Performance Evaluation of Information Retrieval Models in Bug Localization

Mai Alduailij (mai.a.alduailij@wmich.edu)
Mona Alduailij (mona.a.alduailij@wmich.edu)
Departments of Computer Science and Statistics
Western Michigan University Kalamazoo, MI, USA
4433 Wimbledon Way, Kalamazoo, MI 49009

Abstract—A great deal of research work has been dedicated to empirically compare and analyze the performance of text models in information retrieval based bug localization. However, obtaining the required empirical data is typically computationally expensive. Our study uses the P-value test to compare the performance of three text models that have been extensively used in information retrieval based bug localization: Vector Space Model (VSM), Latent Semantic Indexing (LSI), and Latent Dirichlet Analysis (LDA) on the method level. Using statistical significance, our study proves that VSM outperforms LSI and LDA in retrieval of relevant methods to bug reports on the method level. We then examine the factors affecting the behavior of VSM in method retrieval. The factors analyzed include, but are not limited to: methods’ lengths, queries’ lengths, methods’ documentation comments, products’ names and components’ names mentioned in bug reports. The analysis shows a strong correlation between the behavior of VSM and most of the factors examined. Our results can be generalized over any dataset that exhibit similar characteristics to those of our sample.

Index Terms—information retrieval, bug localization, hypothesis testing, correlation analysis, performance evaluation.

I. INTRODUCTION

The two main input elements that need to be investigated in the process of information retrieval based bug localization are bug report, which is extracted from a bugging system, and the source files, which would be located in a revision control system.

Despite the existence of different bugging systems (Bugzilla, BugTracker.Net, FogBugz, etc.), the structure of bug reporting remains invariant. Most bugging systems guide reporters through a series of questions that help identify and limit the ambiguity of the bug.

Source code in object oriented programming is composed of different granularities. A program can include several classes in which several methods, attributes and inner classes are embedded. In one way, encapsulation of code would help software testers to locate all affected areas of the code; once a bug is found in a unit of code, all items included under that unit must be modified accordingly. On the other hand, investigating a large polymorphic class for errors can be a horrendous task, especially if the class includes a massive number of attributes, methods, and inner classes. In order to avoid such difficulties, some researchers have performed information retrieval based bug localization on the method-level [23]. Method-level bug localization can direct testers to the flawed method. Therefore, testers have to search a few lines in the method instead of scrutinizing the whole class file. Moreover, if the relevant class is a large file with only a few flawed methods, there will be a lot of terms that are not related to the query, which will affect the similarity measure.

Text models performance in information retrieval based bug localization has been the subject of comparison and analysis in recent empirical research. Some research work have been dedicated to compare different text models [1, 11, 20, 26]. Another research area has been designated to improve classical text models by developing different weighting tools [11]. Zhou, Zhang, and Lo propose a new bug localization tool by improving the weighting technique used in classical VSM [11]. Some other research has combined several ranking schemes [18], or combines text models and clustering algorithms, to enhance the performance of text models [23]. However, previous comparisons of information retrieval models lack the use of statistical tests to prove the significance of their results and to correctly predict the performance of these models. On the other hand, only a few research attempts have been done on the impact of the length of documents and queries on the performance of text models. Cummins and O’Riordan study the effect of query length on normalization in information retrieval [7]. Karbasi and Boughanem suggest a document length normalization approach that uses effective level of term frequency on large collections containing documents of varying lengths [12].

Researchers have employed empirical experiments to evaluate the performance of information retrieval techniques, but to collect ground truth data including the queries and their associated relevant methods, and then preprocessing this massive dataset, intensely require time, computer and man power alike.

Hypothesis testing is a statistical inference tool that can be used to validate a certain claim. Unlike empirical experiments, proving the claim does not require the collection of massive amount of data from different projects. The hypothesis is tested on a representative sample [3, 22], and then the results can be extrapolated to any dataset that matches the characteristics of the population where the sample is drawn from [2, 5, 19].

The contributions of this study are:

- First, we use the hypothesis test of means to compare the performance of three widely used text models in bug localization: VSM, LSI and LDA on the method level. We use Eclipse v.3.1 as the source file system.
sampling is used to pick a sample of 280 out of 8188 bug reports from Bugzilla. Our results show that VSM outperforms LSI and LDA in retrieval of relevant methods for bug localization.

