A Topic Modeling Based Solution for Confirming Software Documentation Quality

Nouh Alhindawi[1]
Faculty of Sciences and Information Technology, JADARA UNIVERSITY
Jordan

Obaida M. Al-Hazaimeh[2]
Department of Information Technology, AL-BALQA' APPLIED UNIVERSITY
Jordan

Rami Malkawi[3]
Faculty of Sciences and Information Technology, JADARA UNIVERSITY
Jordan

Jamal Alsakran[4]
King Abdullah II School for Information Technology, THE UNIVERSITY OF JORDAN
Jordan

***Abstract*—this paper presents an approach for evaluating and confirming the quality of the external software documentation using topic modeling. Typically, the quality of the external documentation has to mirror precisely the organization of the source code. Therefore, the elements of such documentation should be strongly written, associated, and presented. In this paper, we use Latent Dirichlet Allocation (LDA) and HELLINGER DISTANCE to compute the similarities between the fragments of source code and the external documentation topics. These similarities are used in this paper to improve and advance the existing external documentation. Furthermore, these similarities can also be used for evaluating the new documenting process during the evolution phase of the software. The results show that the new approach yields state-of-the-art performance in evaluating and confirming the existing external documentations quality and superiority.**

Keywords—Software Documentation; LDA; Clusters; HELLINGER DISTANCE; and Information Retrieval

I. Introduction

Modern software often consists of thousands of software development artifacts, such as external documents, design documents, code, bug reports, and test cases. These different kinds of documents are used by different kinds of people, such as developers, testers and also the end customers or clients. Therefore, writing these documents in a clear, easy, and understandable way is considered as an attribute for ideal software development and maintenance processes.

Typically, Software Documentation faults and oversights can increase the errors caused by software engineers. Moreover, it wastes developer's time and increases maintenance costs. For that reason, software engineers should pay much attention to documentation process. Moreover, Software Documentation quality is as significant as program quality. Any missing information about how to use the system, or how the system works, will cause the system to be degraded [1-3].

The external documentation describes each feature of the program, and assists the user in realizing these features, specially the new ones. Moreover, the external documentation can also go thus far as to supply thorough troubleshooting support. Generally, the external documentations are helpful in software engineering for development, maintenance, and evolution processes. Therefore, the external documentation should not be confusing, and they should be up to date. The assumption here is that external superiority documentation has to mirror precisely the organization of the source code. However, the external documentation and, where necessary, the system design and implementation, should be ideally modified and structured, so that changes can be easily documented and considered via external documentation correspondingly.

In this paper, a new methodology is presented that can be used to confirm the existing external documentation quality and superiority. The new approach for document assessment and confirmation consists of building models for source code and models for source code external documents using LDA. We compute the similarity between the documents distribution of the two models using Hellinger Distance.

Thus, we improve the techniques that were developed to deal with documentation quality assessment by integrating topic modeling with structural similarity measures to assess the quality of existing documentation.

In order to provide a base for our new external documentation confirming approach, we will now give more details about LDA modeling as well as a brief introduction to Hellinger Distance.

II. Extracting Topics with Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) [4] is a popular technique for getting probabilistic topic models from textual corpora by means of a generative process. LDA model is based on a fully generative model; for each document in the entire corpus, LDA represents it as a mixture of linguistic topics.

That is, LDA use the probability distribution over the gained topics to represent each document. In other words,

each document is modeled using LDA as multi-membership mixture of K-topics. Moreover, each topic is also represented as multi-Membership mixture of the corpus terms that exist in the vocabulary.

Using LDA, the corpus can be represented by a set of topics, and each document in the corpus can also be described by more than one of these topics. Moreover, each term from the corpus can be included in more than one of these topics. Therefore, any of corpus documents is not limited to being associated with a single topic, but as an alternative, it is modeled in a way that considers the possibility that document may address multiple topics.

Given *S* documents containing *k* topics stated over *u* unique words (w) the distribution of *i-th* topic to *i* over u words can be represented by φi and the distribution of *j-th* document ,document i (*doc* i) over k topics can be represented by *θj*.

