NOISE DETECTION IN SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT USING SPECTRAL CLUSTERING

Patricia G. Manek[1)] and Daniel Siahaan[2)]
[1, 2] Informatics Department, Institut Teknologi Sepuluh Nopember, Surabaya – Indonesia
e-mail: patricia16@mhs.if.its.ac.id [1)], daniel@if.its.ac.id[2)]

ABSTRAK

Spesifikasi kebutuhan perangkat lunak (SKPL) merupakan sebuah spesifikasi fungsional (kapabilitas) dan non-fungsional (keterbatasan, karakteristik, kualitas dan properti) dari sebuah perangkat lunak yang harus dikembangkan berdasarkan kondisi atau kapabilitas yang harus dilengkapi perangkat lunak atau dibutuhkan oleh pengguna. Bagi pengembang, dokumen SKPL digunakan sebagai referensi di setiap tahap pengembangan. Dalam proses pembuatan dokumen SKPL, kemungkinan ada beberapa hal yang cacat. Salah satu yang dikenal sebagai The Seven Sins of Specifier adalah noise. Noise pada SKPL dapat terjadi selama proses SKPL. Noise ini diterjemahkan dalam bentuk informasi yang tidak relevan dengan daftar kebutuhan pengguna. Penelitian ini bertujuan untuk mendeteksi noise dalam dokumen SKPL. Proses deteksi noise menggunakan spectral clustering untuk memisahkan dan mengelompokkan noise (yang dianggap sebagai outlier) selain dari kebutuhan lainnya. Pada metode ini, dengan menggunakan natural language processing bahwa setiap pernyataan kebutuhan direpresentasikan sebagai sebuah vektor frekuensi dari kata-kata yang unik yang muncul dalam pernyataan kebutuhan. Sebuah cluster yang memiliki jarak rata-rata paling besar antara data setiap individu dan centroidnya dianggap memiliki noise. Hasil uji coba menunjukkan bahwa metode yang diusulkan memiliki sensitivitas tinggi yaitu 1.0 untuk data riil dan data sintetis. Namun, metode ini memiliki spesifisitas yang rendah untuk data riil sebesar 0.19 dibandingkan dengan data sintetis sebesar 0.83.

***Keywords**: noise detection, software requirements, spectral clustering, natural language processing.*

ABSTRACT

Software Requirements Specification (SRS) is a specification of functional (capabilities) and non-functional (constraints, characteristics, qualities, and properties) of a software that should be developed based on the conditions or capabilities that must be equipped the software or required by the users. For developers, SRS document is used as a reference in every stage of development. In the process of making the SRS document, there could be some defects. One of those defects, which were called as The Seven Sins of Specifier, is noise. Noise in the SRS could occur during the software requirements specification process. It lateralized in the form of information that is not relevant to the list user requirements. This study aims to detect noise within the SRS document. The noise detection uses spectral clustering to separate and group noise (which is considered as outlier) aside from the rest of requirements. In this method, by using natural language processing each requirement statement is represented as a vector of frequencies of unique words appeared within the requirement statement. A cluster that has the widest average distance between its individual to its centroid is considered to have noise. The experimentation shows that the proposed method has a high sensitivity, i.e. 1.0, for both real and synthetic data. Nevertheless, the method has a low specisivity for real data, i.e. 0.19, compare to the synthetic data, i.e. 0.83.

***Keywords**: noise detection, software requirements, spectral clustering, natural language processing.*

I. INTRODUCTION

IN requirements specification stage, software development becomes an important process according to Boehm and Basili [1]. Fixing errors during the implementation process would spend exponentially higher cost compared to fixing the errors during requirements and design processes [2]. Therefore, developers should pay more attention on requirements process.

There are three approach methods that can be used to analyze requirements specification, namely: natural language, semi-formal language, and formal language. Natural language is a requirements specification formulation approach which uses common language. Semi-formal language uses a graphical language and textual explanation within. Formal language uses mathematical concepts such as finite-state machines. According to Rossi [3], as much as 71,8% software requirements specification documents have been created using natural language approach. This is because the approach is easily understandable for software engineers as well as the customers with various background.

