An Improved VSM-based Post-Requirements Traceability Recovery Approach Using Context Analysis

Jiale Zhou, Yue Lu, Kristina Lundqvist
Mälardalen Real-Time Research Centre, Mälardalen University, Västerås, Sweden
{zhou.jiale, yue.lu, kristina.lundqvist}@mdh.se

***Abstract*—Automatically generating traceability links between software development artifacts existing throughout systems development life cycle, is becoming ever more important for requirements traceability. It remains an open software engineering challenge, especially for legacy systems, when the demand for minimizing human intervention is considered. The Vector Space Model (VSM), a notably known information retrieval technique, attempts to remedy the situation by reducing the required manual effort. One limitation of VSM is its low-level performance in practice, which can be improved by involving human intervention in the requirements traceability process earlier. The contribution of this paper is to present an improved VSM-based post-requirements traceability recovery approach by using a novel context analysis. This is done by firstly removing redundant information in the search space of the artifacts wrt a requirement, and then using both requirement and context queries to refine the results given by the standard VSM. In this way, the subsequent artifacts from the source requirement are more likely to be retrieved in the recovery process. Our approach is evaluated by using two chosen datasets (i.e., eTour and iTrust), of which results show that the proposed approach can achieve better performance in terms of discovering more true trace links and obtaining higher quality lists of traceability links than the standard VSM.**

Keywords*—*traceability links recovery; post-requirements traceability; vector space model; situational and discoursal context analysis; context query;

I. Introduction

Requirements Management (RM) is a critical activity for system development. It should be carried out for all the phases of systems development life cycle (or the software development process in other words), other than a single phase. RM assumes requirements elicitation, tracking and preservation of integrity, and handles a large amount of existing software development artifacts (i.e., the artifacts hereafter). The quality of RM is in the importance of success in system development, mattering to e.g., customers satisfaction, requirements coverage, efficient utilization of money, time and other resources, the likelihood of generating errors and bugs. For development of safety critical systems [1], whose failure could result in loss of life, significant property damage, or damage to the environment, the pertaining RM should be done as good as possible.

The heart of RM is Requirements Traceability (RT), which is defined as the ability to "describe and follow the life of a requirement, in both forward and backward directions" [2]. RT provides critical support for system developers throughout the entire life-cycle of the software development process. Tracing requirements can help to determine whether or not the developers have refined requirements into lower-level design components, implemented components into executable systems, tested and maintained the systems effectively. Although many efforts [3], [4], [5], [6] have been devoted to automatic traceability creation, especially for legacy systems [7], with limited human intervention, such creation working throughout the entire system life cycle, remains a challenging issue which prevents a wide-scale application of traceability in the software engineering practice.

Automated Information Retrieval (AIR) techniques, notably the algebraic model Vector Space Model (VSM) [8] (referred to as the standard VSM hereafter), attempts to reduce the manual effort of retrospectively building traceability. The standard VSM produces ranking lists of candidate trace links by computing the similarity between requirements and subsequent documents based on the occurrence of terms. Different strategies are applied to prune undesired links, and finally, the resulting trace lists are vetted by human analysts. One problem with VSM is that human intervention is involved after the candidate traceability links have already been retrieved and ranked, often resulting in a low-level performance [9]. Topic modeling, a type of statistical model for discovering the abstract "topics" that occur in a collection of documents, are featured by latent semantic index [10], probabilistic latent semantic indexing [11], latent dirichlet allocation [12]. Topic modeling utilizes the context (or more precisely, the content) of requirements to improve trace links retrieval automatically. Later, the work about using prospective capture with topic modeling [13] has been presented, which is efficient in searching trace links and allows for the semantic categorization of artifacts as well as the topical visualization of the software systems. However, none of the mentioned work uses *implicit* information given by requirements, e.g., title, footnote and appendix, which may contribute greatly to improve the traceability results.

