete presentation about how the system is expected to work. SRS should be correct, complete, unambiguous, consistent, and modifiable. In practice, it is tedious and difficult to write requirements specification for a software project. Software requirement specifications are generally written in natural language [13]. The purpose of this study is to demonstrate usefulness of mining software requirement specifications.

In order to extract knowledge from unstructured documents such as SRS, a strong tool is necessary. R, an open source free software environment and programming language with a wide range of statistical libraries, is used for this purpose [14].

This paper is organized as follows: Next section describes background and related work. Section 3 explains Software Analytics Model for Software Specification. The experimental demonstration of the model using the R code and the results achieved, are discussed in Section 4, which is followed by the concluding remarks in Section 5.

II. BACK GROUND AND RELATED WORK

A. Software Intelligence

The work by Xie et al. [15] demonstrated that an important goal of software engineering is to improve the software productivity and quality. Software mining has recently revealed itself as a promising subject to achieve this goal with respect to two main trends: the growing volume of data and its demonstrated utility in finding solutions to great number of real- world problems. Hassan and Xie [16] reported, while business intelligence (BI) presents ideas and techniques to enhance decision making for business processes, software intelligence (SI) also need to support software practitioners in daily decision making. Software Intelligence involves software analytics and mining of software repositories. The challenges faced by software intelligence researchers include:

- Considering techniques to be applied throughout the life cycle of a project
- Designing techniques for other than code
- Focusing on all types of software repositories
- Using effective mining algorithms in software mining domain
- Adopting the results in practice.

Buse and Zimmermann [17] proposed analytics to cover the gap between the knowledge required by project management for making decisions and the knowledge extracted by existing tools. Project manager can manipulate project complexity, improve efficiency, manage risks, predict changes and evaluate previous decisions using insights from mined software data. Later, Buse and Zimmermann [4] had a survey that resulted into several guidelines for software analytics such as engineers might not be expert in data analytics but tools should be easy to use, fast and accurate and also should cover all types of artifacts and indicators. Furthermore, engineers need to drill down into data based on time, organizational structure, and system architecture. Menzies and Zimmermann [18] suggested that every software analytics tool is not suitable for every software system and it might be proper for specific domains. The research by Carreteiro et al. [3] introduces on-going architectural case study using software maintenance tasks as a means to enhance the knowledge flows within the organization. The researchers have applied analytics on different types of SE data and most prominent being code, logs and bug reports.

B. Code Mining

Wang et al. [19] developed a tool for mining API usage patterns to efficiently support developers in practice. They proposed UP-Miner for mining usage patterns of API methods from source code. The clustering strategy before and after mining, successfully decreases redundancy and improves the succinctness of the mined API usage patterns. Microsoft developers reported UP-Miner as effective in practice after evaluating it on large-scale Microsoft codebase. Wang et al. [20] proposed strong demonstration-learning algorithm namely deep learning, to exploit semantic features of programs automatically from source codes that are in form of abstract syntax tree. The semantic features earned automatically by Deep Belief Network (DBN) can be used to improve both within-project defect prediction (WPDP) and cross-project defect prediction (CPDP). A method is suggested for estimating defectiveness of source code by Kapur and Sodhi [21]. Dam et al. [22] introduced a deep tree-based model for software defect prediction.

C. Log Mining

Lou et al. [23] introduced an industrial system named SAS (service analysis studio). SAS is a data-driven system for improving the performance of incident management in a large online service of Microsoft. A major problem in this system is analysis of huge amount of monitoring data, which can be solved by software analytics. A novel approach for contextual analysis of system logs was proposed, for comprehension of performances of a system, by Fu et al. [24]. First they applied execution patterns to demonstrate execution structures revealed by a series of system logs, and suggest an algorithm to mine execution patterns from the program logs. The mined execution patterns relate to diverse execution routes of the system. Yu et al. [25] evaluated real world execution traces and proposed a novel trace-based method containing impact analysis and causality analysis. The impact analysis scales performance impacts on an element basis, and the causality analysis detects patterns of runtime actions that are likely to cause the measured effects.

D. Bug Reports Mining

A Bug Locator proposed by Saha et al. [26] based on code constructs, enables more accurate bug localization. This method, called as BLUiR, takes source code files and extracts all information such as class names, method names, variable names, comments etc. Using mined structural information, a distinct search into the bug report is executed, and the total scores through all the searches are combined with different weights. Tantithamthavorn et al. [27] proposed class level bug localization where co-change histories of each defined bug are inspected instead of the evaluation of earlier bug history of software.