- Second, VSM produced results in our preliminary comparison in retrieving relevant methods which are better than the results produced by LSI and LDA. Thus, we decided to run further tests on VSM to investigate what factors affect its behavior. We use the hypothesis test of proportion and correlation analysis to examine the potential effect of several factors over the performance of VSM. For each tested bug report, the factors include the query’s length, the relevant methods’ length, the methods’ documentation comments, the availability of a product or a component name in the query, and the relationship between these factors. Our results show that most of the tested factors have a strong influence on the performance of VSM on the method level.

We believe our work can help developers understand the strengths and limitations of VSM for future development. It will also help project programmers using classical VSM for bug localization to understand improved ways to approach and prepare methods and queries to deliver better results. Our findings can be extended to include other datasets given that each bug contains: a title, a description, and a summary for any object oriented written project; with the caveat that the source files are decomposed into the containing methods.

The rest of the paper is organized as follows: we discuss the related work in Section II. In Section III, we introduce the research method. Research questions are described in Section IV. In Section V, we show and discuss the experimental results. Limitations and threats to validity are explained in Section VI. Conclusion and future work are discussed in Section VII.

II. RELATED WORK

Text models have been a subject of comparison in different research work [1, 11, 20, 26]. Rao and Kak compare the performance of several generic and composite text models in bug localization using empirical experiments [20]. The performance of BugLocator tool proposed by Zhou, Zhang, and Lo was compared empirically to SUM, LSI, LDA, and VSM to prove its effectiveness in information retrieval [11]. A few research attempts have been done to propose new ways to normalize documents and queries of varying lengths. Cummins and O’Riordan study the effect of query length on normalization in information retrieval [7]. Karlbäck and Bougahem suggest a document length normalization approach that uses effective level of term frequency on large collections containing documents of varying lengths [12]. In their work, Scanniello and Marcus use method-level granularity with the lead comments on each method [23]. While Hiew assigns the same importance to all sections of the bug report [10], Runeson, Alexandersson, Nyholm examine the effect of treating the summary as twice as important as the description in detecting duplicate reports [21]. Zhou, Leung, and Xu study the effect of class size in the association between object-oriented metrics and change-proneness using the confounding effect [29]. Although information retrieval based bug localization has been studied by several researchers, none of the work has been proved statistically. In addition, only a few research attempts study the impact of external factors such as query length and document length on the performance of text models in the task of bug localization.

III. RESEARCH METHOD

In this section we explain the text models to be evaluated, the data collection process, the evaluation metrics and the statistical tools to be used:

A. Information Retrieval (IR) Models

Many deterministic and probabilistic information retrieval techniques have been examined in previous scholarly works [6, 11, 15, 17]. In our study, we will use the following models:

1) VSM: Vector Space Model (VSM) is a deterministic IR model in which each document is represented as a vector where each element of the vector corresponds to a term in the document representation. A vector is constructed to represent terms of the query and a similarity measure is chosen to compute the closeness of each document vector to the query vector.

2) LSI: Latent Semantic Indexing (LSI) [8] is a deterministic indexing and IR technique that can determine the relationship between concepts and terms contained in unstructured text [9]. All documents are represented as a single term-document matrix. To compute the similarity between a query and a set of documents, the query is treated as a document and Singular Value Decomposition (SVD) is employed to find the similarity.

3) LDA: Latent Dirichlet Allocation (LDA) [4] is a probabilistic generation and Bayesian model. It is based on the idea that each set of terms in a document must follow a certain topic. Each topic follows a Dirichlet distribution over terms. LDA divides the document set into several latent topics, where each document can be represented as a mixture of topics [30].

B. Data Collection:

1) Source Files: The subject system used for this study was Eclipse v.3.1. We chose Eclipse system for its open source large scale source files, broadly known reputation, stability, and its user friendly bugs system. The following steps were performed on the system’s source files:

   a) Decomposition of the source files: 13639 source files of Eclipse v.3.1 were decomposed into 161069 documents containing methods of varying lengths.

   b) Preprocessing of the resulting methods: The resulting methods underwent the common text normalization steps of letter transformation, letter separation, stopword removal, and word stemming.