Meyer [4]–[8] explains that the process of Software Requirements Specification using natural language has several weakness compared to formal language. The weakness is caused by seven common mistakes which are often done by software developers. Meyer formulated the seven mistakes as “The seven sins of specifier”. One of seven sins of specifier is noise. Noise may appear in Software Requirements Specification (SRS) document. In a

requirements specification, noise occurs when software developers add some irrelevant information to the overall software requirements.

As the development of information technology, noise on the data has a lot of meaning. Noise can represent useless data, imperfect data, or data that cannot be read by a computer [9]. Noise can affects the results of data mining analysis. Within this study, a noise is considered as any statement within the requirements specification that is not relevant to the system being developed or is not relevant to the requirements engineering process [2], [10]. There are several studies about noise or outlier detection of text data. Detection of text deviations has been developed using conceptual graphs [11]. The study provides graph visualization to the deviations that occur in a text data.

Clustering methods can be used to detect noise or outliers in the data [12]–[17]. Clustering classifies objects which have similar characteristics within specific clusters. Outliers is an object that has dissimilar characteristics with respect to the rest of its neighbors. Cai et al [12] developed an application based on spectral clustering to detect a sentence which is considered as noise within a summarization section within a document. The output of the application is the outlier sentences within the summarization section. Furthermore, another study used one-pass clustering to detect outliers on existing datasets [18]. The method divides the dataset into several hyper-spheres that have almost the same radius. Clustering results are divided into two groups, namely: "normal" or "outlier" based on its outlier factors. In addition, the k-Means clustering method has also been developed to detect noise [13], [19], [20].

The noise detection based on spectral clustering in a textual document has been developed as a preprocessing stage in text summarization method [12]. The study used spectral clustering algorithm to detect noise in sentences. Spectral clustering method with noise detection [21] was considered in detecting noise in software requirements. The study uses a synthetic data for the experimentation. Spectral clustering methods have shown better performance than k-Means clustering or clustering based on hierarchy [21]. One of the advantages of spectral clustering is that it can adapt to various forms of data. Clustering methods can also be implemented to look for data that have fewer characteristic similarities to other data or can be considered as noise.

This research proposes a method for detecting noise in Software Requirements Specification (SRS) by using spectral clustering. This research is intended to provide information for developers about the characteristics of a set of requirements statement in a Software Requirements Specification (SRS).

II. Research Methodology

There are some three main steps in this process, i.e. text preprocessing, spectral clustering, and outlier selection.

A. Text Preprocessing

Software Requirements Specification (SRS) document contains a list of requirement statements. Before the outlier detection process, each statement needs to be textually preprocessed. The preprocessing involves tokenizing, lowercase converting, stopword removing, and lemmatizing sentences of each requirement statement. For the sake of illustration, let's consider the following requirement statement.

Submit jobs with the associated deadline cost and execution time.

Basically, tokenizing is a process that breaks a text into its individual linguistic units. The statement would be first tokenized into the following tokens.

Submit / jobs / with / the / associated / deadline / cost / and / execution / time /.

Then, each token is converted into lowercase.

submit / jobs / with / the / associated / deadline / cost / and / execution / time /.