The LDA assumes the following generative process for each document *doc i* in a corpus *D*:

- Choose N ~ Poisson distribution (ξ)
- Choose θ ~ Dirichlet distribution (α)
- For each of the N words w_i:
  - Choose a topic (k) to_k ~ Multinomial (θ).
  - Choose a word w_i from $P(w_i|z_n,\beta)$, a multinomial probability conditioned on topic to_k.

As conclusion, given a corpus of documents, LDA tries to discover the following:

- Recognizing a set of topics.
- Relates a set of words with a topic
- Specifies an exact mixture of these topics for each document in the corpus.

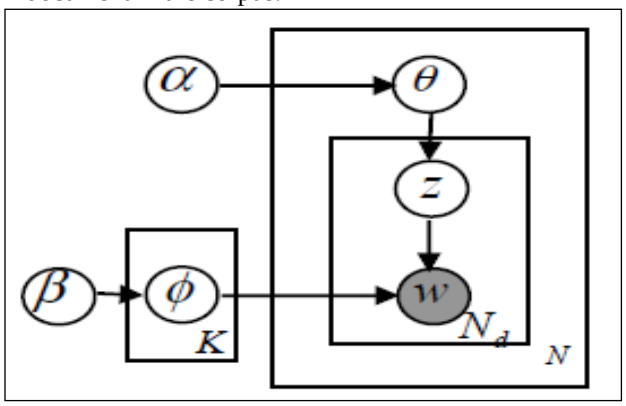

Fig. 1. LDA model. *K* is the number of topics; *N* is the number of documents; *Nd* is the number of word tokens in document *d*

For more details regarding LDA model, we refer the readers to Blei et al. work [4]. As mentioned before, LDA permits a document to have a combination of topics as we see in Figure 1. Moreover, the LDA model allows a document to exhibit multiple topics to different degrees, thus being more flexible than the cluster based techniques.

III. Hellinger Distance

Using HELLINGER DISTANCE with LDA modeling is our main contribution, as it achieves promising results. Using LDA proved its performance in locating and modeling any software artifacts, on the other hand, HELLINGER DISTANCE is also used in the literature as one of the standard methods that can compute the similarities between any dissimilar clusters probability distribution [5]. The main idea of our approach is to use the HELLINGER DISTANCE between document topics distributions to find the most likely similar and relevant topics from the two corpuses (SC and ED):

$$D(\emptyset 1, \emptyset 2) = \sqrt{\frac{1}{2}\sum_{t=1}^{T} (\sqrt{\emptyset 1, t} - \sqrt{\emptyset 2, t})^2}$$

IV. Documentation Assessment and Confirming with LDA and Hellinger Modeling

The proposed methodology is based on a set of parallel and sequential steps, which are partially automated:

STEP1. Extracting source code artifacts

STEP2. Extracting documents from external documentation.

STEP3. Building a corpus for source code artifacts

STEP4. Building a corpus for external documentation

STEP5. Extracting source code corpus topics (SC)

STEP6. Extracting external documentation corpus topics (ED)

STEP7. Computing the HELLINGER DISTANCE between the documents of SC and ED

STEP8. Analyzing the topics documents similarities

As shown in Figure 2, the process is done in pipeline architecture, in other words, the output from one phase constitutes the input for the next phase. The source code artifacts and the external documentation are used to create the corpuses that are used to generate the semantic space for Information Retrieval (IR) (see steps 1, 2, 3, and 4). The semantic topics produced from LDA for both corpuses are automatically generated in phases 5 and 6. More details about this step can be found in [2, 5, 6]. Once the topics of both corpuses are generated, the HELLINGER DISTANCE between the two corpuses documents is computed. As a final phase, we analyze the topics documents similarities, we use the similarities between source code topics and documents to infer missing associations or cross-references between existing sections of a documentation or suggest relations for the new documentation and source code.

In the following paragraphs, we present with details the corpuses building steps, the topics generating procedure, and the HELLINGER DISTANCE method.