2) Bugs: The bugs used for the evaluation were extracted from the Bugzilla on-line system. Several steps were taken to obtain the bugs used in the study:

   a) The criteria for the chosen bugs were: Eclipse v.3.1 and resolved/fixed bugs. The result was 8188 bugs.
b) In order to get the link between the bugs and the fixed methods: we only chose the bugs that included Diff. information. The result was 1665 bugs.

c) We scanned the Diff. information of the 1665 bugs to get the fixed methods: If the fix was to add a new method i.e. the method does not exist in the Eclipse v.3.1., the bug was discarded, otherwise, the bug was kept and the fixed methods names were saved for later evaluation. The number of bugs after this step was 931. Two issues had to be considered when saving the methods’ names: 1) similar methods’ names over several classes and 2) similar methods’ names within the same class (method overloading). To avoid such confusion, each fixed method was saved by its signature and concatenated to the full class name (i.e. package name and class name) where the method existed. The name of the resulting fixed method used for the evaluation process was constructed as:

Fixed method name = package name + class name + method signature

d) Each bug report of the 931 bugs included : the title, summary, and description.

e) The text normalization process : which includes letter transformation, letter separation, stopword removal, and word stemming was performed on the 931 bugs.

C. Evaluation Metrics:

The evaluation metrics used in this study are the following:

1) MAP: Mean Average Precision (MAP) is a metric used to evaluate the quality of IR in bug localization. MAP is appropriate in cases where a query may have multiple relevant documents [13]. It computes precision at each position when a new relevant document is retrieved [25, 27]. Average Precision (AvgP) of each query is then calculated and the average over all queries is computed as follows:

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{M_j} \frac{1}{|Q_j|} \text{precision}(doc_i)$$  \hspace{1cm} (1)

where Q is set of queries, M_j is the number of relevant documents for query j, and precision(doc_i) is the precision at the i^{th} relevant document. The higher the MAP value, the better the retrieval performance.

2) MRR: Mean Reciprocal Rank (MRR) is a metric that computes Reciprocal Rank (RR) of the first relevant document appearing in a list of retrieved documents [14, 28]. MRR is calculated over all queries as follows:

$$MRR(Q) = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}$$  \hspace{1cm} (2)

where Q is set of queries, and rank_i is the rank of the first relevant document in the list of retrieved documents. The higher the MRR value the better the retrieval performance.

D. Population Validity

To ensure population validity, we select the target population by identifying the variables of interest that must be found in a population for our results to apply. We then select the accessible population and employ multistage random sampling to select the sample of study.

1) Target Population: Several variables of interest must be found in a population for the study results to apply:

a) The population size: must be greater than 30 points, in order to follow the normal distribution [5,22].

b) The bugs : each bug must contain a title, a description and a summary.

c) The language: The project should be built using an object oriented language

d) The source file: Each source file (class) must be decomposed into the methods it contains. Method's documentation comments i.e. comments preceding each decomposed method (if any) must be kept.

2) Accessible Population: It is critical to form a well representative sampling frame to ensure population validity. To achieve that, we followed a multistage sampling method to reach our population of interest of 931 bugs. We use two populations for the two main tasks of our study. For the task of comparing VSM, LSI and LDA, the population is the same 931 bugs. Three more steps are taken to select the population for the task of examining the factors affecting VSM:

a) Query length: We calculated the average query length in the 931 bugs.

b) The length of the first relevant method: For each bug report we calculated the length of the first relevant method retrieved by VSM. Then, the average method length across all the 931 bugs was computed.

c) From the 931 bugs: we selected the bugs fixed by VSM. The bug was considered fixed if at least one of its relevant methods was ranked less than or equal to 10. The result of this step was 121 points, where each point contained the bug and its highest ranked relevant method.

3) Sample size: After forming our sampling frame which consists of 931 elements for the first population, and 121 elements for the second population, we determined the required sample size using the following equation:

$$n = \frac{N}{1 + N(e^2)}$$  \hspace{1cm} (3)

where N is total population, e is the margin of error that our estimation will fall within ±e [24]. We have set our margin of error to be 0.05. Solving for equation (3), the needed sample for the comparison task is 280 bugs. For the factors examination task, the sample needed is 55 bugs.