The next process is removing stopwords. Stopword removal is a text preprocess that removes any token of word that listed in a predefined stopword corpus. The stopword removal produces the following tokens.

submit / jobs / associated / deadline / cost / execution / time

The final process of text preprocessing is lemmatizing each token. The lemmatization would identified any inflectional form of word and returns its base or dictionary form of the word. In this study, it changes plural noun into singular noun and inflected ending of verb into its base form.

submit / job / associate / deadline / cost / execution / time

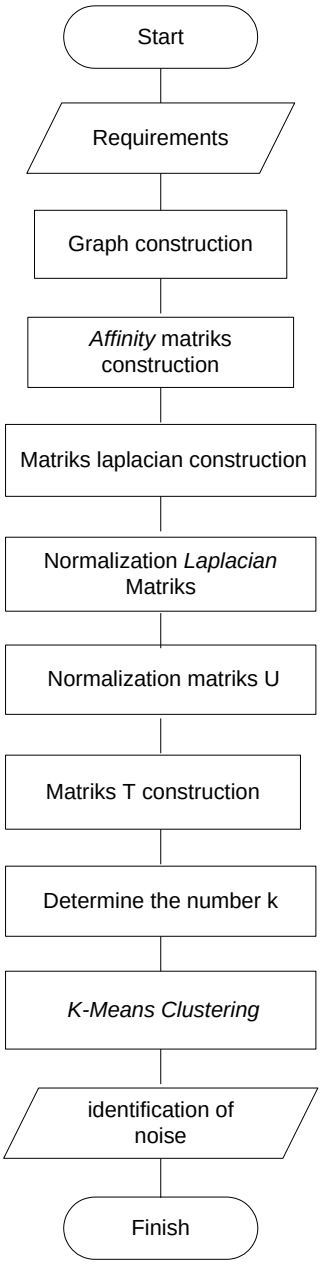

Fig. 1. Spectral Flowchart

B. Spectral Clustering

The next step is clustering the set of preprocessed requirement statements using Spectral Clustering. The clustering should group statements that describe closely related functions into the same cluster. Assume $G=(V,W)$ is an undirected weighted graph with node V contains points $\{v_i \epsilon \mathbb{R}d|i=1,2,...,n\}$, where d is the amount of features in TF-IDF. Matrix W is a symmetrical matrix where the elements are values of gaussian similarity between data in V. D is a diagonal matrix where $d_{ii}=\Sigma_j w_{ij}$, then make a normalized Laplacian matrix (L).

$$L = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \quad (1)$$

Then, it calculates eigenvector of Lnorm. It takes only first k-eigenvector and creates matrix $X=[x_{ij}]nxk$ from them, where k is the number of clusters. Next, it normalizes the length unit for each row of matrix X. Each row of X corresponds to an initial data of TF-IDF. It calculates k-means of matrix X, where each row represents a data point in k-dimension. The clustering process would produce k clusters, that contain data point X. The detail of spectral clutering is described in Figure 1.

C. Outlier Selection

The clusters produced from the previous result are supposed to separate the cluster that contains noise statements from the rest of the clusters that contains requirement statements. The noise statements are considered the

TABLE I
DATASET

ID	Project Name	Num of Reqs
DA-1	Submit Job	13
DA-2	System Administrator	17
DA-3	Archive Administrator	39
DA_4	SPG Aplication	6
DA-5	System Administrator Staf	33
DA-6	Software Develompment	65
DA-7	Display System	106
DA-8	Internet Access	64
DA-9	Meeting Initiator	13
DA-10	Online System	17
DA-11	Library System	86
DA-12	IMSETY System	70
DA-13	Manage Student Information	24
DA-14	PHP Project	75

TABLE II
CONFUSION MATRIX OF NOISE DETECTION

Annotator	Method		
	Noise	Not-Noise	Total
Noise	TP	FN	TP+FN
Not-Noise	FP	TN	TP+TN
Total	TP+FP	FN+TN	TP+FP+FN+TN

outlier individuals within the dataset. The noise statement should have different characteristics compare to any group of requirement statements that being grouped from the clustering process. It means that the noise statements, as individuals, should be relatively farther from their centroid compared to those requirement statements from the rest of the centroids. The centroid where the noise statements resided should also be relatively farther than the rest of centroids. Therefore, this study select cluster that has the largest distance as the outlier clusters. That means that this outlier cluster contains noise statement.