Our goal in this paper is to tackle the above mentioned problem, by improving the accuracy of traceability results of the standard VSM by using a novel context analysis. To be specific, we consider to bring human judgment into an earlier phase of the RT recovery process, by leveraging the existing context of the requirements through the use of experts' experience, and giving the subsequent artifacts of a source requirement a higher chance to be retrieved in the process. In particular, the technical contributions presented in this paper

are two-fold:

1) We introduce the VSM-based context analysis, which uses two different context analysis methods to improve the traceability results given by the standard VSM. By using two novel context analysis methods, we firstly filter out some redundant information in the scope of the artifacts which are relevant to the source requirement, and then use both requirement and context queries to refine the results of the standard VSM.
2) We present our new approach which improves the traceability results, in terms of discovering more true trace links and obtaining more accurate ranking lists of the subsequent artifacts toward a source requirement, comparing the standard VSM. Specifically, such improvements have been demonstrated by two chosen datasets from one electronic tourist guide application and one medical application.

The remainder of the paper is organized as follows. Section II introduces the background theory and related work. Section III firstly gives an overview of our analysis, and then presents two proposed context analysis methods together with the implementation of the algorithm in detail. Next, Section IV that describes the evaluation setup, two chosen datasets and implementation as well as evaluation results, and finally, conclusions and future work are drawn in Section V.

II. Background

This section firstly describes the background on the Vector Space Model technique in Section II-A, followed by some detailed introduction about context-based analysis as well as some related work in Section II-B.

A. Vector Space Model

In Vector Space Model (VSM) [14], given the entire collection of unique terms $T = \{t_1, \ldots, t_n\}$ in a document collection with N documents, each document d is represented as a vector $d = \{w_{d1}, \ldots, w_{dn}\}$ consisting of n unique terms from the corpus with an assigned weight w_{di} using a certain weighting scheme. Therefore, the similarity score, denoted as $sim(q, d)$, between the source query document q and the target document d is calculated by using the cosine of the angle between their vectors:

$$sim(q,d) = \frac{\sum_{i=1}^{n} w_{qi} \cdot w_{di}}{\sqrt{\sum_{i=1}^{n} w_{qi}^2 \cdot \sum_{i=1}^{n} w_{di}^2}} \quad (1)$$

Moreover, here we introduce the *term frequency* - inverse *document frequency*, i.e., *tf-idf*, which is also adopted as the VSM weighting scheme of our proposed context analysis (to be introduced in Section III):

$$w_{qi} = tf_i(q) \cdot idf_i, w_{di} = tf_i(d) \cdot idf_i \quad (2)$$

where $tf_i(q)$ and $tf_i(d)$ are measured by the number of times the term t_i occurs in the query document q and the target document d respectively, and idf_i is computed as $log(\frac{N}{df_i})$, where df_i is the number of documents containing the term t_i.

The standard VSM described above has been applied to the requirements tracing problem [8]. In requirements engineering, a requirement is typically traced forward across all the artifacts in systems development life-cycle. In this case, the document collection would be the entire set of requirements and artifacts (in this paper, we use the two terms *documents* and *artifacts* interchangeably). In applying the standard VSM, we select the requirement r (as the query q) and repeatedly calculate the similarity score with every document in a collection of documents. In this way, a descending-ordered ranking list of candidate trace links to the requirement will be generated by the VSM.

However, the performance of the standard VSM is not always satisfactory [15]. From our viewpoint, it is easy to understand the reason for its low-level performance in practice. This is mainly because that the standard VSM only takes into account the occurrence of terms in the documents as its inputs, but not the quality of such inputs. Dekhtyar [9] also argues that the accuracy of the initial traceability results is the most important factor, impacting the accuracy of the final results. In other words, if the inputs to the standard VSM can express the intentions of the requirement more accurately by containing more enhanced semantics, the better result of the standard VSM can be achieved; otherwise the generated ranking list of trace links will be at a very low accuracy of relevance to the source requirement.

Clearly, one solution to the above problem is that if we can improve the quality of inputs to the standard VSM, by expressing the intentions of the source requirement as good as possible and bringing human intervention to an earlier stage of the requirements traceability process, a better performance given by the standard VSM would be expected. This is like: In a project or product development, the earlier some useful and relevant information given by seniors, experts to (new) developers, the better developers know their tasks, and the more they can contribute at work.

B. Context-based Analysis

Connolly [16] introduces that an important fact about communication is that it always takes place in a context. Suppose that we are interested in studying an intention of others, which we denote as I. The context consisting of whatever surrounds I, serves to facilitate the communication between speakers and listeners. Furthermore, the intentions of people are usually reflected by the context of the conversation between them, based upon their understanding and interpretation.