They executed experiments on two OSS datasets, the Eclipse SWT 3.1 project and the Android ZXing project. Later Rahman [28] proposed a Statement level Bug Localization (SBL) technique which is based in two steps, first recognizing precise buggy methods using source code search space minimization (termed as MBuM) and second statement level bug localization using achieved buggy methods.

E. Document Repositories

Documentation repositories are rarely investigated because of their limited accessibility and static nature. These are deliverables, even if problems are identified it is too late to rectify them, so there is not much work on this type of software repositories. MSR4SM was a method proposed by [8] to exploit the proper knowledge from each software repository created on maintenance demand and the existing system. MSR4SM uses the topic model to extract the topics from software repositories. The related knowledge corresponding to topic, in each software repository, is extracted. Reference [1] presented the outcomes of an analysis of the suggestions to knowledge management in requirements engineering (RE), in which it is obvious that the suggestions fail to meet the needs of work teams.

F. Text Mining

Text is usually a collection of unstructured documents and most of the documents in software document repositories are unstructured documents. Text embeds knowledge not only by carrying information clearly over sentences but also indirectly over how words co-occur with each other. That indirect knowledge can be exploited and discovered, at least in part, via text mining. Gulo and Rúbio [29] developed solutions for text data in social network analysis using R language. Technique determines a bipartite graph among documents and topics made using the Latent Dirichlet Allocation topic model. Gefen et al. [14] presented a text mining with latent semantic analysis (LSA) using R. Relevant texts are collected, a semantic space is constructed and words, phrases, or documents are projected onto that semantic space to calculate their lexical similarities. Arnold [30] proposed a data model to improve experimental data analysis and predictive modeling. The R package cleanNLP, is used as an application of this data model. Particular annotations delivered contain tokenization, part of speech tagging, named entity recognition, and word modeling. Schubert et al. [31] presented a novel methodology to model word significance and word affinity in a text and build the word cloud based on the derived dependency. Welbers et al. [32] provided a summary of common steps and actions in a computational text analysis project and demonstrated how every step can be completed using the R statistical software.

III. SOFTWARE ANALYTICS MODEL FOR SOFTWARE SPECIFICATION

A strong knowledge base can be created from the available document repositories using analytics as shown in Fig.1.

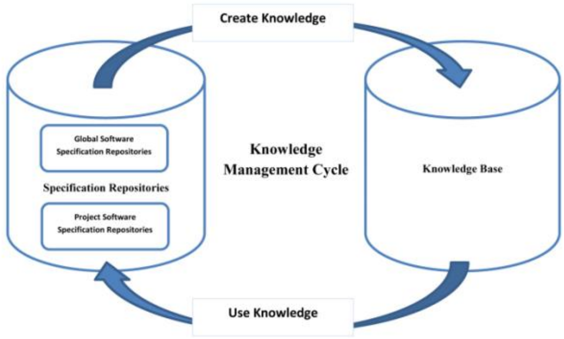

Fig. 1.Knowledge management of document repositories.

Global document repositories can be constructed by depositing software engineering documentation for several successful software projects. This documentation includes project proposals, software requirement specification (SRS), acceptance testing, etc. The first step involves extracting required type of documents for example SRS documents. Most of the documents in document repositories are unstructured documents. The process of creating and using the knowledge base is explained in Fig. 2.

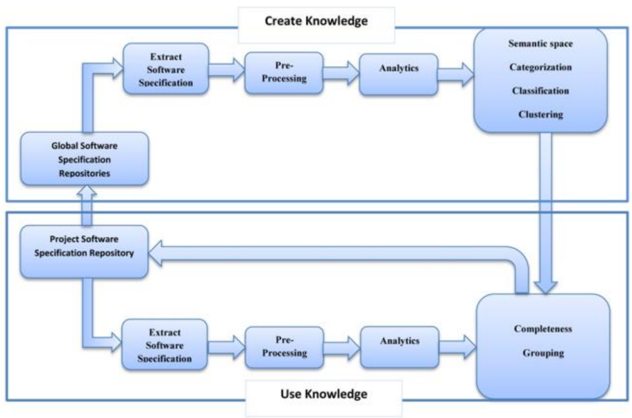

Fig. 2.Knowledge extraction and use.

Pre-processing involves capitalizing, removing numbers, spaces and punctuations, removing stop words and stemming. A term document matrix (TDM) is then constructed to impose structure on the document set where in the major terms in the document act as features. Several mining algorithm can then be applied to extract knowledge and build a knowledge base as shown in Figure 1.