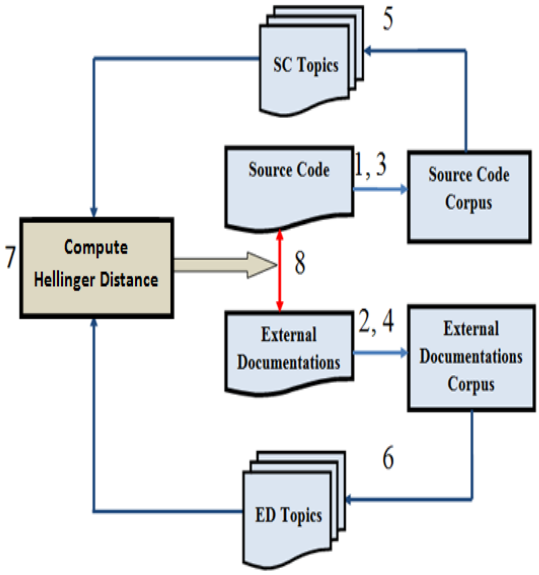

Fig. 2. Steps of Documentation Assessment with LDA and HELLINGER Modeling Approach

STEP 1: as an initial step for building source code corpus, we prepare the collections of artifacts which make up the corpus that LDA can process and infer. This is achieved by extracting all the textual information associated with a given source code; all the words used in comments or identifiers inside the method, class or package are extracted using our efficient corpus builder which was implemented in C++ to extract these important elements from source code that in XML format.

It takes less than 30 seconds to build both corpuses (corpora for of the two systems we used in the experiments). We use SrcML [7] tool to transform the C++ source code to XML format.

STEP 2: The same steps mentioned above for extracting source code artifacts are performed here to prepare the artifacts of the external documentation which make its corpus that LDA can infer. This is also done by extracting all the natural language information associated with a given source code; all the words used in include user documents (e.g., HTML,XML/docbook, LaTeX and Doxygen), build management documents (automake, cmake, and makefile), HowTo guides (e.g., FAQs), release and distribution documents (e.g.,ChangeLogs, whatsNew, README, and INSTALL guides), progress monitoring documents (TODO and STATUS), and extensible mechanisms (e.g., Python, Ruby, and Pearl bindings for an API) [2, 8, 9].

STEP 3: For both corpuses, we preprocess the words that can be found in both corpuses, starting by running them through a tokenizer. This allows us to split identifier names written with camel case or underscores (i.e., CamelCase or under_score) into their component words, giving us a better idea of what natural language topics and words are used in implementation.

STEP 4: In this step, we get rid of a set of reserved keywords and some other words that are very commonly used, such as “the” and “get”. Our approach allows the developer to specify easily any other stop words list.

STEP 5: The next step taken is stemming the words that make up our corpuses. Stemming includes removing the endings from words in order to recognize any corpus word despite what grammatical usage it appears in. We use porter default English stemmer [6].

STEP 6: After completing the previous steps, we are now able to generate and compute the topics with LDA. We use the LDA implementation provided by the Gensim library. Subsequently here, we choose the parameters to use in the computation, and then we extract topics from the documents. More details about this step are covered later in the following sub-sections.

STEP 7: In order to extract relevancy between the two corpuses linguistic topics, we use HELLINGER DISTANCE approach in two manners; in the first one, we compute the similarity between the topic i from ED topics and all the SC topics, while in the other one, we compute the similarities between all ED topics and all SC topics at the same query. Thus, we propose the following two methods for extracting and computing the two corpuses topics similarities: multi-topic and single-topic.

A. Single- Topic (LDA-S)

The LDA model is built based on all of the training documents of the source code. Given an ED test topic, we measure the HELLINGER DISTANCE between this topic distribution and the distributions of all SC topics. The SC topics with the lowest mean distance are returned as the most likely relevant SC topics to the taken ED topic. That is, the ED topic is queried over SC topics to retrieve the most similar topics.

B. Multi- Topic (LDA-M)

Here, the similarities between all SC and ED topics are measured, the result of this step is a ranked list that contains and shows any of SC and ED topics that have the maximum similarity percentage. Once the list is retrieved, the developer can distinguish and locate the related topics from both corpuses.

V. Experiments Setup and Datasets

In this section, we describe the experimental setup and datasets used in our experiments, followed by the evaluation of our new approach.