Similarly, the intentions of the requirement are often indicated by its context, which is comprised of *discoursal* context, *situational* context and the content of the requirement in terms of *sub-flow* information. To be specific, discoursal context incorporates any relevant texts or statements that surround the requirement intention, and hence is helpful to determine the meaning of the requirement; while there is some objective information which is usually associated with the requirement, such as creators, recipients, creation time, modification time, the place of communication, and so on. Such information is situational in itself, and could contribute to study the intentions of the requirement greatly. Note that the analysis of situational context should be considered as a subjective analytical method,

since it is highly dependent on the understanding about the requirement, resulting in the appropriate judgment given by the persons involved, who are usually requirements analysts.

In order to have a better understanding about the idea of using context-based analysis to improve the performance of the standard VSM in terms of providing more precise inputs to the latter, we give the following example of a little boy buying ice cream: After the boy decided to buy an ice cream, he wrote the words "ice cream" on his mother's shopping list. Later on, he updated the status of his social networking website (e.g., Twitter [17]) with the statement "Ice cream is my favorite dessert!". In this case, we can consider his intention "buying an ice cream" as the requirement, the shopping list as the document A and his Twitter status as the document B. Since the words "ice cream" occurred with the same frequency in both documents, it is thereby very hard to conclude which document has the higher similarity score by using the standard VSM. Nevertheless, we have noticed that although both documents A and B contain the term "ice cream", such terms exist in different context: The document A is a shopping list which reminds the owner what she or he needs to buy at the store, while the document B is a status which shares the feeling of the boy. Furthermore, the document A is also with the implicit expression about *buying* items in a shop, granted by the discoursal context of the list, i.e., a *shopping* list. Therefore, the document A is more relevant to the requirement about buying an ice cream, when the discoursal information about the title of the list is considered. With the purpose of improving the performance of the standard VSM in this case, such information about discoursal context should be used in the analysis to provide precise inputs to the standard VSM.

III. The Proposed VSM-Based Context Analysis Approach

A requirement is a singular documented physical and functional need that a particular product or process must be able to perform. It can also be regarded as one or a set of intentions, of which the detailed interpretations are subsequent different software development artifacts, e.g., design documents, source code, testing and maintenance documents. Accordingly, the traceability recovery process is about finding different relevant interpretations of such intentions. In matters of interpretation, it is very important for us to understand context. Since context not only plays a significant role in influencing the way that the intention is interpreted with a certain level of satisfaction, but also is a rather experienced construct, which can help project experts to improve the performance of traceability recovery process. In this work, such useful requirements context is extracted and used in the context, with the aid of experts' experience.

In the following, we introduce the proposed VSM-based context analysis for post-requirements traceability recovery process in detail. Firstly, an overview of the analysis is given, which is followed by the description of the situational context analysis of software development artifacts (i.e., documents in this work) as well as the discoursal context analysis of requirements in Section III-B and III-C respectively. Also, in this section the overall algorithm is presented in pseudo-code format.

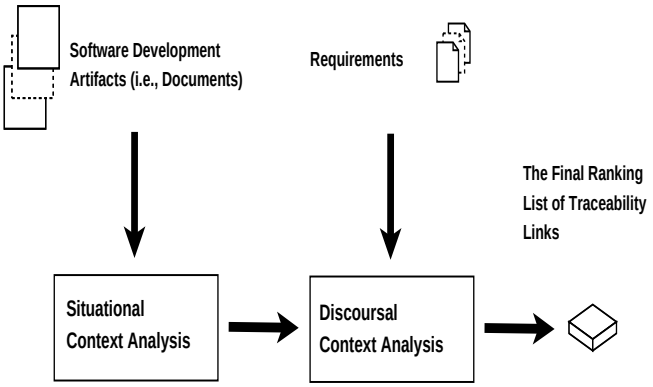

Fig. 1. The work flow about our proposed VSM-based context analysis approach.