III. RESULTS AND DISCUSSION

For the experimentation, this study uses 648 manually extracted software requirement statements from 14 SRS documents. Table I shows the dataset used in this study. Each document refers to different project from various problem domains. Each requirements statement labeled by 3 human annotators. The labels are Boolean type values. An annotator would label a statement with 1 (true) if and only if he/she thought that the statement is a noise. An annotator would label a statement with 0 (false) if and only if he/she thought that the statement is a relevant requirement statement. In this study, the method was considered as the fourth annotator that would be evaluated based on its reliability with respect with the human annotators. Therefore, the same labeling rules applied to the method. The reliability of the method would be measured with each annotator using a confusion matrix. Table II shows the confusion matrix used in this study.

The experimentation scenario follows these steps:

a. Store each preprocessed statement of each dataset into a single .csv document. Each statement is separated by a line.
b. Create a questionnaire form for each dataset. Each human annotator labels each statement within the questionnaire with 1 (noise) or 0 (not-noise, good requirement statement). Validate the result and aggregate the answers from all annotators based on majority rule.
c. Cluster each dataset (.csv) using spectral clustering algorithm. Select the outlier cluster. Based on the selected outlier cluster, label each statement.
d. Measure the reliability between method and each human annotator.

The noise prediction result generated by the proposed method was evaluated using kappa statistics. It measures the reliability of method with respect to the human annotator. The measurement is expressed by a kappa value [22]. Table II shows the confusion matrix used for calculating kappa coefficient value. Kappa coefficient measures the degree of agreement of 2 assessments.

In this study, the value of kappa coefficient used to represent consistency between noise prediction and noise assessment by human annotators. This measurement also used to determine the consistency of noise assessment among the human annotators. Calculation of the kappa coefficient requires the number of events of assessor 1 and assessor 2 in giving a noise assessment to the software requirements statements. In this study, the assessor 1 and assessor 2 can either be the noise detector, and the human annotators, or one human assessor and other human annotators. Assessor 1 and assessor 2 can provide the requirements statements a true or false value as a noise assessment. Here is the explanation of kappa statistics calculation input:

- P_{tp} = The number of occurrences of human annotator and method (second annotator) give true values (TP) to the given requirement statement.
- P_{fn} = The number of occurrences of human annotator gives a true value and method (second annotator) gives a false value (FN) to the given the requirement statement.
- P_{fp} = The number of occurrences of human annotator gives a false value and method (second annotator) gives a true value (FP) to the given the requirement statement.

P_{tn} = The number of occurrences of human annotator and method (second annotator) give false values (FF) to the given requirement statement.

Based on GWET-AC1, the kappa value (κ) can be calculated if the number of assessments of assessor 1 and assessor 2 is known. First the probability of the observed agreement between the two annotators, or Po is calculated by (2).

$$P_o = \frac{P_{tn} + P_{tn}}{P_{tp} + P_{fn} + P_{fp} + P_{tn}} \quad (2)$$

Then the probability of the two annotators give a random value of true, or we can call it P_T, can be calculated as well as the value of P_F for the value of false. The probability value of random agreement, or P_e, can be calculated by (3). Thus, the coefficient value of kappa (κ) can be calculated by (4).

$$P_e = P_T + P_F \quad (3)$$

$$\kappa = \frac{P_o - P_e}{1 - P_e} \quad (4)$$

As already mentioned, to evaluate the performance of the proposed method, this study compares the annotations by the human annotators with the annotations by the method. The experiment was done by calculating the kappa coefficient value on the noise assessment made by both noise detector method and the human annotators. In this study there are 3 human annotators involved. Each annotator was given 14 questionnaires. Each questionnaire contains requirement statements of a specific software project. The annotator annotated each statement of a given questionnaire manually. In each questionnaire, a SRS document was also attached to allow the annotator to grasp the mission and business background of the developed system.