1. Latent semantic analysis (LSA) can be used to build a semantic space.
2. Visualization of TDM using wordcloud
3. Clustering can be used for grouping and categorizing of documents.
4. Classification can be used to group documents based on domain knowledge.

This can be subsequently used when creating documents for new projects.

This knowledge base can be used for improving the document by
1. Validating new documents for completeness
2. Categorization of documents
3. Identification of significant concepts and related keywords.
The refined and completed document can then be deposited in the global document repository.

IV. EXPERIMENTAL DEMONSTRATION OF THE MODEL

The experimental demonstration process splits into two stages namely create and use. Table 1 shows the phases of each stage:

Table- I: The phases of create and use

Create	Use
1- Extract and Validate data 2- Pre-processing 3- Construct TDM, distance matrix, semantic space 4- Apply clustering , classification 5- create Wordcloud, Dendrogram, term relationships, document relationships	1- Extract document 2- Pre-processing 3- Visualization using wordcloud 4- Checking Completeness 5- Categorization 6- Identifying keywords and concepts

a. Sample of a Table footnote. (*Table footnote*)

Documentation repositories include Software requirement specification, software acceptance testing, and other Software deliverables. Software requirements specification is chosen as data for further analysis. SRS is text data and it is unstructured. Unstructured data has uncertain length, academic text, floating of singular and plural forms of words, alphanumeric characters and punctuations, and the contents are not predefined to adhere to a set of values [29]. Experimental demonstration of the model is the process of managing data discovery and analysis for achieving actionable and insightful information.

A. Create Knowledge

1) Extracting data

Software requirement specifications are downloaded from across the internet. Collected data is 15 to 20 requirement specifications of online shopping for performing software analytics. Availability of specifications is very limited. For each text mining task, the scope of the text is very easy to determine. Emails or call log files normally put into a single vector for every message. But, for longer documents one need to determine whether to put whole document or to split the document out into sections, paragraphs or sentences. For some tasks like classification or clustering, mostly whole document is the appropriate scope; however for semantic analysis, sentiment analysis, information retrieval or document summarization, smaller measures of text like sections or paragraph are more suitable [33].

2) Automated Slicing of data

Every specification can be sliced contextually into several sections by manually comparing it with a standard software requirement specification template (IEEE SRS template) [30]. For large amount of data, it is difficult to apply slicing manually because it is time consuming process. For automated slicing following steps are applied for each specification separately. Every specification is automatically split into paragraphs without considering concept. Then apply pre-processing and create a term document matrix as well as a distance matrix. Hierarchical clustering is then applied to generate a dendrogram. The tree can be cut at different levels to generate required number of clusters. To find the best k (number of clusters) Silhouette analysis is used. It is a measure for every observation to realise how well it is close to other observations in its own cluster with how close it is to observations in other clusters. Silhouette coefficients have a range of [-1, +1]. Silhouette values close to +1 mean that the observation is well located in its cluster and far away from the neighbouring clusters. Silhouette value of -1 directs that the observation is close to its neighbouring clusters than to the cluster it's assigned and it might have been allocated to the wrong cluster. The average silhouette width is used to select the best value of k when k is examined from 5 to 30. Table 2 shows initial terms, slice and the chosen k value and corresponding average silhouette width for 19 specifications. The documents slices in the clusters are merged thus generating k number of slices for each document.

Table- II: Automated slicing using silhouette analysis

Specification	No of terms	No of automated slicing	The best k(No of clusters)	Silhouette average width
1	37	458	5	0.4796654
2	77	1158	11	0.2791863
3	13	64	28	0.4111662
4	30	114	15	0.1850315
5	21	111	14	0.234965
6	28	108	29	0.5413711
7	20	201	5	0.5831709
8	34	253	30	0.4715626
9	44	161	5	0.3339266
10	12	151	5	0.5040533
11	38	272	5	0.2259984
12	12	96	30	0.6007088
13	51	404	30	0.4150056
14	94	497	13	0.2535404
15	275	1801	5	0.2820093
16	10	129	30	0.7856327
17	107	664	7	0.3701346
18	29	206	30	0.5596881
19	33	138	5	0.1944905

3) Data pre-processing

Textual data can be available in diversity of file formats. R natively supports reading normal text files such as CSV and TXT, but some text files such as JSON, HTML, XML, Word, Excel and PDF require installing of additional packages [32][34]. It was necessary to install and load text mining(tm) package of R for text mining [35]. Data pre-processing is transforming unstructured and semi-structured text into the structured vector space model. The main stages of text pre-processing are: Converting to Lower case, Removing numbers, Removing Punctuations, Removing stop words, Stemming, and Stripping whitespace [14].

