International Journal on Advanced Science Engineering Information Technology

Vol.12 (2022) No. 4
ISSN: 2088-5334

Software Traceability in Agile Development Using Topic Modeling

Nuraisa Novia Hidayati [a,*], Siti Rochimah [a], Agus Budi Rahardjo [a]

[a] *Department of Informatics Engineering, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia*
Corresponding author: [*]nunohida@gmail.com

***Abstract*— Tracing the implementation of requirements for making better software identifies whether the application fulfils users' desires; progress of development; problematic areas in the testing process, and how far those apply to the source code. In this paper, the software development method we studied was the agile method, Extreme Programming (XP). The artifacts in the agile approach considered vital include the requirement documents, test documents, and source codes. We used Topic Modelling to map the content similarities from those documents to make trace links. The three topic modelling methods we compared consist of Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA), and Non-negative Matrix Factorization (NMF). The NMF method proved itself the most stable, with an accuracy value of 67% for the requirement, 59% for testing, and 48% for defect lists. The second application results proved more accurate with 70%, 79%, and 54%. Although NMF lost to LSA in the second application (LSA achieved an accuracy of 79%, 84%, and 56%), the precision and recall values showed almost similar results. We successfully found the link in the source code based on keywords extracted from each topic. This research provides a way of explaining the requirement in detail, simplifying it for tracing purposes such as the consistent use of terms, technical details inclusion, and mentioning all the variables involved. In the future, sentence structure and synonyms need recognition as part of pre-processing to build better trace links.**

***Keywords*— Software traceability; Agile; topic modeling; latent semantic analysis; latent dirichlet allocation; non-negative matrix factorization.**

Manuscript received 10 May 2021; revised 24 Sep. 2021; accepted 18 Oct. 2021. Date of publication 31 Aug. 2022.

I. INTRODUCTION

There are quick changes or additions to requirements in the agile application development process. This situation happens because the customer is the part of the development team who provides input and adjustments. Very intense communication processes and the simplification of documentation often make it difficult for developers to track the progress of implementing the application's overall requirements; therefore, requirements tracking is necessary to ensure that application development is running on the right track.

Our previous research paper found a close correlation between story points on requirements and integration testing with the appearance of defects [1]. Story point changes have a more significant impact on the appearance of defects rather than an addition to story points. Apart from story points, integration testing also strongly correlates with the appearance of defects. It was found that malfunctions detected and handled by integration testing could recur when multiple users used the application. Based on that fact, tracing was carried out on three documents, namely story points of requirement, test cases, and source code.

Requirements traceability is the process of tracking the application of requirements to a variety of software development artifacts. Automated tracing with good efficacy and scalability needs to be implemented to form the correct trace links. The traceability of requirements is carried out based on the formulation of written language structure. The manual application of requirements tracing requires effort and time, especially with a high complexity level. One of the technologies often used to create trace links is information retrieval [2].

In this paper, the information retrieval method we use is topic modeling included in the clustering process. The document was checked for the similarity of constituent words and identify the topic grouping based on the compiler words. This grouping facilitated the formation of the traceability link. In this method, there is no need for training data. So that grouping in limited artifact documents can still be carried out without fear of an overfitting model. The information structure strongly influences the results of applying information retrieval technology in the artifacts and frequent evaluations to see whether the trace links created properly are Precision and Recall [2].

There are seven types of artifacts often used as sources of information when researching software traceability: features, requirements, source code, architecture, components, tests, and variability models [3]. This paper studies only three types – requirements documents, test documents, and source code. The domain is the similarity of keywords in the development of each module.

Two topic modeling methods, namely Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA), can retrieve information and determine possible linkages [4]. Term frequency-inverse document frequency (TF-IDF) supports both ways to map terms whose existence in the document is represented in the form of metrics. The TF-IDF method combined with cosine similarity is not enough to see similar words in application artifact documents. The application of LSA and LDA can provide better results. The combination of TF-IDF as input for LSA and LDA is expected to improve the two modeling algorithms' performance for the topic.

LSA and LDA have also been implemented to create a search link between the requirements document and the source code. It aims to find the source code that needs to be changed when the requirement is changed. However, both methods still produce low precision and recall values. The recall value obtained is 0.23, and the precision value obtained is 0.305 [5]. This value is obtained because the requirements document describes the process in general. Not many technical terms appear in the requirements document, so finding similar terms or words in the source code is rare. Requirements documents cannot be directly mapped based on the similarity of terms to the source code. We need other documents with more technical terms used in the source code and general terms in the requirements document.

Another comparison between Probabilistic Latent Semantic Analysis (PLSA) and LDA was adopted to track business processes and software components. The recall value on PLSA is higher than LDA for relevant value because software components are suitable for business process activities. The LDA precision value is higher than PLSA because calculating LDA's relevant value is specific. Optimal results can be taken if the dataset has a particular class based on specific activities [6]. Previous studies discuss the comparison of those three methods with their strength and limitation. LSA has limitations in handling polysemy, PLSA tends to be overfitting, and LDA has difficulties finding correlating words or topics [7].

Topic modeling with the NMF method also found a significant topic result related to disease identification; the correlation between the topic and the value for certain variables showed a good correlation [8]. LDA and NMF are considered to be the topic modeling methods that produce the most valuable output on short text compared with other methods like LSA, Principal Component Analysis (PCA), and random projection (RP) [9]. From the two previous studies, both LDA and NMF seem to stand out by finding topics in short texts that are relatable and easy to conclude.

Meanwhile, Non-negative Matrix Factorization (NMF) produces a higher quality topic spread than LDA with the same experimental setting across multiple experimental data with short text documents [10]. Data with short text contained a small number of words and varied, and the recurrence of the exact words in one sentence occurred very rarely. There are also not many combinations of word similarities in comparing one short text data with other short text data. The lack of information about combining words like this cannot be appropriately processed with probabilistic models such as LDA. Gibbs sampling on the LDA provided a large variety of learning and inference in short texts.

We propose to make traceability from the three selected documents as follows: story points on requirements, test cases, and defect lists, using topic modeling. In topic modeling, documents were grouped into particular topics that save time compared to manually. The three topic modeling methods to be compared are LSA, LDA, and NMF. These three methods have their advantages when used on various datasets. The author looked for a topic modeling method that best fits our software development dataset. We identified the words that compiled each topic according to the best model result. Those words can be the input to trace the source code. Each topic modeling was different in determining the optimal number of topics. Topic modeling such as LSA suggests fewer topics than LDA and NMF on the same data [11]. In this study, the actual number of topics is known, and we are looking to model which topics can group data into predetermined topics appropriately.

Based on a previous study, the tracing model uses several points: source, destination, meaning, assumption, consequences, pre-process, process, and tool [12]. In this paper, the source requires a testing document and source code as the destination. Meanwhile, the meaning is dependency, assuming that the topic has been determined based on the agile development module. The consequences of the application process are measured using precision and recall. The raw data was structured first through pre-process to be used further. The process is the application of modeling topics to each artifact using python as the tool.