4) Randomness of the Sample: After determining the sample size of each of the two populations, a script was run to assign a random number to each of the population points. This method of randomly selecting records from our population reduces selection bias, because each element in the population has the same probability of being selected [24]. In addition, due to the huge number of bug reporting systems, which requires enormous effort to collect the bugs, a well defined and complete listing of elements like ours allows us to extrapolate to other populations that exhibit similar characteristics.

E. Hypothesis

In this study we use the P-value approach to hypothesis testing. First, the hypothesis test of means is performed to
statistically prove the superiority of VSM over LSA and LDA in information retrieval for bug localization on the method level over any dataset. Second, the hypothesis test of proportions and correlation analysis are used to evaluate the impact of external factors on the behavior of VSM. External factors include, but are not limited to, the query’s length, the relevant methods’ length, and the availability of a product’s name or a component’s name in the query. We followed the steps taken to test the null hypothesis as explained in [2, 3, 5, 19, 22]

IV. RESEARCH QUESTIONS

RQ1: Can we prove that VSM outperforms LSI and LDA on the method level over any dataset that follows the variables of interest described in Section III.D?

To answer this question, we start by collecting the evaluation data. For each bug we collect the relevant methods that have been modified to fix that bug. From the population of 931 bugs we calculated the required sample size to be 280, which was randomly selected. We then run VSM, LSI and LDA on the 280 bugs to find out which methods of Eclipse v.3.1. fix each bug in the sample. MAP and MRR evaluation metrics are then applied on the results to measure the performance of each of the experimented IR models. After that, we use the $P$–value hypothesis test of means to verify the superiority of VSM over LSI and LDA:

- $H_1$ (MAP of VSM vs LSI): The null hypothesis $H_{1\text{-null}}$ states that $MAP_{VSM} \leq MAP_{LSI}$ While the alternative hypothesis $H_{1\text{-alt}}$ states that $MAP_{VSM} > MAP_{LSI}$.

- $H_2$ (MRR of VSM vs LSI): The null hypothesis $H_{2\text{-null}}$ states that $MRR_{VSM} \leq MRR_{LSI}$ While the alternative hypothesis $H_{2\text{-alt}}$ states that $MRR_{VSM} > MRR_{LSI}$.

- $H_3$ (MAP of VSM vs LDA): The null hypothesis $H_{3\text{-null}}$ states that $MAP_{VSM} \leq MAP_{LDA}$ While the alternative hypothesis $H_{3\text{-alt}}$ states that $MAP_{VSM} > MAP_{LDA}$.

- $H_4$ (MRR of VSM vs LDA): The null hypothesis $H_{4\text{-null}}$ states that $MRR_{VSM} \leq MRR_{LDA}$ While the alternative hypothesis $H_{4\text{-alt}}$ states that $MRR_{VSM} > MRR_{LDA}$.

We have to note that MAP is equivalent to $\mu_{AP}$ and MRR is equivalent to $\mu_{RR}$.

RQ2: Does the method length have any effect on the performance of VSM?

To answer RQ2, the following hypothesis is defined:

- $H_5$: The null hypothesis $H_{5\text{-null}}$ states that $\pi_{SM} \leq 0.5$. While the alternative hypothesis $H_{5\text{-alt}}$ states that $\pi_{SM} > 0.5$, where $\pi_{SM}$ is the proportion of Short Methods. Hypothesis test of proportions is used to answer the above hypothesis. To accomplish this task, we calculate the average method length of the 121 methods in the population, which equals 46, where the method length represents the number of words in the method after the text normalization process. Based on that average, each method shorter than the average is considered a short method; on the contrary, each method longer than or equal to the average is considered long. After calculating the average method length in the population, we compute the sample size to be 55, which was randomly selected using a random number generator script.

RQ3: Does the documentation comment of each method affect the performance of VSM?

To examine if there is any correlation between each method length and the method’s documentation comment, the following hypothesis is defined:

- $H_6$: The null hypothesis $H_{6\text{-null}}$ is that $\rho_1 = 0$. While the alternative hypothesis $H_{6\text{-alt}}$ is that $\rho_1 \neq 0$.