The selection of human annotators was based on their professional and academic experiences in the field of

TABLE III
KAPPA VALUE BETWEEN NOISE PREDICTOR & HUMAN ANNOTATORS, INDIVIDUALLY AND MAJORITY

No	Comparison	Kappa Value
1	Noise Predictor & Human Assessor 1	0.4308
2	Noise Predictor & Human Assessor 2	0.4082
3	Noise Predictor & Human Assessor 3	0.3928
4	Noise Predictor & Human Assessor Majority	0.4426

TABLE IV
KAPPA VALUE AMONG THE THREE HUMAN ANNOTATORS

No	Comparison	Kappa Value
1	Human Annotator 1 & 2	0.2068
2	Human Annotator 1 & 3	0.3245
3	Human Annotator 2 & 3	0.5390

TABLE V
KAPPA VALUE AMONG THE THREE HUMAN ANNOTATORS

Dataset	Num -Req	Num Clusters	Kappa Score		Gwet's ACI		Sensitivity		Specificity		F1-Score	
			A	B	A	B	A	B	A	B	A	B
DA-1	13	2	-0.13	0.05	0.58	-0.35	0.00	1.00	0.75	0.25	0.00	0.18
DA-2	17	2	-0.10	0.03	0.70	0.51	0.00	1.00	0.25	0.19	0.00	0.13
DA-3	39	2	0.07	-0.05	0.41	-0.07	1.00	0.00	0.60	0.3z9	0.11	0.00
DA-4	6	2	-0.29	0.18	0.02	0.03	0.00	1.00	0.06	0.04	0.00	0.04
DA-5	33	2	0.00	0.00	0.93	-0.87	0.00	0.00	0.93	0.06	0.00	0.00
DA-6	65	2	-0.16	0.06	0.33	-0.15	0.04	0.95	0.81	0.14	0.06	0.51
DA-7	106	2	-0.23	0.12	0.36	-0.10	0.00	1.00	0.65	0.35	0.00	0.32
DA-8	64	2	-0.11	0.02	0.75	-0.59	0.00	1.00	0.91	0.09	0.00	0.24
DA-9	33	2	-0.04	0.00	0.90	-0.82	0.00	1.00	0.93	0.06	0.00	0.06
DA-10	17	2	0.01	-0.01	0.19	0.06	0.05	0.05	0.53	0.47	0.02	0.18
DA-11	86	2	-0.04	0.02	0.53	-0.29	0.25	0.78	0.71	0.29	0.13	0.20
DA-12	70	2	0.00	0.00	0.79	-0.61	0.00	0.00	0.82	0.17	0.00	0.00
DA-13	24	2	-0.24	0.19	0.10	0.16	0.00	0.00	0.52	0.48	0.00	0.35
DA-14	75	2	0.00	0.00	0.95	-1.00	0.00	0.00	0.96	0.00	0.00	0.00
	Average		-0.13	0.05	0.58	-0.35	0.00	1.00	0.75	0.25	0.00	0.18

software requirements engineering. All human annotators have academic background in software engineering. They have experience in teaching requirements engineering course in university level. They have been working on software requirements specification document for at least the last five years.

The noise prediction result from the noise detection system is compared with the noise assessment from each human annotator and the majority answer of the three human annotators, as shown in Table III. In the table, the consistency between the noise prediction and the noise assessment from the human annotators is still very low. The highest consistency is resulted from the noise prediction and the majority noise assessment from the three human annotators, with kappa coefficient value 0.4426.

A comparison of reliability assessment among the three human annotators had also been done to deepen the noise detection evaluation information in the software requirements specification document, as shown in IV. Through this experiment scenario, the consistency of the noise assessment among the three human annotators can be concluded.

Each of SRS (Software Requirements Specification) documents has different structure of natural languages. The different structure of natural language in the SRS (Software Requirements Specification) documents could be the reason why the kappa values of each dataset was at the low level. Table IV is one example of a software requirements statements that produces a kappa value between the noise prediction and the noise assessment of all three human annotators by majority. As in the table, the requirements statements that produce high coefficient kappa value, has a simple structure and has a short length.