This study's contribution is to improve the requirements' explanation and make it easier to trace the implementation of requirements down to the source code for agile software development. This tracing can later be used for better defect handling. It can be a way to facilitate application management and improve application quality because we can be aware of the suspected defect location in the source code, other code that is affected by it on the test case step, and other modules regarding the story points from the requirement.

II. Materials and Method

The written language for the requirements, test cases, and comments on the source codes were Indonesian. Only a few technical terms were written in English. The requirements here would be divided into general and technical terms. In the test case, a detailed step was taken. There were very striking differences between the requirements document and the test case, and the appearance of the same word was infrequent. Therefore, topic modeling was carried out separately with the same topics. Data were taken from two applications with the same programming language but using different frameworks. The first application used the PHPMaker generator, and the second application used Yii2. Both were simple applications that were not too complex.

A. *Prepare The Data*

All written documents were cleaned of symbols, numbers, and single characters that did not provide any information. Then a stop word removal – which was a list of words in automatic indexing to filter out words that have no real purpose in describing the document content, especially in searches – was performed. Tokenization was then carried out, which broke the flow of text into words, phrases, symbols, or other meaningful elements called tokens. The purpose of tokenization was the exploration of words in a sentence. Stemming was the last thing that combined variant forms of words into a general representation of the root words.

1) Requirement Data: The requirements were obtained from a business analyst to identify the stages of business processes applied to the application to support user performance. Meanwhile, technical needs would arise from the System Analyst and input from programmers. Technical requirements for more details regarding what attributes were involved and how they were processed were translated to make applying them to the source code easier. Such as explaining employee data, grouping documents, storing processes into a database, etc. The pre-processing words on the user story were included in Table 1 in the Requirement column, and each user story came from the existing meeting and discussion notes. The document had a code for easy identification. Requirements status for user stories were saved as high, medium, or low priority. Needs also fell into two categories – changes or additions, and the last column was which group of modules these needs were.

TABLE I
REQUIREMENT DATA

Code	Requirement	Dependent	Status	Change/Add	Modul
M3	• Every employee's personal document is stored neatly and electronically organized • b. Every employee document is always updated		High	-	1
M7	The official documents that have been published and uploaded are not deleted from the e-filling database.	M3	High	Change	1

2) Test Case Data: The results of pre-processing words in the test were entered in Table 2. The table just stored data for step detail, and the expected results from the test case document would be separately stored because they would be processed independently. The pre-processing step detail results for each test would be given a code as an identity to make it easier to trace the testing document source. In the last column, it was entered into which module group these needs were. In the latest column was which group of application modules the test was.

TABLE II
TEST CASE DATA

Code	Step	Modul
P5	Choose the file format for storage	1
P6	Save files in the file repository in the application	1

3) Defect and Bug Data: The results of pre-processing words on bugs, defects, and expected results on testing were included in Table 3 in the content column. The difference was whether the content was a bug, defect, or expected result to be entered into the type. Each word in the content was given an identity code to make it easier to identify which test document it came from. The last column was which module group it belongs to.

TABLE III
DEFECT AND BUG DATA

Content	Type	Test Case	Modul
File saved with input name and appropriate file extension, but file contents are corrupt. The last stage failed	Bugs	P3	1

4) Source Code Data: Source code was also collected but not through pre-processing in Table 4. A necessary function code was entered in the source code column; meanwhile, the name of the file was entered in the name column. The last column was which module group it belongs to.

TABLE IIIV
SOURCE CODE DATA

Source Code	Name	Modul

<?php
//Mencari nama pegawai dari database
//mysql_connect("localhost","root","");
//mysql_select_db("edoc");
include_once('koneksi_db.php');
$nip18 = $_GET['nip18'];
$nama = mysql_query("SELECT nama_nongelar FROM ms_pegawai WHERE nip18='$nip18' order by nip");
while($k = mysql_fetch_array($nama)){
echo "<option selected value=\"".$k['nama_nongelar']."\">".$k['nama_nongelar']."</option>";
}
?>
	ambilnama.php	1

B. *Topic Modeling*

First of all, all sentences in the document list were vectorized using TF-IDF using Unigram, bigram, or trigram. The three topic modeling algorithms based on word vectorization input identified all text data with the topic and the constituent words' similarity. It should be underlined that all words match the original document without stemming; this facilitates the topic interpretation easiness. The overall topic modeling process, as shown in Fig. 1.

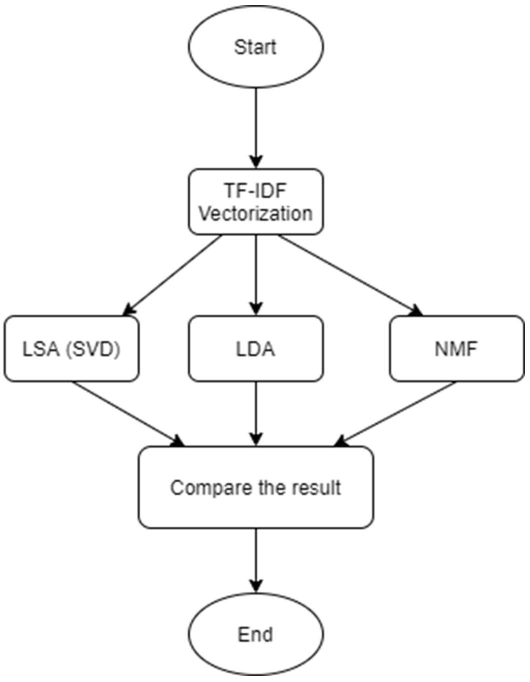

Fig. 1 Topic Modeling Stage

Every text document consists of words that are related to each other. When we needed to find out the document's topic, it was necessary to identify the words that represent it. In each document, words that were considered essential and a representative would be given weight. This search and weighting of words was a necessary process. The search and weighting process of words used was TF-IDF in this research. In the TF-IDF algorithm, TF was the frequency of terms that appear in a document, IDF was a document distribution with certain words from the entire data set [13].

$$w_{td} = TF_{td} \times log\left[\frac{N}{DF_t}\right] \quad (1)$$

- TF_{td} was the number of appearances t in document d.
- DF_t was the number of documents containing the term t.
- N was the total number of documents in the corpus.

The IDF was the reciprocal of the document frequency, measuring the informativeness of the t term. When we calculated the IDF, the low weight was given to the most frequent words such as stop words (because stop words such as "that" were present in most documents, and N / df would offer shallow scores for that word). Therefore, in the end, it would give us what we wanted – the relative weight.

A search for document similarity by word appearance cut into Unigram, bigram, or trigram had been carried out to search for requirements, resulting in low precision and recall values [14]. Therefore, in this study, the process of finding words in the Unigram and bigram or trigram formations was formulated with TF-IDF. The word vectorization results from TF-IDF would then be used as input for the three topic modeling algorithms, namely LSA, LDA, and NMF.

LSA studied latent topics by performing matrix decomposition on the term-document matrix using *singular value decomposition* (SVD) [5]. The python method that would be used is SVD. SVD broke the matrix into an orthogonal column matrix, an orthogonal row matrix, and a single matrix. There were several options for determining the optimal number of topics.

$$M = U\sum V^* \quad (2)$$

- M is the $m \times m$ matrix
- U is the left single *matrix* $m\ x\ n$
- Σ is *an* n × n diagonal matrix with non-negative real numbers.
- V is the right single matrix $m \times n$
- $V*$ is *an* $n \times m$ matrix, which is the transpose of V.