4) Constructing TDM

In pre-processing high number of terms is reduced. Now, text data is clean and ready for analysing. Based on all remaining terms a matrix of documents and terms is created.

It is a mathematical matrix that shows the frequency of terms (number of times that a term occurs into a document).
This evaluation generates the weight of a term directly symmetric to its frequency in each document and inversely symmetric to its frequency to the set of documents. In a term-document matrix rows correspond to terms and columns correspond to documents in corpus [32]. Weighting of TDM is term frequency- inverse document frequency (TF-IDF) [14]. TDM computed for 20 documents after slicing includes 508 terms and 202 documents with 87% sparsity.

5) Constructing distance matrix and the dendrogram

After the vector representation (term document matrix) is created, the similarity among documents is computed as the distances between the vectors displaying them. The cosine similarity is a scale to computes the cosine of the angle between two vectors. After creating distance matrix a dendrogram is generated. Fig. 3 shows the dendrogram.

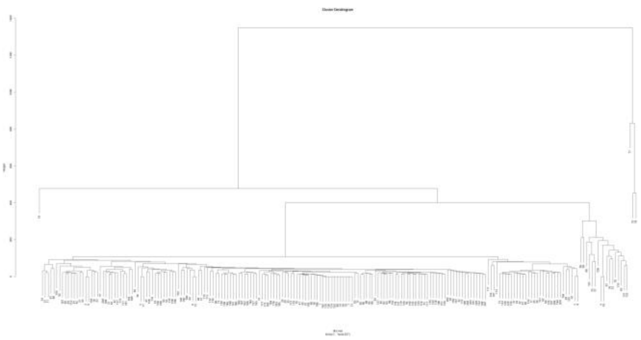

Fig. 3.Dendrogram after automated slicing

6) Creating semantic space

A semantic space is creating using Latent Semantic Analysis (LSA). LSA is a technique in natural language processing that brings out implicit knowledge conveyed when one word co-occurs with another [14]. Dimensionally reduction is built into LSA [36] as terms are internally displayed by LSA as vectors of certain ranks (number of dimensions) based on a transformation of the co-occurrence matrix. LSA is using a bag of word approach, it analyses words irrespective to their part of speech (noun, verb, adjective, adverb) or their place in the sentence. Mathematically LSA is performed by operating singular value decomposition (SVD) on the term document matrix (TDM). SVD is a two mode data reduction analysis that converts TDM to three matrices: 1- Terms (vector u or left singular values that display the rows of TDM). 2- Documents (vector v or right singular values that display the columns of TDM). 3- Vector d includes singular values [14].

B. Use knowledge

The created Knowledge is stored in the Knowledge base and can be used in subsequent SE tasks.

1) Visualizing Semantic space using Wordcloud

A wordcloud is a visual quantitative summary of the frequency of words in term document matrix or corpus. Through the use of wordcloud function, most frequently used keywords are highlighted and one can easily visualize all the related keywords [31]. The wordcloud variations for different document sets are presented in Fig. 4.

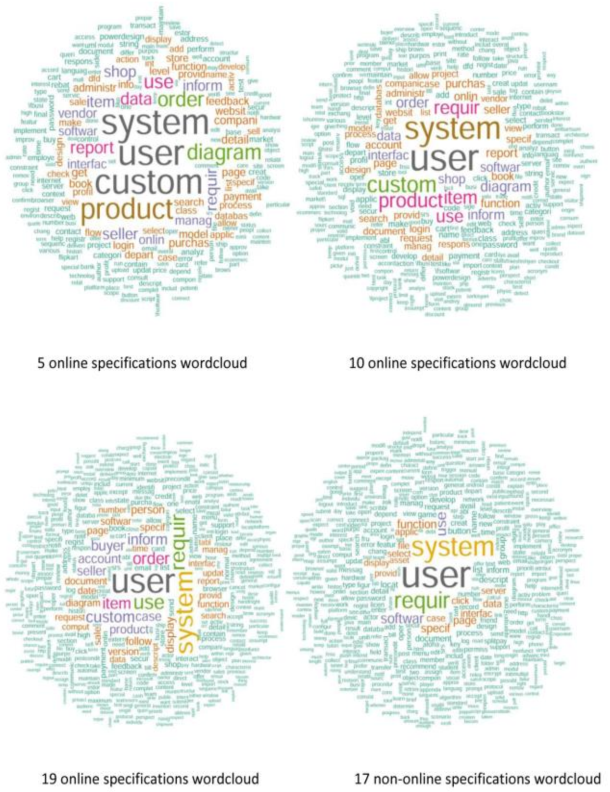

Fig. 4.Wordcloud variations

2) Checking Completeness of specifications

One of the usages of knowledge base is to check the completeness of the specification. SRS contains both functional requirements which are specific to an application and non-functional requirements which are common and mandatory requirements. Thus SRS contains both application specific terms as well as Generic terms which are essential for the specification to be complete. To check completeness, consider several domains of software requirements specifications and a set of specifications in each domain. For each domain, a semantic space is constructed using LSA.