A. Overview of Our VSM-based Context Analysis

The main idea of our proposed VSM-based context analysis is: 1) to reduce the search space of documents by filtering out some redundant artifacts using situational context analysis based on experts' experience, and 2) to fuse the results given by the standard VSM using both a requirement query and a new context query, in order to obtain more accurate traceability links between the artifacts and the source requirement. To be specific, the basic approach followed in this work is summarized by the following three stages:

1) The first stage of our analysis is to apply situational context analysis by using experts' experience. At this step, some non-related artifacts existing in the scope of the software development process are filtered out by using some objective information associated with the artifacts. As the evaluation (to be introduced in Section IV-C2) confirmed that by using situational context analysis, more accurate traceability links are retrieved, which is also in line with the thoughts, i.e., by involving the human intervention to the early stage of requirements traceability recovery process, the quality of the lists of traceability links can improved.
2) Next, the standard VSM is firstly adopted to generate a ranking list of trace links between artifacts and the source requirement. Then, our discoursal context analysis extracts some useful information from the source requirement to generate a new context query, which will be used again by the standard VSM to generate another ranking list.
3) Finally, the two generated ranking lists are fused together to give a final ranking list containing recalculated similarity scores, based upon the rationale we proposed in this work.

Figure 1 shows the detailed work flow of our approach.

B. Situational Context Analysis of Artifacts

As we briefly introduced previously, our proposed situational context analysis is responsible for selecting *candidate artifacts* out of any artifacts existing in the software development process, in terms of extracting the *situational context*. Examples of such situational context include, but are not limited to the following: the *type*, *location*, *creation*

time, *modification time*, *package/class information* in object-oriented programming, and other objective information associated with the artifacts. Moreover, in requirements engineering, the permitted traceability relations between artifacts can be defined according to some intended usage, e.g., a certain type of relationships between artifacts can be required or ignored intentionally in the requirements traceability recovery process. In order to have a better understanding about the motive for proposing situational context analysis, we give a simple example here: Supposed that we want to build traceability links of the Java source code files in two separate packages and one requirement in a project. Further, such Java files particularly describe two fruits *orange* and *apple*. The requirement here is that *to select the tasty apples*. Moreover, the name of two packages contains some keyword, e.g., orange or apple. Therefore, in this case, we can refine the search space of the subsequent Java code files toward the requirement by removing the files in the package which is associated with orange. In doing this, we have brought human intervention in the early stage of requirements traceability recovery process, and clearly, such situational context analysis (as a subjective context analysis approach) is helpful to avoid the cases for linking the requirement to some irrelevant artifacts or mistakenly missing the links to intended artifacts from a requirement. Some of its other advantages are: we also save time and perform the post-requirements traceability recovery process more efficiently. At the current stage, we perform the situational context analysis manually, and the implementation of the analysis is shown in lines from 3 to 10 in Algorithm 1 (to be introduced in Section III-D). Next we give the definition of situational context in this work.

Definition 1. *Situational context refers to some objective information associated with artifacts (i.e., documents in this work), of which examples can be authors, recipients, type, location, creation time, modification time, package information (in object-oriented programming) of the artifact. It is used to reduce the search space of the documents toward the source requirement in the post-requirements traceability recovery process.*

C. Discoursal Context Analysis of Requirements

Our discoursal context analysis is mainly working with requirements. At its first step, we use the requirement r as the query (i.e., the requirement query) and the document d (i.e., one of the candidate documents obtained by using situational context analysis) as inputs to the standard VSM, which generates a ranking list of trace links with similarity scores. Next, we extract the discoursal context of the requirement r to form a context query c, and then apply the VSM by using c and d as inputs to get another ranking list. Finally, as shown in Equation 3, we combine the similarity scores of two ranking lists in order to calculate the final similarity score of document d and requirement r, based on the following rationale:

1) The first term in Equation 3 implies that if a list is shorter than the other one, the corresponding query matches more a specific set of documents. Therefore, its score should play a bigger role in the final similarity score $sim\_vsm-ca(r,c,d)$ in the equation.
2) The second term in Equation 3 indicates that the higher rank of d has in the ranking list, the more likely the candidate artifact has a true link toward the source requirement. Hence, its score should play a more important role in the final similarity score $sim\_vsm-ca(r,c,d)$ in the equation.