LDA extracted document features from the word level, go to the document level and finally reach the corpus level. Feature extraction with LDA was carried out through the knowledge stage of connectedness reasons, and the results could be implemented for extracting information. Connectedness was vital in the LDA process because it determined the document's distribution of topics. Therefore, the process of selecting a method of connection reasons needed special attention. Previous research on Indonesian language text processing compared the mean variational inference and Gibbs sampling methods showing that Gibbs Sampling performed better than the Mean Variational Inference for LDA [15].

$$p(z_{d,n}) = \frac{n_{d,k}+\alpha_k}{\sum_i^K n_{d,i}+\alpha_i} \frac{V_{k,w_{d,n}}+\lambda_{w_{d,n}}}{\sum_i V_{k,i}+\lambda_i} \quad (3)$$

- *n (d, k)* was the number of times document d used topic k
- v (k, w) was the frequency with which topic k used the given word
- *αk* was Dirichlet parameter for the document-to-topic distribution
- λw was the Dirichlet parameter for topic-to-word distribution.

NMF could be applied to multivariate data's statistical analysis by providing a set of n-dimensional multivariate data vectors. The vectors were placed in the n x m matrix V column where m was the number of samples in the data set. The matrix was then factored into *n x r* for *W* matrix and *r x m* for *H* matrix, where r was the number of generated topics [16]. Usually, r was chosen less than or m, so that *W* and *H* were less than *V* as the original matrix. This algorithm produced a compressed version of the original data matrix [17]. Several measures of reconstruction error between *V* and *WH* estimated:

$$\frac{1}{2}\|V - WH\|_F^2 = \sum_{i=1}^n \sum_{j=1}^m \left(A_{ij} - (WH)_{ij}\right)^2 \quad (4)$$

Maximization optimization to smooth *W* and *H* was used to minimize reconstruction errors. A common approach was to iterate between two renewal rules of multiplication until convergence.

$$H_{cj} \leftarrow H_{cj}\frac{(WV)_{cj}}{(WWH)_{cj}}\ W_{ic} \leftarrow \frac{(AH)_{ic}}{(WHH)_{ic}} \quad (5)$$

The requirements document sentences tend to be general and do not consist of technical terms and detailed processes. In contrast, the testing documents collected many technical terms and the detailed process flow of application operations. Therefore, both documents were grouped into different topics without any link when comparing the documents of requirements and testing. The link between the two documents would be very difficult to detect. Each document would be identified with its respective topics following the results of the pre-processing data. After the topics were identified in each document, the documents' linkages were determined based on the same topics they had through expert judgment. Every word in the sentence that belongs to the same topic (stemming results) would be entered into the search list to find the related source code. The overall modeling topic in the document and source code, as shown in Fig. 2.

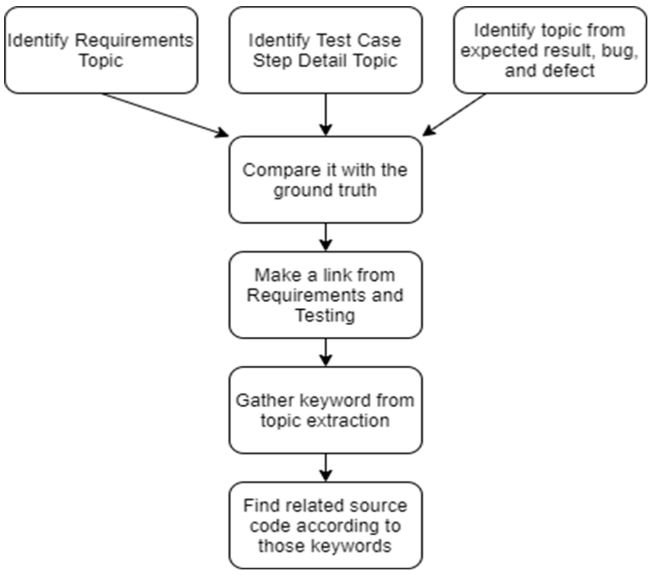
Identify Requirements Topic
Identify Test Case Step Detail Topic
Identify topic from expected result, bug, and defect
Compare it with the ground truth
Make a link from Requirements and Testing
Gather keyword from topic extraction
Find related source code according to those keywords

Fig. 2 Make traceability with topic modeling

C. Evaluation

The modeling topic would be carried out using three methods of LSA, LDA, and NMF. The topic modeling results would be compared with the existing ground truths. To determine how well a topic modeling method was carried out, it was necessary to measure accuracy, precision, and recall. When a system's learning model was run on the data set, a confusion matrix is shown in Table 5.

TABLE V
CONFUSION MATRIX

		Actual Value	
		True	False
Prediction Value	True	TP (True Positive) Correct positive class prediction results	FP (False Positive) Unexpected results
	False	FN (False Negative) Incompatible results	TN (True Negative) Correct negative class prediction results

Precision (also called positive predictive value) was the fraction of the relevant instances among the retrieved instances, as in formula (6) [18]. Recall (also known as sensitivity) was the fraction of the total number of relevant instances retrieved as in formula (7) [18]. In other words, precision was used to measure the accuracy of the desired information with the answers given by the model function. Then, recall measured the model's success in finding the desired information. Accuracy was the real proportion of both positive and negative in the overall data, measuring how close the correct predictive result was compared to the true value in formula (8) [18]. If the data used were not balanced in each class, then Cohen's Kappa was needed where E (Zk) was the expectation of Zk in the population of an item, and Eind (Zk) was an expectation that assumes statistical independence from ratings made by two observers [19], as shown in formula (9).

$$P = \frac{TP}{TP+FP} \tag{6}$$

$$R = \frac{TP}{TP+FN} \tag{7}$$

$$A = \frac{TP+TN}{TP+TN+FP+FN} \tag{8}$$

$$\kappa = 1 - \frac{E(Z_k)}{E_{ind}(Z_k)} \tag{9}$$

III. RESULT AND DISCUSSION

A. First App

We wanted to recognize the three topics in the first application: file management, promotion, and document validation status. Modeling topics using LDA often cannot separate the list of requirements and testing into these three topics. Several lists of requirements, which belonged to the status & validation module, often became clustered into the same topic with the promotion or file management module. This clustering happened because the file or document's status affects its process stages. When storing files, ordinary employees must first send them to the operator. The operator determined the employee's file validity and whether those files were suitable to keep and processed later for promotion needs. Therefore, the sentence for file management requirements also contained words about the file's status to indicate if the file could be processed further. Likewise, the promotion module's requirement sentence would have words about the file status and the validation process, limiting which files could be processed in the promotions stage. Meanwhile, NMF and LSA gave slightly better results in separating the three topics on requirements. NMF got the requirements separated well and successfully classified almost all requirements and testing for two topics: file validation status and promotion.