We conducted our experiments over KDE/KOFFICE open source system. We performed LDA topics modeling for both of KDE/KOFFICE source code system and over its external documentation. The evaluation of the new approach is done by comparing how many relevant topics from both corpuses were retrieved as relevant in the retrieved list, and the number of traceability links that exist between the two corpuses, which we found in our previous work [10].

Table I, shows the elements and the attributes for both of the two corpuses we built for KDE/KOFFICE system.

TABLE I. ARTIFACTS OF THE KDE/KOFFICE SYSTEM

KDE/KOffice	Count	Documents
Source Code Files	1057	11492
Non-Source Code Files	89	102
Total of External Documents		11594
Vocabulary	12839	_

The goal in [10], was to uncover traceability between source code and other artifacts using the TraceLab [11]. As mentioned before, this includes: user documents (e.g., HTML,XML/docbook, LaTeX and Doxygen), build management documents (automake, cmake, and makefile), How To guides (e.g., FAQs), release and distribution documents (e.g.,ChangeLogs, whatsNew, README, and INSTALL guides), progress monitoring documents (TODO and STATUS), and extensible mechanisms (e.g., Python, Ruby, and Pearl bindings for an API).

We performed the required preprocessing of the input texts. Both of the source code and the external documentation need to be broken up into the proper granularity to define the corpuses documents, which will be represented as vectors [2, 9, 12-14]. Therefore, we split up the source code into documents with function granularity level. As a result, each function has a corresponding document in the corpus of source code; this document contains the function name, local variable, global variable, function calls, and the internal comment of that function.

For external documentation, the paragraph is used as the granularity level. Table I contains the size of the system, as well as the dimensionality used for the LDA subspace and the determined vocabulary. For the LDA parameters, we can change the number of topics to be generated, as well as other LDA parameters, such as a number of iterations used and values of alpha and beta.

Typically, LDA model takes two parameters *Alpha* and *Beta,* where *Alpha* controls the division of documents into topics and *Beta* controls the division of topics into words. Larger values of *Beta* yields coarser topics, and larger values of *Alpha* yields coarser sharing of document into topics. For this reason the correct values of *Alpha* and *Beta* are required to obtain fine quality topics and to link topics to the original documents. A number of LDA implementations estimate these values on-the-fly while other implementations rely on the user to provide appropriate values [6, 15, 16].

We followed the recommendations in Gensim documentation, and set the Dirichlet hyper parameters to *Alpha*= min (0.1, 50/T) and *Beta* = 0.01, varying only the number of topics T. We ran the Gensim sampling process for *S* = 1000 iterations, and based the document representations on the last sample.

VI. EVALUATION AND DISCUSSION

The results are evaluated using categorization accuracy, i.e., the percentage of test documents topics that were correctly assigned to its corresponding source code topics. Moreover, we employ diverse accuracy series in the figures that reflect our results for precision of presentation.

The results show that using the LDA topic modeling along with the HELLINGER DISTANCE for confirming and linking the external documentation to its related source code fragments is working efficiently. As mentioned before, these outcomes have been proved using the already uncovered traceability links as shown in Table II.

In other words, for each of the extracted ED topics, we measured the HELLINGER DISTANCE between each of them and all of SC topics. We consider that a topic x from SD topics is related to set of topics from ED if the HELLINGER DISTANCE between them is the smaller. Thus, The SC topics with the lowest mean distance with respect to ED topic are returned as the most likely related topics. We called the related two topics as a pair. Next, we compared the pairs we have with the uncovered traceability links we found in our previous work [10].

TABLE II. DISCOVERED LINKS AND RECALL USING COSINE VALUE THRESHOLD

Cosine threshold	Total Links Recovered	Recall
0.60	184	84.2%
0.70	95	61.79%

In some cases, part of the documentation may refer to more than one source-code document, or a source-code document may be described by more than one external document. This fact has been proved in the results here, 103 ED topics based on the distance measure appear to be relevant to more than one SC topics, and this result confirms the efficiency of the proposed approach in spotting the relevancy between the source code fragments and between the significant external documentation.