The terms thus generated can contain some terms which are generic and some specific to the domain. To get the generic terms, intersection of all term sets can be taken.

Let D1, D2 …. Dn denote the term sets for each domain then

Generic terms = $\bigcap_{i=1}^{n} Di$

Specific terms corresponding to each domain can also be generated

Specific terms of Di = setdiff(Di – Generic term)

Domains can be several but here only two sets of specifications are considered. First selected domain is online shopping wherein there are 19 specifications. All the remaining 17 specifications form the second set of non-online specifications. The all terms derived from online shopping specifications and non-online specifications respectively are 508 and 547 terms. The generic terms of two domains are 382 terms. For each domain if generic terms are subtracted from all terms, specific terms of that domain will remain. The specific terms for online shopping are 126 terms.

Presence of generic terms ensures that software requirement specification is complete and Jaccard similarity can be used to verify the completeness of a document.

For each single specification term document matrix is generated with reducing threshold frequency. Jaccard similarity between terms of each single specification and the Generic terms can be used as a measure of completeness of the document. Fig .5 and Fig. 6 show comparative study of completeness of 19 on-line specifications and 17 other specifications respectively.

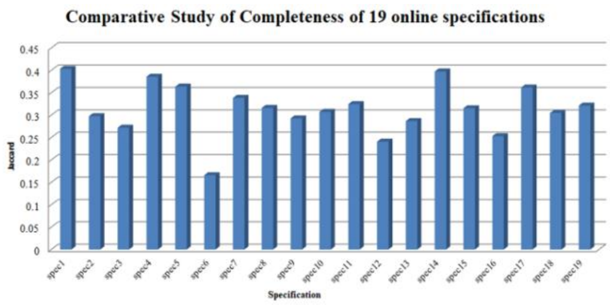

Fig. 5.Comparative Study of Completeness of 19 online specifications

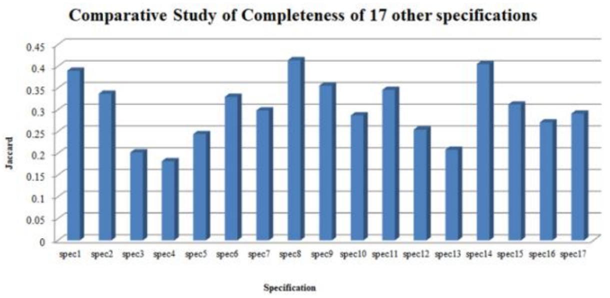

Fig. 6.Comparative Study of Completeness of 17 other specifications

3) Categorization of specifications using specific terms

The specific terms generated for a particular domain and Jaccard similarity can be used to categorize a new specification. For each specification, remaining terms were computed by removing the generic terms. The jacccard similarity between specific terms of the online domain and the remaining terms was computed which acts as a measure of specificity of that specification to online domain. A threshold value of Specificity measure can be used to categorize a new application to belong to a particular domain. Fig. 7 reveals high values of similarity of each online shopping specification to online domain while other specifications show less value to online domain in comparison.

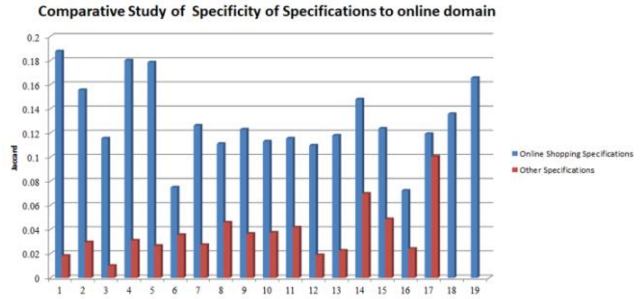

Fig. 7.Comparative Study of Specificity of Specifications to online domain

4) Domain related terms and concepts

Semantic space is very high dimensional and it cannot be plotted as a whole. Thus, some functions are able to plot subsets of semantic space. Related terms and concepts of each domain specially the similarity structure for a given list of words can be visualized by using these functions to create a particular plot. All pairwise similarities amongst the given words in the list are calculated and saved in a cosine matrix. A principal component analysis (PCA), or a multidimensional scaling (MDS) is implemented to this cosine matrix. The resulting matrix is shortened to two or three dimensions. Therefore, a two- or three-dimensional vector is allocated to each of the n neighbours and the input. Fig. 8 and Fig. 9 represent the 3 and 2 dimensional plot of specific set of online shopping domain respectively. Fig. 10 illustrates the most frequent terms of online shopping domain.