The topic of status and document validation became more inseparable at testing. This case happened because the steps taken were very similar to each other. The attributes involved in the file storage stage would appear when the operator validated the document and then changed the saved file's status. In the promotion module testing step, the file status essential attribute and the words indicate the document's status module process. These two modules overlap words and attributes raised because only files with a specific status could be used for processing as a promotion needs.

Meanwhile, the file storage process would cover all attributes and words about promotion, such as the choice of files for promotion by operators and how incomplete documents were uploaded by ordinary employees to meet the needs. Many attribute words were not specific to a particular topic in the testing step sentence. In this case, again, LDA failed to separate the three desired topics.

These fewer specific words made us look for the occurrence of one unique word and look for the occurrence of two or three uncommon words that coincided. Switching to Bigram and Trigram reduced the accuracy, and even LDA failed to separate the topic into what was expected. Again, NMF had the highest accuracy, although its value was smaller than before.

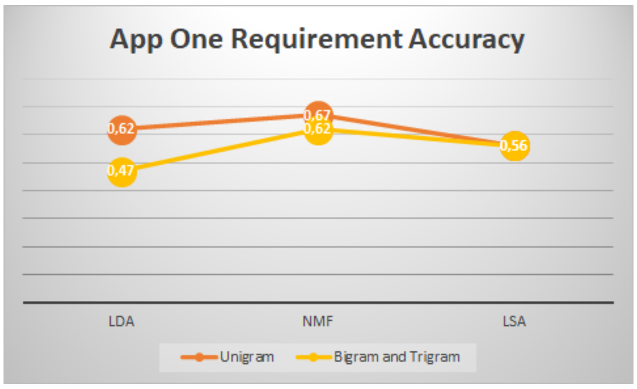

Fig. 3 App One Requirement Accuracy

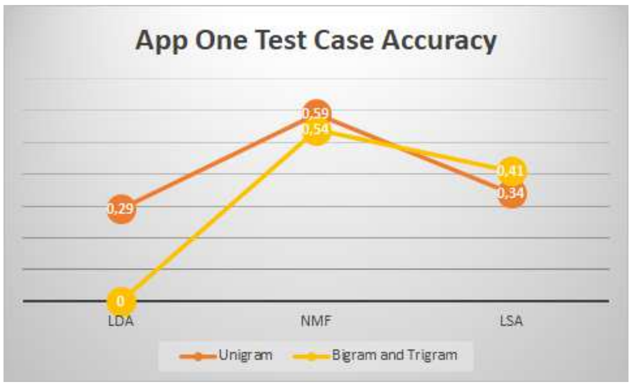

Fig. 4 App One Test Accuracy

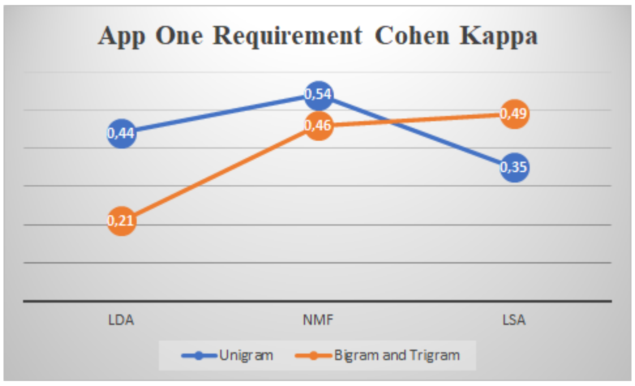

Fig. 5 App One Requirement Cohen Kappa

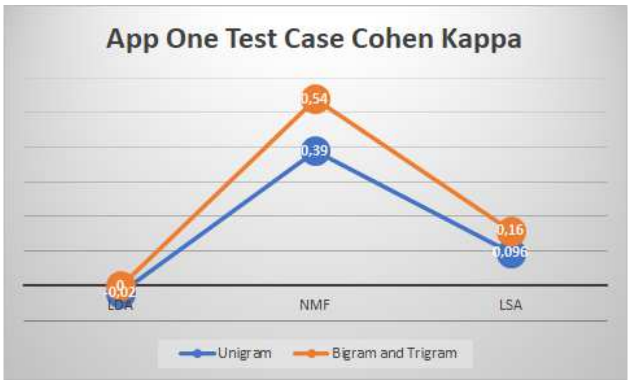

Fig. 6 App One Test Case Cohen Kappa

TABLE VI
APP ONE REQUIREMENT AND TEST CASE PRECISION AND RECALL

	Requirement		Test Case	
	Precision	Recall	Precision	Recall
LDAUnigram				
1	0.688	0.423	0.421	0.211
2	0.667	0.750	0.238	0.238
3	0.526	0.909	0.250	0.615
LDA Bigram or Trigram				
1	0.625	0.385		
2	0.474	0.562		
3	0.333	0.545		
NMF Unigram				
1	1.000	0.423	0.778	0.368
2	0.737	0.875	0.419	0.857
3	0.478	1.000	1.000	0.846
NMF Bigram dan Trigram				
1	1.000	0.385	0.765	0.342
2	0.632	0.750	0.517	0.714
3	0.458	1.000	0.423	0.846
LSA Unigram				
1	1.000	0.423	0.667	0.263
2	0.424	0.875	0.250	0.190
3	0.556	0.455	0.268	0.846
LSA Bigram or Trigram				
1	1.000	0.423	0.533	0.211
2	0.750	0.750	0.344	0.524
3	0.423	1.000	0.440	0.846

Overall, the Unigram word search accuracy value showed better results than bigram or trigram. Likewise, with the Precision and Recall values according to Fig. 3, Fig. 4, Fig. 5, and Fig. 6. Topic modeling in LDA testing was difficult to translate into promotion, document validation status, and file management. NMF had a reasonably stable accuracy value using Unigram and bigram or trigram. Meanwhile, LSA was in second place, and LDA was last. In the LDA and NMF methods, the precision value and recall score were also better when using the Unigram method. We discovered something different applied to the LSA method. Even though the accuracy value was high with Unigram, the precision and recall values were better when using bigram and trigram.

As shown in Fig. 3, the requirements with the NMF - Unigram method produced an accuracy value of 0.67, and when using NMF -Bigram and Trigram, it only reached 0.62. Fig. 4 showed similar results where the NMF once again had the best accuracy with 0.59 for the test steps using the Unigram method and only 0.54 when we used Bigram or Trigram.

Data on topic grouping in the requirements document and testing documents were not balanced for each module representation, and because the data was imbalanced, accuracy, precision, and recall values could be affected. For example, larger than the other two modules, the data on file management would be grouped differently and could be identified as overlapping topics or even the same as the second or third module. Therefore, the Cohen Kappa Coefficient needed to be calculated to see whether the three modules had an agreement on the pattern of application of each topic modeling algorithm. Cohen Kappa measured the accidental dependence of the predicted grouping measure and the actual result to remove intrinsic characteristics from the existing data [20]. In Fig. 5, requirements with NMF topic modeling had a high positive value, namely 0.54 for the Unigram search and 0.46 for Bigram and Trigram search. This result meant that the NMF algorithm model runs well in grouping the three modules compared to the other two algorithms LSA and LDA. In Fig. 6, it could be seen that LDA failed to perform the grouping of the test steps. The resulting Cohen Kappa value was also negative, which means that the model did not work, and it could be seen that NMF gave a good Cohen Kappa score once again compared to the other two algorithms; however, in the test case document, NMF did a better job using Bigram and Trigram searches than Unigram. NMF-Unigram only reached Cohen kappa value 0.39, while NMF - Bigram and Trigram could reach up to 0.54. These results differ in accuracy, precision, and recall, where Unigram gave better scores. So, it could be concluded that requirement topic modeling worked well using the NMF algorithm with the Unigram search. Still, the NMF algorithm worked better using Bigram and Trigram searches in the grouping of testing steps.