- $H_7$: The null hypothesis $H_{7\text{-null}}$ is that $\pi_{SQ} \leq 0.5$. While the alternative hypothesis $H_{7\text{-alt}}$ is that $\pi_{SQ} > 0.5$.

- $H_8$: The null hypothesis $H_{8\text{-null}}$ is that $\rho_2 = 0$. While the alternative hypothesis $H_{8\text{-alt}}$ is that $\rho_2 \neq 0$.

RQ4: Does the query length have any effect on the performance of VSM?

The following hypothesis is defined:

- $H_9$: The null hypothesis $H_{9\text{-null}}$ states that $\pi_{SQ} \leq 0.5$.

- $H_6$: The null hypothesis $H_{6\text{-null}}$ is that $\rho_1 = 0$. While the alternative hypothesis $H_{6\text{-alt}}$ is that $\rho_1 \neq 0$.

- $H_7$: The null hypothesis $H_{7\text{-null}}$ is that $\pi_{SQ} \leq 0.5$. While the alternative hypothesis $H_{7\text{-alt}}$ is that $\pi_{SQ} > 0.5$.

RQ5: Is there any correlation between the query length and the length of the method retrieved with the highest rank for that query?

The sample used in RQ4 is used to examine if there is any relationship between the query length (bug report’s length), and the length of the retrieved method with highest rank for that query:

- $H_5$: The null hypothesis $H_{5\text{-null}}$ is that $\rho_2 = 0$. While the alternative hypothesis $H_{5\text{-alt}}$ is that $\rho_2 \neq 0$.

RQ6: Does the product name or the component name mentioned in the bug report have any impact on the performance of VSM?

To examine if the presence of the product or the component name in the bug report - where the bug report in our case includes a title, a description and a summary- has any effect on the performance of VSM, the following hypothesis is defined:

- $H_5$: The null hypothesis $H_{5\text{-null}}$ is that $\pi_{QWP} \leq 0.5$. While the alternative hypothesis $H_{5\text{-alt}}$ is that $\pi_{QWP} > 0.5$.
to examine if the proportion of queries that include product or component name, are larger than the proportion of the queries without product or component name from a sample of 55 points randomly chosen from the 121 population points.

### V. EXPERIMENTAL RESULTS

#### A. Experimental Results for Research Questions

**RQ1:** Can we prove that VSM outperforms LSI and LDA on the method level over any dataset that follows the variables of interest described in Section III.D?

variables: the tested IR techniques (VSM, LSI and LDA) are the independent variables. The dependent variables are the values of AP and RR.

Comparing VSM with LSI (testing $H_1$ and $H_2$): Table 1 shows that the sample’s MAP of LSI is 0.005 is smaller than the sample’s MAP of VSM 0.04. The $z$-test equals 3.58, which is greater than the standard score of 1.645. Therefore, we can reject $H_{1\text{-null}}$ and accept $H_{1\text{-alt}}$ that states that VSM has higher MAP on finding relevant methods than LSI. Table 1 also shows that MRR of LSI is 0.005, which is smaller than the sample’s MRR of VSM 0.06. The $z$-test score is 4.19, which is greater than the standard score of 1.645. Therefore, we can reject $H_{2\text{-null}}$ and accept $H_{2\text{-alt}}$ that states VSM has higher MRR on finding relevant methods than LSI.

Comparing VSM with LDA (testing $H_3$ and $H_4$): Table 1 shows that the sample’s MAP of LDA is 0.0006 which is smaller than the sample’s MAP of VSM 0.04. The $z$-test is 4.35, which is greater than the standard score of 1.645. The $P$-value is 6.64 × $E^{-6}$, which is notably less than the significance level of 0.05. Therefore, we can reject $H_{3\text{-null}}$ and accept $H_{3\text{-alt}}$ that states VSM has higher MAP on finding relevant methods than LDA. The results in Table 1 also show that MRR of LDA is 0.0006, which is smaller than MRR of VSM 0.06. The $z$-test score is 4.74, which is greater than the standard score of 1.645. Therefore, we can reject $H_{4\text{-null}}$ and accept $H_{4\text{-alt}}$ that states VSM has higher MRR on finding relevant methods than LDA.