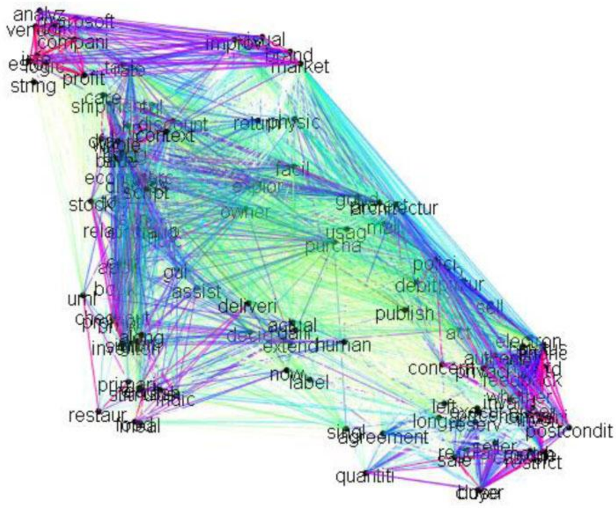

Fig. 8.Three dimensional visualization of similarity structure of specific set of online shopping domain

Retrieval Number: E5849018520/2020©BEIESP
DOI:10.35940/ijrte.E5849.018520
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

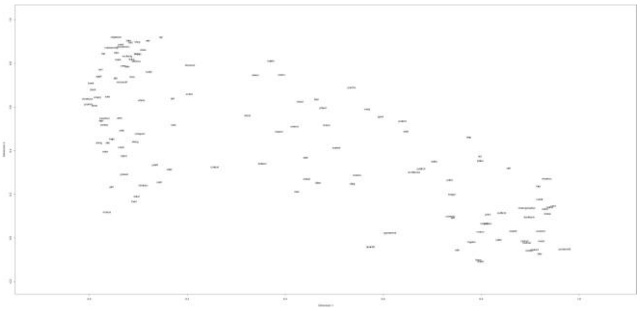

Fig. 9.Two dimensional visualization of similarity structure of specific set of online shopping domain

Fig. 10. The most frequent terms of online shopping domain

V. CONCLUSION

Software intelligence helps complicated software construction by providing insightful information, through analysis of massive volume of data generated as deliverables, throughout the life cycle of a software development system. One of the most important amongst this documentation is a software requirement specification. In this paper, a knowledge base is created using software analytics on software specifications. This knowledge base can be used for validating a new specification for completeness, categorization of software requirement specifications and identifying of major concepts and related keywords. In reality, the document repository can be huge and grow continuously and all the above algorithms can be extended to be applied on big data.