Based on Table 6, Unigram with the three algorithms produced the NMF precision value of 0.737 for topic two for the requirements and 0.778 for topic one for the test case. This value was higher than the bigram or trigram method, which only achieved a precision value of 0.632 for the requirements and 0.765 for the test case. In this first application, the NMF topic modeling method best separated requirements and test steps in a test case based on the desired topic.

Seeing that Unigram worked better when compared to bigram or trigram, we only used the unigram method to find bugs and defect links with testing expected results to find the defect relation. The three topic modeling methods were compared again to see their similarities. The comparison was from the list of desired results in the test case document, a list of bugs that appeared during testing (which could be during unit testing or integration testing), and a list of defects when the user tested the application's full function after the product was released (regression testing). Once again, the NMF method gave the best accuracy value results shown in Fig. 7. Table 7 showed that NMF provided almost the same precision value as LSA with 0,650 in topic 1 and achieved the highest recall values on topics 2 and 3 with 0.680 and 0.708.

The first topic was the issue of promotion and document storage. NMF succeeded in classifying defects with accompanying bugs and problematic testing steps in the file storage process, which failed to appear correctly on the promotion document list. In the second topic, filtering document types and document categories were no longer a problem because no defects had been found. On topic 3, a problem with the document status appearance was found. The validation process was problematic. The defect that was a solved bug before happened again. When a document status that had been rejected still appeared on the operator page and did not reappear on the employee page. The LSA method could not classify defects well because all were collected on one topic only, and the other two topics only presented testing without any connection with bugs and defects at all. LDA performed well in a grouping on topics regarding the defects of promotion and file management modules. For example, the file storage through scanning could not appear and failed to save, resulting in the system sending blank files for validation. In the end, the list of files for the promotion process did not appear correctly.

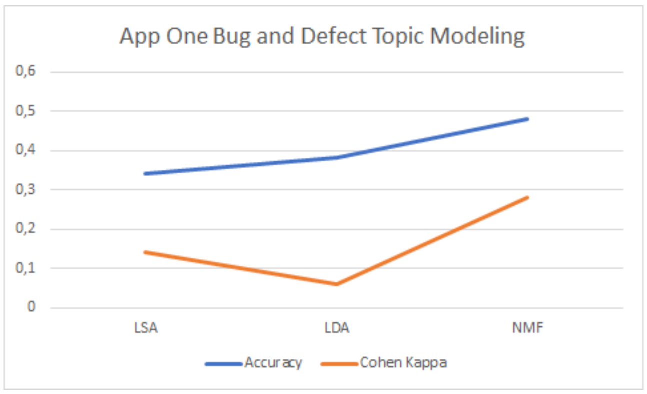

Fig. 7 App One Bug and Defect Topic Modeling

TABLE VII
APP ONE BUG AND DEFECT PRECISION AND RECALL

	LSA		LDA		NMF	
Topic	Precision	Recall	Precision	Recall	Precision	Recall
1	0.600	0.128	0.500	0.340	0.650	0.277
2	0.600	0.240	0.286	0.400	0.548	0.680
3	0.316	1.000	0.345	0.417	0.378	0.708

Based on modeling topics in requirements documents, test cases, and defect lists, it was found that each module had a dominant word as its identity. In this first application, the keywords that would be used as a search tool for the code source were as follows:

- The first module was a file management. The keywords were "*pindai*" (scan), "scan", "*simpan*" (save), "upload", "kategori dokumen" (document category), and "PDF".
- The second module was the validation status. The keywords were valid, reject, process, and operator.
- The third module was the promotion process. The keywords were "*syarat*" (terms), "*persyaratan*" (terms), and "*pangkat*" (promotion).

Apart from the mentioned words, other words were the module's identity, but those words were constantly changing. These terms never appear in the source code. Tracking the implementation of requirements in source code via keywords in each module only managed to map 40% of the related code in this first application.

In all three datasets, namely requirements data, test cases, and defect lists, NMF worked best. NMF could separate the three types of requirements and correctly predict the category's existing requirements with little data, such as promotion and status validation. In this first application, the process occurred through a single stream where the variables involved were the same, and only the function was different. That was why there were not many unique words that could be used to make up a topic in a module. Therefore, it was not easy to distinguish between one topic and another. In addition, the terms used in requirements were very general, while the terms in test cases were specific and hugely different. It could be said that the test case data had a lot of noise with the technical terms and the same variables that were mentioned repeatedly.

NMF performed better on short text with noise when compared to the other two algorithms [10]. This result was supported by previous findings where NMF performed slightly better on short text data with noise when compared to LDA, but the LDA used a bag of word matrix input instead of TF-IDF [21]. Meanwhile, the number of exact words in each module made it difficult for LSA to categorize topics. This was because the LSA did not care about word order and could not even distinguish a sentence from just a collection of words [22]. LDA tends to provide generalized and non-specific views and probabilistic functions that continually alter word results and occurrences [6], [10]. This made the grouping of topics unstable because sometimes there were lost topics when it was regenerated with different word distribution. The instability of the modeling results actually occurred in NMF. However, a study showed that NMF provided more stable results compared to LDA with the Normalized Pointwise Mutual Information (NPMI) value as its evaluation. [23].

B. Second App

In the second application, there were five modules. Therefore, the initial number of topics was also five. The five modules mentioned were master data, budget reference data management, activity reports, budget reports, and overall report view. There was almost the same grouping pattern when topic modeling was applied in requirement data, using LDA, NMF, and LSA. The separation of the five topics from the three algorithms can be reduced to two topics. When we used Unigram for word searching and weighting, the first topic was about funds or budgets, and the second topic was about activities and their attributes. In modeling topics with five topics in the LDA method, the first topic was about master data related to detailed actions such as measuring targets, deleting and adding activities, and partners in implementing activities. The second topic also revolved around activities such as adding master data in the field of technology. The activities were carried out in these work units, and the names of the actions were added. Activity reports such as obstacles, the location of activities carried out, and data to support activities were also included in the second topic. It could be seen that the first topic and the second topic discussed the attributes of the action.

Meanwhile, the budget report, the attributes of the source of funds, and the distribution of budget reports for each unit were included in the third topic. The fifth topic could not be concluded. The fifth topic was about the main reference of the budget and uploading it into the application. So, the third and fifth topics discussed funds or budgets. The topic modeling results using NMF separated the funding components into one topic on the second topic. The rest were attributes of activity implementation. LSA produced a new combination, where all matters related to master data were included in the third topic. Most of the budget attributes were included in the second topic, while a few others were included in the first topic with the activity reports' main attributes. The fourth topic was mixed, but it dominantly talked about activities. The accuracy score was quite high when the topic separation was reduced to two out of five topics due to the similarities of things discussed. NMF in first place reached 0.79, LDA and LSA in second place reached 0.75.