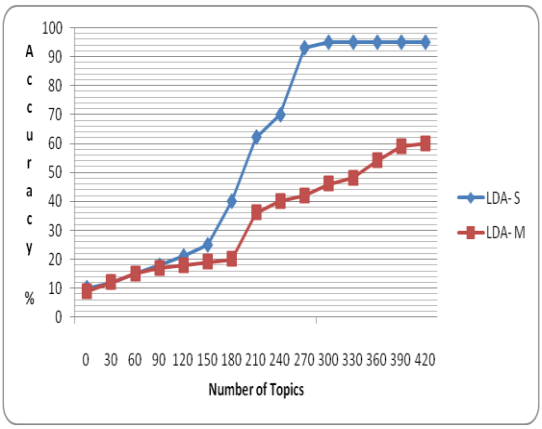

Fig. 3. Results of LDA-S and LDA-M

One notable result here, that 45 ED topics have poor relevancy with respect to all of SD topics. When investigating those topics, we found that most of them refer to authorship information and non-functional requirement information such as security recommendations. We argue here that labeling the external documentations that have such kind of information would be very efficient for developer's progression. For our experiments, we ran our LDA+Hellinger alternatives with 20, 40, 60 . . ., 300 and 400 topics. For LDA-M, the best accuracy we obtain is when the number of topics equal 400 as shown in Figure 3. However, LDA-S yielded a much higher accuracy than LDA-M.

Table III, shows the accuracy of investigated pairs matching compared with the recovered links. The second column in the table represents the number of pairs that were investigated, and the third column represents the percentage of accepted investigated pairs with respect to uncovered links. As we see in the table, LDA-S performs better accuracy that LDA- M with 226 investigated pairs. However, LDA-M only performs 0.30 as accuracy despite of the huge number of pairs that were retrieved within the specified threshold.

TABLE III. THE ACCURACY FOR BOTH TECHNIQUES (LDA-S AND LDA-M) WITH 0.25 AS THRESHOLD

Mechanisms	Number of Retrieved Pairs	Accuracy
LDA- S	226	0.80
LDA- M	391	0.30

When comparing the results of the two mechanisms (LDA-S and LDA-M), we note that LDA-S gives high precision even when only few topics are used, as we see in Table IV, The second column (Total links retrieved) represents the total number of recovered links (correct + incorrect), the third column (K value) represents number of topics that gives the best accuracy for each mechanism.

As we see the in the table, the difference between LDA-S and LDA-M is statistically significant. As we see, LDA- S discovered 181 traceability links, where LDA-M discovered 117 traceability links. The Table also shows the best K (number of topics) value where each mechanism gives the best accuracy.

TABLE IV. THE TOTAL NUMBERS OF LINKS WHICH DISCOVERED USING LDA-HELLINGER. K EQUALS THE NUMBER OF TOPICS THAT GIVES THE BEST ACCURACY FOR EACH MECHANISM

Mechanisms	Total Links Recovered via Matched Pairs	K- Value
LDA- S	181	300
LDA- M	117	420

An advantage of LDA-S over LDA-M is that LDA-S requires much less time to classify a test document when many SD per ED are available. However, this improvement in runtime may come at the punishment of accuracy and precision. The reason that LDA-M do better when more topics are considered may be that some important source code concepts are distributed to longer documents. That is, one concept/feature of source code fragments can be described by one or more external documentation. Furthermore, one source code concept/feature can usually be implemented by different parts of source code.

VII. RELATED WORKS

Several approaches have been developed in the past two decades to assist developers in obtaining an overview of the source code artifacts including the fragments of code, and the internal and the external documentation. However, the previous research in this area is limited. IR methods are considered as one of the most successful approaches in this field of research i.e., LSA and LDA [2, 8, 17].

There is a substantial amount of research which illustrates the relevance and the importance of documentation quality in the context of software evolution and development. Chen et al [18] presented the documentation quality problems as a major key problem in the domain of software engineering along with the main principles for writing the documentation for any software.

In [19], the author presented an automated quality assessment approach for software documentation using a developed document quality analysis framework and software document quality rules and principles.

Another framework for assessing documentation adequacy is also presented in [20], the authors mainly used a predefined taxonomic structure to assess a project documentation which funded by Naval Surface Warfare Systems (NSWC). Based on their findings of the authors, there is a need for a tool and method that can automatically evaluate any software documentation quality especially for large systems.