REFERENCES

1. M. Edgar Serna, A. Alexei Serna, and S. Oscar Bachiller, "A framework for knowledge management in requirements engineering," Int. J. Knowl. Manag. Stud., vol. 9, no. 1, pp. 31–50, 2018.
2. J. Ward and A. Aurum, "Knowledge management in software engineering - Describing the process," Proc. Aust. Softw. Eng. Conf. ASWEC, vol. 2004, no. c, pp. 137–146, 2004.
3. P. Carreteiro, J.B. de Vasconcelos, A. Barão, Á. Rocha, "A knowledge management approach for software engineering projects development," In New advances in information systems and technologies, pp. 59-68,Springer, Cham., 2016.
4. R. P. L. Buse and T. Zimmermann, "Information Needs for Software Development Analytics - In Proceedings of the 34th international conference on software engineering, pp. 987-996, IEEE Press, 2012.
5. D. Zhang, S. Han, Y. Dang, J. G. Lou, H. Zhang, and T. Xie, "Software analytics in practice," IEEE Softw., vol. 30, no. 5, pp. 30–37, 2013.
6. J. Han, J. Pei, M. Kamber, "Data mining: concepts and techniques," Elsevier, 2011
7. R. Feldt, F. G. De Oliveira Neto, and R. Torkar, "Ways of applying artificial intelligence in software engineering," Proc. - Int. Conf. Softw. Eng., pp. 35–41, 2018.
8. X. Sun, B. Li, H. Leung, B. Li, and Y. Li, "MSR4SM: Using topic models to effectively mining software repositories for software maintenance tasks," Inf. Softw. Technol., vol. 66, pp. 1–12, 2015.
9. A. E. Hassan, "The road ahead for mining software repositories," Proc. 2008 Front. Softw. Maintenance, FoSM 2008, pp. 48–57, 2008.
10. P. Clements, D. Garlan, R. Little, R. Nord, and J. Stafford, "Documenting software architectures: Views and beyond," Proc. - Int. Conf. Softw. Eng., pp. 740–741, 2003.
11. R. Witte, Q. Li, Y. Zhang, and J. Rilling, "Text mining and software engineering: An integrated source code and document analysis approach," IET Softw., vol. 2, no. 1, pp. 3–16, 2008.
12. A. Aurum, C. Wohlin, "The Fundamental Nature of Requirements Engineering Activities as a Decision-Making Process," Information and Software Technology, vol. 45, no. 14, pp. 945–954, 2003.
13. Shull, Forrest, I. Rus, V. Basili, "How perspective-based reading can improve requirements inspections," Computer, 33(7), pp.73-79, 2000.
14. D. Gefen, J. E. Endicott, J. E. Fresneda, J. Miller, and K. R. Larsen, "A guide to text analysis with latent semantic analysis in r with annotated code: Studying online reviews and the stack exchange community," Commun. Assoc. Inf. Syst., vol. 41, no. 1, pp. 450–496, 2017.
15. T. Xie, S. Thummalapenta, D. Lo, and C. Liu, "Data mining for software engineering," Computer (Long. Beach. Calif)., vol. 42, no. 8, pp. 55–62, 2009.
16. A. E. Hassan and T. Xie, "Software Intelligence: The future of mining software engineering data," Proc. FSE/SDP Work. Futur. Softw. Eng. Res. FoSER 2010, pp. 161–165, 2010.
17. R. P. L. Buse and T. Zimmermann, "Analytics for software development," Proc. FSE/SDP Work. Futur. Softw. Eng. Res. FoSER 2010, pp. 77–80, 2010.
18. T. Menzies and T. Zimmermann, "Software analytics: So what?," IEEE Softw., vol. 30, no. 4, pp. 31–37, 2013.
19. J. Wang, Y. Dang, H. Zhang, K. Chen, T. Xie, and D. Zhang, "Mining succinct and high-coverage API usage patterns from source code," IEEE Int. Work. Conf. Min. Softw. Repos., pp. 319–328, 2013.
20. S. Wang, T. Liu, and L. Tan, "Automatically learning semantic features for defect prediction," Proc. - Int. Conf. Softw. Eng., vol. 14-22-May-2016, pp. 297–308, 2016.
21. R. Kapur and B. Sodhi, "Estimating defectiveness of source code: A predictive model using GitHub content," 2018.
22. H. K. Dam, T. Pham, Ng. Shien Wee, T. Tran, J. Grundy, A. Ghose, T. Kim, C. J. Kim, "A deep tree-based model for software defect prediction," arXiv preprint arXiv:1802.00921, 2018.
23. J. G. Lou, Q. Lin, R. Ding, Q. Fu, D. Zhang, and T. Xie, "Software analytics for incident management of online services: An experience report," 2013 28th IEEE/ACM Int. Conf. Autom. Softw. Eng. ASE 2013 - Proc., pp. 475–485, 2013.
24. Q. Fu, J. G. Lou, Q. Lin, R. Ding, D. Zhang, and T. Xie, "Contextual analysis of program logs for understanding system behaviors," IEEE Int. Work. Conf. Min. Softw. Repos., pp. 397–400, 2013.
25. X. Yu, H. Shi, Z. Dongmei, and X. Tao, "Comprehending Performance from Real-World Execution Traces," Asplos, pp. 193–206, 2014.
26. R. K. Saha, M. Lease, S. Khurshid, and D. E. Perry, "Improving bug localization using structured information retrieval," 2013 28th IEEE/ACM Int. Conf. Autom. Softw. Eng. ASE 2013 - Proc., pp. 345–355, 2013.
27. C. Tantithamthavorn, A. Ihara, and K. I. Matsumoto, "Using co-change histories to improve bug localization performance," SNPD 2013 - 14th ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput., pp. 543–548, 2013.
28. S. Rahman and K. Sakib, "An appropriate method ranking approach for localizing bugs using Minimized search space," ENASE 2016 - Proc. 11th Int. Conf. Eval. Nov. Softw. Approaches to Softw. Eng., no. Enase, pp. 303–309, 2016.
29. A.S.J. Carlos, Gulo, R.P.M. Thiago Rúbio, "Text Mining Scientific Articles uses the R," In Doctoral Symposium in Informatics Engineering, January. 2015.
30. T. Arnold, "A Tidy data model for natural language processing using cleanNLP," R J., vol. 9, no. 2, pp. 248–267, 2017.
31. E. Schubert, A. Spitz, M. Weiler, J. Geiß, and M. Gertz, "Semantic Word Clouds with Background Corpus Normalization and t-distributed Stochastic Neighbor Embedding," no. Section 2, 2017.
32. Welbers, W. Van Atteveldt, K. Benoit, "Text Analysis in R, Kasper," Commun. Methods Meas., vol. 11, no. 4, pp. 245–265, 2017.