The method of searching and weighting words using bigram and trigram also still had the same pattern. In the LDA method, topic three was challenging to recognize because it was mixed. However, it tended to explain more about the budget because it contained words about the main budget reference. Topics one and two consisted of words that concern budget funds. Topics three and four were about activities and details of the implementation of activities. The NMF method produced each topic that was easier to identify than the LDA. Topics one and five were about activities, and the rest were about the budget. There were no topics that contained a mixture of the two. LSA still had better topic separation results than LDA, topics two, three, and four regarding the budget, and there was no need for activities on these three topics. The needs regarding activities were gathered on the first topic. Some of the fifth topic requirements were regarding the budget, but most of them explained the addition of data master and activity linkages. Bigram and trigram applications separated very well in the NMF method, which achieved an accuracy value of 0.81, and LSA in second place with 0.77. However, LDA accuracy decreased to only 0.61.

The problem arose when the activity and budget grouping patterns could not be applied to the testing document. Testing activity reports and testing budget reports were always grouped into the same topic for all topic modeling algorithms. The grouping pattern must be changed to avoid many topics grouping mistakes, especially in determining which group the activity report and budget combination belong to; thus, the grouping of requirements also changed to see the link between the existing patterns in testing. Everything about the master module would become one group. Everything was discussed in the activity report module, and the budget report would become another group. Everything about the budget reference and the report view would turn into one group. Three groups would be concluded from the results of the five topics resulting from the topic modeling process. The value of accuracy on the newest requirements topic pattern, as shown in Fig. 8 that NMF is no longer the best method. The LSA accuracy value rose to be the best with the unigram method, namely, 0.79. NMF was down in second place with 0.70, and LDA was in last place. However, NMF produced the best accuracy value on the bigram and trigram methods, 0.68 for requirements.

The emergence of the same unique word in the activity report test case and the financial report grouped these two modules on the same topic. As shown in Fig. 9, the LSA had better accuracy with new topic groupings. LSA hit 0.84, following NMF 0.79 in second place and LDA in the last place with 0.75. The high *kappa cohen* value on the LSA both in the requirements document and the test case document shown in Fig. 10 and Fig. 11 further strengthened that LSA was the winner this time. LSA achieved a Cohen Kappa coefficient value of 0.62 in the requirements document and 0.72 in the test case document with unigram word searches. However, in the requirements document, LSA was inferior to NMF when using bigram and trigram word searches. Where LSA only reached 0.45 while NMF reached 0.47.

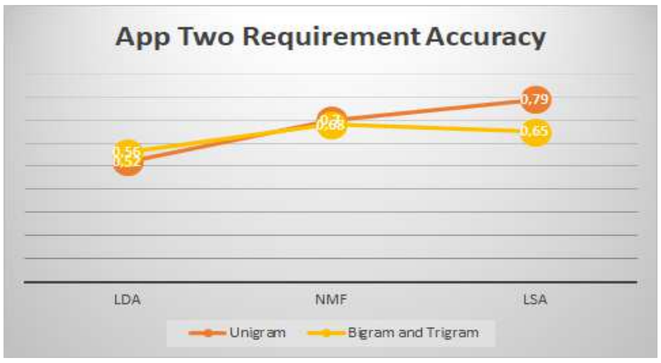

Fig. 8 App Two Requirement Accuracy

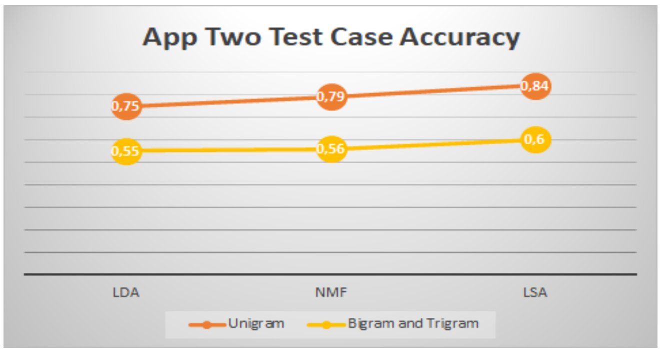

Fig. 9 App Two Test Case Accuracy

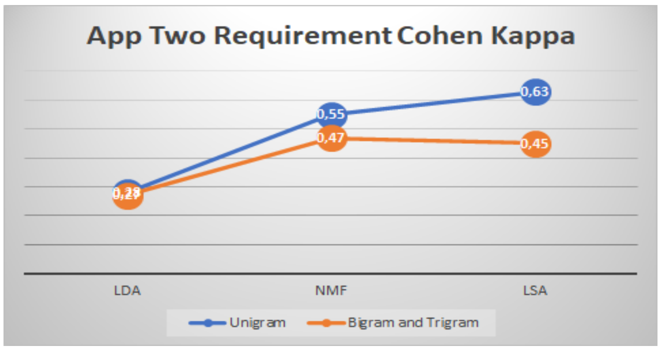

Fig. 10 App Two Requirement Cohen Kappa

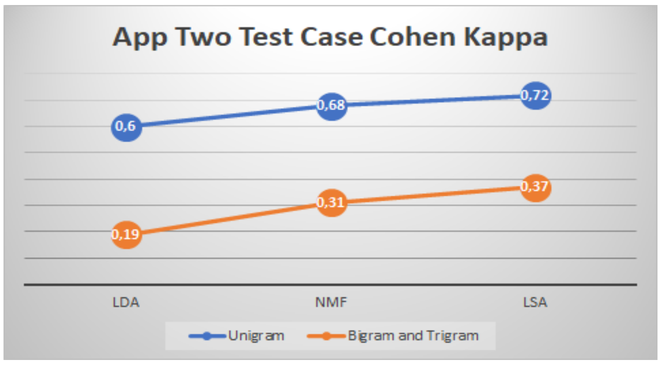

Fig. 11 App Two Test Case Cohen Kappa

TABLE VIII
APP TWO REQUIREMENT AND TEST CASE PRECISION AND RECALL

	Requirement		Test Case	
	Precision	Recall	Precision	Recall
LDA Unigram				
1	0.381	0.800	0.900	0.500
2	0.643	0.375	0.941	0.821
3	0.667	0.600	0.440	0.917
LDA Bigram or Trigram				
1	0.429	0.600	0.455	0.278
2	0.625	0.625	0.644	0.744
3	0.667	0.400	0.308	0.333
NMF Unigram				
1	0.533	0.800	0.621	1.000
2	0.875	0.583	1.000	0.692
3	0.692	0.900	0.769	0.833
NMF Bigram or Trigram				
1	0.455	0.500	0.600	0.333
2	0.870	0.833	0.821	0.590
3	0.500	0.500	0.290	0.750
LSA Unigram				
1	1.000	0.500	0.680	0.944
2	0.793	0.958	0.946	0.897
3	0.700	0.700	0.857	0.500
LSA Bigram dan Trigram				
1	0.500	0.400	0.556	0.556
2	0.857	0.750	0.781	0.641
3	0.467	0.700	0.368	0.583

As shown in Table 8, the precision value of using Unigram was again superior to using Bigram and Trigram. Modeling using the LSA algorithm gave perfect scores on the first grouping of topics and 0.793 on the requirements document's second topic. The LSA and NMF test documents provided almost as good results. NMF provided an excellent precision value on the second topic, while LSA only gave a precision result of 0.946. However, NMF could not provide a better precision value than LSA on the other two topics. LDA came last. On all topics, the precision value could not even touch 0.700.