To further analyzed the result, Table V shows the detail reliability measurements of each dataset. The table shows five measurements, i.e. kappa score, kappa GWET's AC1, sensitivity, specificity, and F1-score. The kappa score and kappa GWET's AC1 measures reliability of annotators. The sensitivity measures the ability of the method to correctly identify noise statements. The specificity measures the ability of the method to correctly identify not-noise statements. Thus, F1-score measures the balance between sensitivity and specificity of the method. This experimentation also compared two policies on selecting outlier cluster. Column 'A' refers to the policy where a cluster with the furthest distance is the outlier cluster (proposed in this method). While column 'B'refers to the policy where a cluster with the closest distance is the outlier cluster. The result shows that in general by using the further distance policy, the method would perform better. This indicate that clustering process tends to group noise statement separated from the rest of the requirement statements. Nevertheless, the balance between sensitivity and specificity is low. This may be due to two reasons. First, some of the statement is relevant to the system, but not a requirement. During the requirements engineering process, engineer might add statements related to project management (e.g. project schedule, human resource, deliverables, etc), design (e.g. approach, modeling language, etc.), implementation (framework, programing language, vendors, etc.), or testing (e.g. tester, test methods, test data, etc.) These statements may have close distance to the rest of requirement statement, but they are also a noise. Second, the text preprocesses consider all types of POS. However, a requirement statement primarily contains two important parts, i.e. actor and action. An actor is formed by a noun phrase. An action is formed by a verb phrase. Therefore, many unnecessary POS were included in the sentence

vector that shape the affinity matrix. This cause that the similarity value between sentences are not well distributed between noise and not noise.

Based on the above experimental results, the noise prediction and the noise assessment from the three human annotators yields very low kappa value. Thus, the noise prediction has low consistency with the noise assessment from human annotators. This means that the method is considerably in fair agreement with the human annotators. This low consistency could be the result of the structure of natural language used on the software requirements statements. From further observation on each statement, the experimentation also indicates that a simpler structure sentences would be more accurately predicted than the complex ones.

IV. CONCLUSION

The noise detection in the software requirements specification has benefits in software development. The benefits of noise detection in the software requirements include avoiding errors as early as possible so as to reduce repair costs and can provide information about software requirements priorities.

Spectral clustering methods can be used to detect noise in software requirements. However, there are several factors that affect the performance of noise detectors, such as sentence structure, sentence length, and number of software requirements present in a software requirements engineering document. The experiment in this research uses kappa coefficient value which measure the level of agreement between noise prediction by noise detector and noise assessment by 3 human annotators. The resulting kappa coefficient is 0.4426 which is still unsatisfactory.

Further study may focus on adding classification process before the clustering process in order to improve the performance of the method. The classification process would separate the non-requirement but project related statement from the rest of requirement statements. In addition, further research is required to improve the text preprocesses in order to make sure that only actor or action related tokens is included in the sentence vector.

V. RECOMMENDATION

For the further research, the analysis of sentence context in detecting noise can be done. The noise rating would be better if done by an expert in software requirements engineering.

VI. ACKNOWLEDGEMENT

The authors thank Informatics Department, Institut Teknologi Sepuluh Nopember who support this research and also Universitas Timor for their financial support to carried out this research.