33. G. Miner, J. Elder IV, T. Hill, B. Nisbet, D. Delen, A. Fast, "Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications", Elsevier, 2012.
34. L. Torgo, "Data mining with R: learning with case studies," Chapman and Hall/CRC, 2011.
35. I. Feinerer, "Introduction to the tm Package: Text Mining in R," R vignette, pp. 1–8, 2015.
36. B. Van Looy, B. Baesens, T. Magerman, and K. Debackere, "Assessment of Latent Semantic Analysis (LSA) Text Mining Algorithms for Large Scale Mapping of Patent and Scientific Publication Documents," SSRN Electron. J., 2012

AUTHORS PROFILE

Lida Bamizadeh is pursuing her Ph.D in computer science of Savitrabai Phule Pune University. She has received bachelor's degree of the electronic engineering of Islamic Azad University, Kerman Branch of Iran, and M.Tech degree in computer science of Savitrabai Phule Pune University of India. She has more than 2 years of Industrial experience in quality inspection and issue detection as a quality control (QC) supervisor. Her area of interests are Machine learning, Data mining (especially text mining and Natural Language Processing), Database Management System and Software Engineering. She has attended and presented some papers in the National and International conferences.

Dr. Binod Kumar is Director & Professor at Jayawant Institute of Computer Applications (JSPM's Group), affiliated to Savitribai Phule Pune University, India. He is having more than 22 years of experience in various capacities in research, teaching and academic administration. He has obtained PhD (Computer Sc.) in 2010, M.Phil. (Computer Sc.) in 2006, MCA (NIT Jamshedpur) in 1998 and M.Sc.(BHU) in 1995. He worked as Associate Professor at School of Engineering and Computer Technology, Quest International University, MALAYSIA . He is nominated as Subject Expert in Computer Science by University of Pune and M.S. University Baroda for the Selection Committee for post of Professor (MCA). He served as Conference Chairman Committee Member at QUAESTI 2015, The 3rd Virtual Multidisciplinary Conference, Slovakia. He is reviewer of Journals like Elsevier, SpringerPlus and TPC of various IEEE sponsored conferences. He is Editorial Board member of nearly 45 International Journals. He has been associated with Technical Program Committee member (TPC) of nearly 60 International Conferences in India and abroad. He is Senior Member of IEEE Computer Society, Senior Member of Association for Computing Machinery (ACM, USA). He has published nearly 45 papers in International & National Journals/Conferences. His areas of interest are Machine Learning & IoT.

Dr. Ajay Kumar experience covers more than 26 years of teaching and 6 years of Industrial experience as IT Technical Director and Senior Software project manager. He has an outstanding academic career completed B.Sc. App. Sc. (Electrical) in 1988, M.Sc. App. Sc. (Computer Science-Engineering and Technology) in1992 and PhD in1995. Presently, working as Director at JSPMs Jayawant Technical Campus, Pune (Affiliated to Pune University). His research areas are Computer Networks, Wireless and Mobile Computing, Cloud computing, Information and Network Security. There are 74 publications at National and International Journals and Conferences and also worked as expert, appointed by C-DAC to find Patent-ability of Patent Applications in ICT area. Six commercial projects are completed by him for various companies/ Institutions. He holds variety of imperative position like Examiner, Member of Board of Studies for Computer and IT, Expert at UGC.

Dr. Shailaja Shirwaikar has a Ph. D. in Mathematics of Mumbai University, India
and worked as Associate Professor at Department of Computer Science, Nowrosjee Wadia College affiliated to Savitribai Phule Pune University, Pune for 27 years. Currently working as visiting Faculty teaching courses in Data Mining, Machine learning, Design and Analysis of Algorithms, Software Architecture and design Patterns. Her research interests include Soft Computing, Big Data Analytics, Software Engineering and Cloud Computing. She has published more than 30 papers in National and International Journals. She has attended and presented papers in many National and International conferences including EuroPlop 2016 in Germany.