**RQ2:** Does the method length have any effect on the performance of VSM?

variables: Short Methods (SM) is the dependent variable for $H_5$. Since it is a one-sample test, we do not have independent variables.

Testing the percentage of SM (testing $H_{5\text{-null}}$): Figure 2 (a) shows the proportion of short methods in high rank positions. Table 2 shows that the number of short methods in the sample is 37 with a proportion of 0.67, which is greater than the claimed value 0.5. The $z$-test is 2.56, which is greater than the standard score of 1.645. The $P$-value is 0.0052, which is less than the significance level of 0.05. Therefore, we can reject $H_{5\text{-null}}$ and accept $H_{5\text{-alt}}$. This implies that the method length has a strong impact on the performance of VSM on the method level.

**RQ3:** Does the documentation comment of each method affect the performance of VSM?

variables: Method Length (MLE) and Method Documentation Comment (MDC) of each method are the independent variables for $H_6$.

Determining the correlation between MLE and MDC (testing $H_{6\text{-null}}$): Table 3 shows a low correlation value of 0.1157. The $r$-test is 0.85. The $P$-value is 0.4, which is higher than the significance level of 0.05. Figure 3 depicts the low correlation between the method length and its documentation comments. Therefore, we cannot reject $H_{6\text{-null}}$ which means that there is no correlation between the method length and its documentation comment.

### Table I

| Hypothesis Test of Means from a Sample of 280 Points |
|------------------------|-----------------|-----------------|-----------------|-----------------|
| $H_1$                  | $H_2$           | $H_3$           | $H_4$           |
| Average P             | RR              | Average P       | BR              |
| VSM LSI               | VSM LSI         | VSM LDA         | VSM LDA         |
| 0.025                  | 0.041           | 0.025           | 0.041           |
| 0.009                  | 0.004           | $9.9 \times E^{-5}$ | 0.006           |
| z-Value: 3.58          | z-Value: 4.19   | z-Value: 4.35   | z-Value: 4.74   |
| P-Value: 0.00017       | P-Value: 1.4 × $E^{-5}$ | P-Value: 6.64 × $E^{-6}$ | P-Value: 1.04 × $E^{-6}$ |

**Fig. 1.** The comparison between VSM, LSI and LDA
RQ4: Does the query length have any effect on the performance of VSM?

Variables: Short Queries (SQ) is the dependent variable of $H_7$. Since it is a one-sample test, we do not have independent variables.

Testing the percentage of SQ (Testing $H_7$-null): Figure 2 (b) shows the proportion of short queries associated with methods in high rank positions. The results in Table 2 show that the number of short queries in the sample is 35 with a proportion of 0.64, which is greater than the proportion of the long queries.

The $z$ test score is 2.022, which is greater than the standard score of 1.645. The $P$-value is 0.02, which is less than the significance level of 0.05. Therefore, we can reject $H_7$-null and accept $H_7$-alt. This means that the query length has a significant effect on the performance of VSM’s performance on the method level.

RQ5: Is there any correlation between the query length and the length of the method retrieved with the highest rank for that query using VSM?

Variables: Query Length (QLE) and Method Length (MLE) for the method retrieved with highest rank for that query using VSM are the independent variables for $H_8$.

Determining the correlation between MLE and QLE (Testing $H_8$-null): Table 3 shows a low correlation value of 0.197. The $t$ test is 1.46. The $P$-value of 0.14, is higher than the significance level of 0.05. Figure 4 depicts the low correlation between the query length and the length of its relevant method. Therefore, we cannot reject $H_8$-null. Which means there is no correlation between the query length and the length of the method retrieved with highest rank for that query using VSM.

RQ6: Does the product name or the component name mentioned in the bug report have any impact on the performance of VSM?

Variables: Queries With Product or Component name (QWPC) is the dependent variable of $H_9$. Since it is a one-sample test, we do not have independent variables.

Testing the percentage of QWPC (Testing $H_9$-null): Figure 2 (c) shows the proportion of queries with product or component names associated with methods in high rank positions. Table 2 shows that the number of queries with product or component name in the sample is 24, with a proportion of 0.436, which is lower than the proportion of the queries without product or component name. The $z$ test is 0.94, which is lower than the standard score of 1.645. The $P$-value is 0.36, which is greater than the significance level of 0.05. Therefore, we can not reject $H_9$-null. This implies that specifying the product name or the component name in the query does not affect the performance of VSM.