We only used Unigram to model bugs and defect topics and did not compare it with Bigram or Trigram because the Unigram Requirements and Test Case topics modeling always showed better results. As shown in Fig. 12, LSA still excelled in finding suitable topics compared to the other two methods. The accuracy of LSA was at 0.56, in the second place was NMF, and in the last was LDA. Meanwhile, LSA showed that the model worked quite well, separating topics with the highest Cohen Kappa coefficient of 0.33 compared to the other two methods. Looking at the value of precision and recall on the three topics from Table 9, LSA gave good results in grouping the two topics. The LSA precision value was the highest on topic 2 with 0.963 compared to NMF 0.958 and LDA 0.800. The LSA recall value was also in the highest result on topic 3, 0.857 compared to NMF, only 0.607, and LDA 0.536. The LSA topic modeling method performed best in finding suitable topics in this second application.

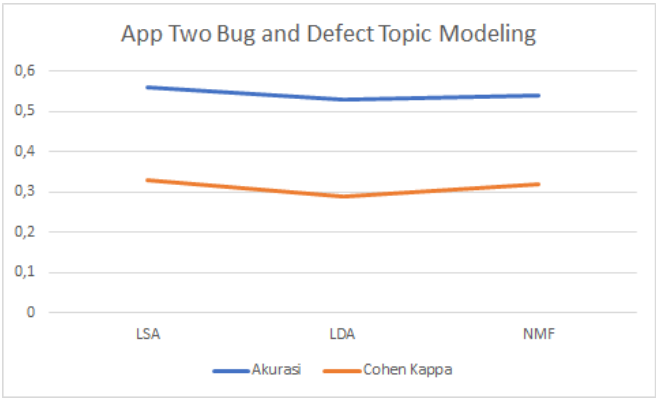

Fig. 12 App Two Bug and Defect Topic Modeling

TABLE IX
APP TWO BUG AND DEFECT PRECISION AND RECALL

	LSA		**LDA**		**NMF**	
Topic	Precision	Recall	Precision	Recall	Precision	Recall
1	0.364	0.174	0.400	0.522	0.400	0.522
2	0.963	0.565	0.800	0.522	0.958	0.500
3	0.407	0.857	0.405	0.536	0.395	0.607

Based on modeling topics in requirements documents, test cases, and defect lists, it was found that each module had a dominant word as its identity. In this second application, the keywords that are used as a search tool to the code source were as follows:

- The first module was the data master. The keywords were "master", "*kaitan kegiatan*", "*data awal*", "*data satuan*", "*nama kegiatan*"
- The second module was the activity and budget reports. The keywords were *pnbp, blu, status, target, dokumen pendukung*
- The third module was DIPA data management. The keywords were sub, output, component, DIPA, upload, import
- The fourth module was search and display reports based on DIPA activity data. The key words were *cari, pencarian, pilih, tampil, list, bidang teknologi, bidang prinas*

Apart from the words already mentioned, other words were the module's identity, but the terms were constantly changing. These terms never appeared in the source code. Tracking the implementation of requirements in source code via keywords in each module showed better results than the first application. The associated source code could be traced up to 62% in this second application.

Unlike the first application, the second application had unique words that are consistently mentioned starting from the requirements for each module so that the obstacles encountered in the study [5] were not found. The only changes were the things discussed by the requirements were in general. Those were separated into two categories, activities and funds. Then the test case explained those in more detail through the stages of the process in the application. This condition caused the number of topics about requirements to need to be changed so that the need could be mapped with existing test cases precisely.

Based on the consistency of the unique constituent words in each module, LSA could perform better than NMF and LDA. However, the LSA could not distinguish the meaning of words in a sentence. Therefore, adjustments such as the location of the subject and object needed to be identified as well as the types of words such as nouns and adjectives according to the previous study [24]. A word had a lot of meaning depending on its placement in a sentence, so this separation process was required for better LSA topic modeling. The LSA could be successful with data containing unique words that were consistent in use or by adjusting synonyms for words [25]. However, the LSA method with TF-IDF had the best result when used in short text from tweets, and the email contained health care issues rather than LSA combined with Doc2Vec and another method like LDA with TF-IDF or Doc2vec [26].

IV. CONCLUSION

This paper compares the tracing process of two types of developed applications in Agile. The tracing used topic modeling in three documents: the requirements document, test cases, and source code. In the first application, the topic modeling algorithm that works best is NMF, while in the second application, the topic modeling algorithm that works best is LSA. The first application does not consistently apply the terms used, and the variables involved are not unique to each module. Meanwhile, the second application has unique variables involved in each module so that each unique term is consistently mentioned from the requirements to the source code. Despite being in second place, NMF also performs quite well in the second application, where all evaluation values give results that are not too different from the LSA. In both applications, LDA always ranks last because the instability of the generated results is worse when inferred compared to NMF.

It can be concluded that tracing can be done more easily when consistent terms are occurring. It is better if the technical matters have been discussed clearly along with each development stage phase at the beginning of making the requirements. In the testing document, the terms need to reappear (as in the requirement document), not be replaced. Programmer comments on the source code are necessary to make it easier to find similarities apart from giving the variable names following the two previous documents (requirement and test case).

In this study, several weaknesses were found, including not paying attention to sentence structure in the existing data, not providing synonyms, and being dependent on expert judgment. These three weaknesses make the results of topic modeling to track the application of requirements imprecise. There are still many misidentifications. In the future, the data was processed by sentence structures and word synonyms so that the words that are checked for similarities have the same meaning. In addition, the agreement from the expert judgment that was compared later as ground truth should be measured not only based on *cohen kappa* but also using other measuring tools such as calculating the "Area Under the Curve" (AUC) of the "Receiver Characteristic Operator" (ROC) to emphasize how well segregated each group is.

ACKNOWLEDGMENT

This work is part of "Requirement Traceability Adoption Research in Agile Environments to Detect Defects" supported by Saintek Scholarship, Indonesian National Research and Innovation Agency, in collaboration with Institut Teknologi Sepuluh Nopember (ITS).

References

[1] N. N. Hidayati and S. Rochimah, "Requirements traceability for detecting defects in agile software development," *EECCIS 2020 - 2020 10th Electr. Power, Electron. Commun. Control. Informatics Semin.*, pp. 248–253, 2020, doi: 10.1109/EECCIS49483.2020.9263420.