REFERENCES

[1] B. Boehm and V. R. Basili, "Software Defect Reduction Top 10 List," *Computer (Long. Beach. Calif).*, vol. 34, no. 1, pp. 135–137, 2001.
[2] D. Siahaan, *Analisa Kebutuhan dalam Rekayasa Perangkat Lunak*, 1st ed. Yogyakarta: Penerbit Andi, 2012.
[3] A. Rossi, "Incentives in managerial compensation: a survey of experimental research," *ROCK Work. Pap.*, 1999.
[4] B. Meyer, "on Formalism in Specifications.," *IEEE Softw.*, vol. 2, no. 1, pp. 6–26, 1985.
[5] W. Purnomo and D. O. Siahaan, "Pendeteksian Overspesification Pada Dokumen Spesifikasi Kebutuhan Perangkat Lunak," *J. Inspir.*, vol. 7, no. 1, pp. 1–9, Jun. 2017.
[6] D. Enda and D. Siahaan, "Rekomendasi Perbaikan Pernyataan Kebutuhan yang Rancu dalam Spesifikasi Kebutuhan Perangkat Lunak Menggunakan Teknik Berbasis Aturan," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 5, no. 2, p. 207, May 2018.
[7] F. V. Sari Sahadi, D. O. Siahaan, and U. L. Yuhana, "PENDETEKSIAN ISTILAH BERBEDA PADA DOKUMEN SPESIFIKASI KEBUTUHAN PERANGKAT LUNAK (SKPL)," *SCAN - J. Teknol. Inf. dan Komun.*, vol. 10, no. 3, pp. 9–16, 2015.
[8] D. Siahaan and I. Umami, "Natural Language Processing for Detecting Forward Reference in a Document," *J. Technol. Sci.*, vol. 22, pp. 138–142, 2011.
[9] L. Sunitha, M. B. Raju, and B. S. Srinivas, "A Comparative Study between Noisy Data and Outlier Data in Data Mining," pp. 1–3, 2013.
[10] P. Gelu, R. Sarno, and D. Siahaan, "Requirements Association Extraction based on Use Cases Diagram," *Lontar Komput. J. Ilm. Teknol. Inf.*, vol. 9, no. 1, pp. 11–19, May 2018.
[11] M. Montes-y-gómez, A. F. Gelbukh, and A. López-lópez, "Detecting Deviations in Text Collections: An Approach Using Conceptual Graphs," *Mex. Int. Conf. Artif. Intell.*, pp. 176–184, 2002.
[12] X. Cai, R. Zhang, D. Gao, and W. Li, "Simultaneous Clustering and Noise Detection for Theme-based Summarization," *Proc. 5th Int. Jt. Conf. Nat. Lang. Process.*, pp. 491–499, 2011.
[13] Y. Zhou, H. Yu, and X. Cai, "A Novel k-Means Algorithm for Clustering and Outlier Detection," *2009 Second Int. Conf. Futur. Inf. Technol. Manag. Eng.*, pp. 476–480, 2009.
[14] A. Puspaningrum, D. Siahaan, and C. Fatichah, "Mobile App Review Labeling Using LDA Similarity and Term Frequency-Inverse Cluster Frequency (TF-ICF)," in *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 2018, pp. 365–370.
[15] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection," *ACM Trans. Knowl. Discov. Data*, 2015.
[16] B. Perozzi, L. Akoglu, P. Iglesias Sánchez, and E. Müller, "Focused clustering and outlier detection in large attributed graphs," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, 2014.
[17] G. Gan and M. K. P. Ng, "k-means clustering with outlier removal," *Pattern Recognit. Lett.*, 2017.
[18] S. Jiang and Q. An, "Clustering-Based Outlier Detection Method," *2008 Fifth Int. Conf. Fuzzy Syst. Knowl. Discov.*, pp. 429–433, 2008.

[19] R. Pamula, J. K. Deka, and S. Nandi, "An Outlier Detection Method Based on Clustering," *2011 Second Int. Conf. Emerg. Appl. Inf. Technol.*, pp. 253–256, 2011.
[20] G. Gan and M. K. P. Ng, "K-Means Clustering With Outlier Removal," *Pattern Recognit. Lett.*, vol. 90, pp. 8–14, 2017.
[21] Z. Li, J. Liu, S. Chen, and X. Tang, "Noise Robust Spectral Clustering," *Comput. Vision, 2007. ICCV 2007. IEEE 11th Int. Conf.*, pp. 1–8, 2007.
[22] J. Cohen, "A Coefficient of Agreement for Nominal Scales," *Educ. Psychol. Meas.*, vol. 03, 1960.