B. Discussion of the Results

Several conclusions can be inferred from the experimental results:

1) Our results prove that VSM outperforms LSI and LDA in retrieving relevant methods for bug localization purposes.
2) The majority of methods in high rank positions are short methods, which proves that VSM favors short documents over long documents. Developers can modify VSM so that it can overcome this shortage of ignoring long documents.
3) VSM have the tendency to correctly fix short bug reports more than long bug reports. In other words, most of the methods in high rank positions are associated with

### Table II

Hypothesis test of proportions from a sample of 55 points

<table>
<thead>
<tr>
<th>H</th>
<th>Var</th>
<th>Count</th>
<th>( \hat{p} )</th>
<th>z</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( H_5 )</td>
<td>SM</td>
<td>37</td>
<td>0.67</td>
<td>2.56</td>
<td>0.0052</td>
</tr>
<tr>
<td>( H_7 )</td>
<td>SQ</td>
<td>35</td>
<td>0.64</td>
<td>2.022</td>
<td>0.02</td>
</tr>
<tr>
<td>( H_9 )</td>
<td>QWPC</td>
<td>24</td>
<td>0.436</td>
<td>0.94</td>
<td>0.827</td>
</tr>
</tbody>
</table>

### Table III

Hypothesis test of correlation from a sample of 55 points

<table>
<thead>
<tr>
<th>H</th>
<th>Var</th>
<th>r</th>
<th>t</th>
<th>Degrees of freedom</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( H_6 )</td>
<td>MLE</td>
<td>0.1157</td>
<td>0.848</td>
<td>n-2 = 53</td>
<td>0.4</td>
</tr>
<tr>
<td>( H_8 )</td>
<td>QLE</td>
<td>0.197</td>
<td>1.46</td>
<td>n-2 = 53</td>
<td>0.14</td>
</tr>
</tbody>
</table>
short bug reports, which means that short queries have a higher probability to be correctly fixed than long queries when using VSM. To help obtain better results, developers using VSM in bug localization can encourage bug reporters to write shorter summary and/or description.

4) Unlike the common notion among developers that well documented comments help detecting the similarity between the bug and the flawed method, our results show that the method comments does not contribute to the high ranks of the methods.

5) Although VSM favors short documents and short queries over long documents and long queries, there is no relationship between the query length and the document length. In other words, short methods found in high ranks can either be associated with long queries or short queries.

6) Specifying the product name or the component name where the bug was found does not increase the chances of finding the relevant methods.

VI. THREATS TO VALIDITY AND LIMITATIONS

The study we presented suffers from several limitations and threats to validity:

• Our results can only be generalized to projects that match the variable of interest described in Section III.D.

• The superiority of VSM applies only to LSI and LDA when used on the method-level. The tested text models might exhibit different behavior on the class-level.

• In determining the query length, we only considered the title, summary and description of the bug report. Results may vary when considering other sections of the bug report such as the testers’ comments.

• When calculating the term weights, all sections of the query were assigned the same importance. Assigning different weights to different query section may produce different results.

• The methods used in determining the method length were written in Java for the Eclipse v.3.1 project. Methods written for different projects in different languages might vary in length, which might affect the results.

VII. CONCLUSION AND FUTURE WORK

We have used hypothesis testing and statistical inference to perform the following:

• First, prove the superiority of VSM over LSI and LDA in information retrieval based bug localization on the method level.

• Second, study the impact of external factors such as: query’s length, methods’ length and the presence of documentation comments on the performance of VSM. We also used correlation analysis to detect the relationship between the query’s (bug report) length and method’s length for successfully located bugs.

Future work involves studying the factors associated with method change proneness. Based on the results, a hybrid tool of VSM and a logistic model of the affecting factors will be developed to help improve the information retrieval in bug localization on the method level.

The dataset and tools used in this study are available at: homepages.wmich.edu/~mnn7262/
VIII. ACKNOWLEDGMENT

This work is supported by Western Michigan University. We thank Faisal Allaboon for his help in data sampling.

REFERENCES