[2] B. Wang, R. Peng, Y. Li, H. Lai, and Z. Wang, "Requirements traceability technologies and technology transfer decision support: A systematic review," *J. Syst. Softw.*, vol. 146, pp. 59–79, 2018, doi: 10.1016/j.jss.2018.09.001.

[3] T. Vale, E. S. de Almeida, V. Alves, U. Kulesza, N. Niu, and R. de Lima, "Software product lines traceability: A systematic mapping study," *Inf. Softw. Technol.*, vol. 84, pp. 1–18, 2017, doi: 10.1016/j.infsof.2016.12.004.

[4] C. Mills, J. Escobar-Avila, and S. Haiduc, "Automatic traceability maintenance via machine learning classification," *Proc. - 2018 IEEE Int. Conf. Softw. Maint. Evol. ICSME 2018*, pp. 369–380, 2018, doi: 10.1109/ICSME.2018.00045.

[5] D. Nanang, P. L. Penelusuran, and P. L. Penelusuran, "Pembangunan Link Penelusuran Kebutuhan Fungsional Dan Method Pada Kode Sumber Dengan Metode Pengambilan Informasi," *ELTEK*, vol. 16, pp. 151–165, 2018, [Online]. Available: https://doi.org/10.33795/eltek.v16i2.106.

[6] A. S. Ahmadiyah, R. Sarno, and F. Revindasari, "Adopted topic modeling for business process and software component conformity checking," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 18, no. 6, pp. 2939–2947, 2020, doi: 10.12928/TELKOMNIKA.v18i6.13381.

[7] S. Rani and M. Kumar, "Topic modeling and its applications in materials science and engineering," *Mater. Today Proc.*, vol. 45, pp. 5591–5596, 2021, doi: 10.1016/j.matpr.2021.02.313.

[8] J. Zhao, Q. P. Feng, P. Wu, J. L. Warner, J. C. Denny, and W. Q. Wei, "Using topic modeling via non-negative matrix factorization to identify relationships between genetic variants and disease phenotypes: A case study of Lipoprotein(a) (LPA)," *PLoS One*, vol. 14, no. 2, pp. 1–15, 2019, doi: 10.1371/journal.pone.0212112.

[9] R. Albalawi, T. H. Yeap, and M. Benyoucef, "Using Topic Modeling Methods for Short-Text Data: A Comparative Analysis," *Front. Artif. Intell.*, vol. 3, no. July, pp. 1–14, 2020, doi: 10.3389/frai.2020.00042.

[10] Y. Chen, H. Zhang, R. Liu, Z. Ye, and J. Lin, "Experimental explorations on short text topic mining between LDA and NMF based Schemes," *Knowledge-Based Syst.*, vol. 163, pp. 1–13, 2019, doi: 10.1016/j.knosys.2018.08.011.

[11] Q. Fu, Y. Zhuang, J. Gu, Y. Zhu, and X. Guo, "Agreeing to Disagree: Choosing Among Eight Topic-Modeling Methods," *Big Data Res.*, vol. 23, p. 100173, 2021, doi: 10.1016/j.bdr.2020.100173.

[12] H. Kaiya, A. Hazeyama, S. Ogata, T. Okubo, N. Yoshioka, and H. Washizaki, "Towards a knowledge base for software developers to choose suitable traceability techniques," *Procedia Comput. Sci.*, vol. 159, pp. 1075–1084, 2019, doi: 10.1016/j.procs.2019.09.276.

[13] A. Guo and T. Yang, "Research and improvement of feature words weight based on TFIDF algorithm," *Proc. 2016 IEEE Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2016*, pp. 415–419, 2016, doi: 10.1109/ITNEC.2016.7560393.

[14] H. Suhartoyo and S. Rochimah, "Membangun Hubungan Kerunutan Artifak Pada Lingkungan Pengembangan Cepat," *SYSTEMIC*, vol. 02, no. 01, pp. 1–17, 2016.

[15] P. M. Prihatini, I. Putra, I. Giriantari, and M. Sudarma, "Indonesian text feature extraction using gibbs sampling and mean variational inference latent dirichlet allocation," *QiR 2017 - 2017 15th Int. Conf. Qual. Res. Int. Symp. Electr. Comput. Eng.*, vol. 2017-Decem, pp. 40–44, 2017, doi: 10.1109/QIR.2017.8168448.

[16] P. Suri and N. R. Roy, "Comparison between LDA & NMF for event-detection from large text stream data," *3rd IEEE Int. Conf.* , pp. 1–5, 2017, doi: 10.1109/CIACT.2017.7977281.

[17] T. D. Hien, D. Van Tuan, P. Van At, and L. H. Son, "Novel algorithm for non-negative matrix factorization," *New Math. Nat. Comput.*, vol. 11, no. 2, pp. 121–133, 2015, doi: 10.1142/S1793005715400013.

[18] H. Dalianis and H. Dalianis, "Evaluation Metrics and Evaluation," *Clin. Text Min.*, no. 1967, pp. 45–53, 2018, doi: 10.1007/978-3-319-78503-5_6.

[19] S. Vanbelle, "Comparing dependent kappa coefficients obtained on multilevel data," *Biometrical J.*, vol. 59, no. 5, pp. 1016–1034, 2017, doi: 10.1002/bimj.201600093.

[20] E. Bagli and G. Visani, "Metrics for Multi-Class Classification : an Overview," *arXiv*, vol. abs/2008.0, pp. 1–17, 2020.

[21] S. A. Curiskis, B. Drake, T. R. Osborn, and P. J. Kennedy, "An evaluation of document clustering and topic modelling in two online social networks : Twitter and Reddit," *Inf. Process. Manag.*, vol. 57, no. 2, p. 102034, 2020, doi: 10.1016/j.ipm.2019.04.002.

[22] D. Braun and M. Langen, "Evaluating Natural Language Understanding Services for Conversational Question Answering Systems," *Proc. 18th Annu. {SIG}dial Meet. Discourse Dialogue*, no. August, pp. 174–185, 2017.

[23] M. Belford, B. Mac Namee, and D. Greene, "Stability of topic modeling via matrix factorization," *Expert Syst. Appl.*, vol. 91, pp. 159–169, 2018, doi: 10.1016/j.eswa.2017.08.047.

[24] R. M. Suleman and I. Korkontzelos, "Extending latent semantic analysis to manage its syntactic blindness," *Expert Syst. Appl.*, vol. 165, no. January 2020, p. 114130, 2021, doi: 10.1016/j.eswa.2020.114130.

[25] A Amalia et al, "Automated Bahasa Indonesia essay evaluation with latent semantic analysis Automated Bahasa Indonesia essay evaluation with latent semantic analysis," *J. Phys. Conf. Ser. 1235 012100*, pp. 0–8, 2019, doi: 10.1088/1742-6596/1235/1/012100.

[26] J. A. Lossio-Ventura, S. Gonzales, J. Morzan, H. Alatrista-Salas, T. Hernandez-Boussard, and J. Bian, "Evaluation of clustering and topic modeling methods over health-related tweets and emails," *Artif. Intell. Med.*, vol. 117, no. March, p. 102096, 2021, doi: 10.1016/j.artmed.2021.102096.