LDA was utilized for the first time to locate concepts in source code Linstead et al [21] by extracting the source code topics using LDA. Their approach can extract the concepts exist within the identifiers and the comments in the source code. Baldi et al [22] proposed a theory that software concerns are equivalent to the latent topics found by statistical topic models. They applied their approach to identify the global set of topics in many large systems.

In [16], LDA was utilized with the goal of enhancing and improving the process of analyzing the process of software evolution. Based on the results of the paper, the evolution process of software is more comprehendible when using the topics generated by LDA.

In [5], the authors use the HELLINGER DISTANCE between document topic distributions to find the most likely author of a specific document. Maskeri et al [23] considered the usage of the topics extracted with LDA from a software system.

Moreover, Classifying software systems into related groups in automatic way using LDA has been presented by Tian et al [24]. LDA was utilized to find traceability links between bug reports and program code by Lukins et al [25], their evaluation showed that LDA often drastically outperforms LSI.

In [8], Latent Semantic Indexing (LSI) was applied and utilized in order to find the similarities between fragments of code, the proposed approach aided the programmers when comprehending source code by clustering the similar and related fragments of source code. Moreover, LSI was also used in [13], the authors utilized and enhanced the usage of LSI to be used as a mapping technique for the concepts which expressed in natural language by relating them to their related fragments of code.

Topic Modeling was employed by the authors in [26], they used LSI to semantically cluster the artifacts which have similar or common vocabulary. The yielded clusters or groups are then linked based on the similarity between them along with visualization for these clusters. Moreover, labels are retrieved automatically for each cluster and for the linked ones. The visualization which provided by the authors can help greatly in program comprehension process.

A study on software documentation quality in practice was conducted and presented in [27]. The authors presented a survey which categorizes the current state of software documentation quality and employed analysis approaches for achieving software documentation quality checking process. Based on their findings, they confirm that the most significant quality characteristics for the documentation quality are precision, clearness, constancy, and readability.

VIII. Conclusion

In this paper, an approach to evaluate and confirm the existing external documentation quality and superiority is presented. The new approach uses Latent Dirichlet Allocation (LDA) along with HELLINGER DISTANCE to compute the similarities among the source code artifacts and its external documentation. A set of experiments was presented and the results validated by comparing them with uncovered links extracted in previous work over KDE/Koffice system.

The results show clearly that the new approach proved its efficiency in classifying and confirming the quality of source code external documentation. Moreover, based on the results, we argue here that labeling and grouping the external documentation would impact positively on the quality of the documentation. Based on the results we found, the needs for tools that can assess the software documentation quality in an automatic way are highly demanded.

Acknowledgement

The authors would like to thank faculty of sciences and information Technology, JADARA UNIVERSITY for support funding to carry out this research project.

References

[1] IEEE Standard for Software User Documentation. IEEE Std 1063-2001, 2001: p. 1-24.

[2] Marcus, A. and J.I. Maletic. Recovering documentation-to-source-code traceability links using latent semantic indexing. in Software Engineering, 2003. Proceedings. 25th International Conference on. 2003.

[3] Marcus, A. and J.I. Maletic, Recovering documentation-to-source-code traceability links using latent semantic indexing, in 25th International Conference on Software Engineering2003, IEEE Computer Society: Portland, Oregon. p. 125-135.

[4] Blei, D.M., A.Y. Ng, and M.I. Jordan, Latent dirichlet allocation. J. Mach. Learn. Res., 2003. 3: p. 993-1022.

[5] Seroussi, Y., I. Zukerman, and F. Bohnert, Authorship attribution with latent Dirichlet allocation, in Proceedings of the Fifteenth Conference on Computational Natural Language Learning2011, Association for Computational Linguistics: Portland, Oregon. p. 181-189.

[6] Savage, T., et al. TopicXP: Exploring topics in source code using Latent Dirichlet Allocation. in IEEE International Conference on Software Maintenance (ICSM). 2010. IEEE Computer Society.

[7] Collard, M.L., M.J. Decker, and J.I. Maletic. Lightweight Transformation and Fact Extraction with the srcML Toolkit. in IEEE 11th International Working Conference on Source Code Analysis and Manipulation. 2011.

[8] Maletic, J.I. and A. Marcus. Using latent semantic analysis to identify similarities in source code to support program understanding. in 12th IEEE International Conference on Tools with Artificial Intelligence (ICTAI) 2000.

[9] Alhindawi, N., et al., A TraceLab-Based Solution for Identifying Traceability Links using LSI, in 7th ACM International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE)2013: California, USA. p. 79-82.

[10] Alhindawi, N., et al. LSI-Based Solution for Categorizing Software Repositories Commits for Maintenance in Working Conference on Reverse Engineering (WCRE). To Be Submmitted. 2013.

[11] Keenan, E., et al. TraceLab: An experimental workbench for equipping researchers to innovate, synthesize, and comparatively evaluate traceability solutions. in 34th International Conference on Software Engineering (ICSE) 2012.

[12] Alhindawi, N., et al. Improving Feature Location by Enhancing Source Code with Stereotypes. in International Conference on Software Maintenance (ICSM) Submitted. 2013.

[13] Marcus, A., et al. An Information Retrieval Approach to Concept Location in Source Code. in 11th Working Conference on Reverse Engineering. 2004. IEEE Computer Society.

[14] Poshyvanyk, D. and A. Marcus. Combining Formal Concept Analysis with Information Retrieval for Concept Location in Source Code. in 15th IEEE International Conference on Program Comprehension (ICPC). 2007.

[15] Tian, K., M. Revelle, and D. Poshyvanyk. Using Latent Dirichlet Allocation for automatic categorization of software. in 6th IEEE International Working Conference on Mining Software Repositories (MSR). 2009. IEEE Computer Society.

[16] Linstead, E., C. Lopes, and P. Baldi. An Application of Latent Dirichlet Allocation to Analyzing Software Evolution. in Seventh International Conference on Machine Learning and Applications. 2008. IEEE Computer Society.

[17] Binkley, D. and D. Lawrie, Information Retrieval Applications in Software Maintenance and Evolution, in Encyclopedia of Software Engineering, P. Laplante, Ed.2010: Taylor & Francis LLC.

[18] Chen, J.-C. and S.-J. Huang, An empirical analysis of the impact of software development problem factors on software maintainability. J. Syst. Softw., 2009. 82(6): p. 981-992.

[19] Dautovic, A., Automatic assessment of software documentation quality, in Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering2011, IEEE Computer Society. p. 665-669.

[20] Arthur, J.D. and K.T. Stevens, Document quality indicators: A framework for assessing documentation adequacy. Journal of Software Maintenance: Research and Practice, 1992. 4(3): p. 129-142.

[21] Linstead, E., et al. Mining Eclipse Developer Contributions via Author-Topic Models. in Mining Software Repositories, 2007. ICSE Workshops MSR '07. Fourth International Workshop on. 2007.

[22] Baldi, P.F., et al., A theory of aspects as latent topics, in Proceedings of the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications2008, ACM: Nashville, TN, USA. p. 543-562.

[23] Maskeri, G., S. Sarkar, and K. Heafield, Mining business topics in source code using latent dirichlet allocation, in Proceedings of the 1st India software engineering conference2008, ACM: Hyderabad, India. p. 113-120.

[24] Kai, T., M. Revelle, and D. Poshyvanyk. Using Latent Dirichlet Allocation for automatic categorization of software. in Mining Software Repositories, 2009. MSR '09. 6th IEEE International Working Conference on. 2009.

[25] Lukins, S.K., N.A. Kraft, and L.H. Etzkorn. Source Code Retrieval for Bug Localization Using Latent Dirichlet Allocation. in Reverse Engineering, 2008. WCRE '08. 15th Working Conference on. 2008.

[26] Rousidis, D. and C. Tjortjis. Clustering Data Retrieved from Java Source Code to Support Software Maintenance: A Case Study. in Software Maintenance and Reengineering, 2005. CSMR 2005. Ninth European Conference on. 2005.

[27] Plosch, R., A. Dautovic, and M. Saft. The Value of Software Documentation Quality. in Quality Software (QSIC), 2014 14th International Conference on. 2014.