In other words, our discoursal context analysis in practice provides the standard VSM with both the requirement query and the context query. Typically, such a combination contains more enhanced semantics, accurately representing the intentions of the requirement. As a result, the performance of the standard VSM toward finding true trace links can be improved. One example of such discoursal context used in our evaluation is shown by Figure 2, where the discoursal context is the title of the use case. In the following, we give the definition of discoursal context used in this work.

Definition 2. *Discoursal context of the requirement r refers to its title, footnote and appendix. Such information is used as a context query, together with the requirement query, to form a combination which contains more enhanced semantics and expresses more accurate intentions of the requirement r, compared with the requirement query itself.*

D. Algorithm Description

The precise description of the algorithm using pseudo-code is outlined in Algorithm 1, which takes three parameters and returns a final ranking list of trace links in the end of its execution.

Parameters:
D: list - the collection of documents D
r: string - the requirement query r
c: string - the context query c of the requirement r
Returns:
$LIST_{vsm-ca}$: list - the final ranking list containing the final similarity score of the document d, the requirement r and the context query c, in the descending order of relevance to the requirement r.

Algorithm 1 $VSM-CA(D,r,c)$

1: $success \leftarrow false, m \leftarrow 0, l \leftarrow 0$
2: $D \leftarrow d_1, d_2, ..., d_{n-1}, d_n$
3: **for all** i such that $1 \leq i \leq n$ **do**
4: **if** $relevanttoRequirement(d_i, r) == 0$ **then**
5: $D^{'} \leftarrow d_i^{'} \leftarrow d_i$
6: $m \leftarrow m+1, success \leftarrow true$
7: **else**
8: $success \leftarrow false$
9: **end if**
10: **end for**
11: $c \leftarrow discourse(r)$
12: **for all** $d_i^{'}$ such that $1 \leq i \leq m$ **do**
13: $LIST_r \leftarrow VSM(r, d_i^{'})$
14: $LIST_c \leftarrow VSM(c, d_i^{'})$
15: $l \leftarrow l+1$
16: **end for**
17: **for all** $d_i^{'}$ such that $1 \leq i \leq l$ **do**
18: $LIST_{vsm-ca} \leftarrow sim\_vsm-ca(r,c,d^{'}{}_i)$
19: **end for**
20: **return** $LIST_{vsm-ca}$

$$sim\_vsm{-}ca(r,c,d) = \frac{len(c)}{len(r)+len(c)} * \frac{len(r)-rank_r(d)}{len(r)} * sim(r,d) + \frac{len(r)}{len(r)+len(c)} * \frac{len(c)-rank_c(d)}{len(c)} * sim(c,d) \quad (3)$$

where $len(c)$ and $len(r)$ are the lengths of the ranking list of the context query c and the requirement query r respectively; $rank_c(d)$ and $rank_r(d)$ are the ranks of the document d in both ranking lists, corresponding to c and r separately; $sim(r,d)$ and $sim(c,d)$ are the similarity scores obtained by using the standard VSM.

IV. Evaluation

This section is split into three parts: Section IV-A introduces the setup of the evaluations performed in this case study, including interesting evaluation properties as well as performance and correctness measure. Section IV-B outlines the two chosen datasets and some implementation details. Finally, Section IV-C presents the results and improvements over the standard VSM.

A. Evaluation Setup

In the evaluation of the proposed method, we are interested in the number of true trace links, some improvements over the standard VSM, given by both our situational context analysis and the proposed VSM-based context analysis as a whole, as well as computing time of our algorithm. Note that in order to validate the quality of the generated list of trace links, we compare the results with some known answer sets, i.e., the lists of the known true trace links for the chosen datasets which can be determined manually in advance.

There are many different measures for evaluating the performance of Information Retrieval (IR) systems. In our case, for correctness and improvement measures of the experimental data, we present the precision score [18] of each individual requirement in the two chosen datasets (i.e., on the individual requirement level), and Mean Average Precision (MAP) [19] of every chosen dataset (i.e., on the dataset level). To be specific, for precision score, it is the fraction of the documents retrieved that are relevant to our source requirement, which can be expressed by the following equation:

$$precision = \frac{|D_{rel} \cap D_{ret}|}{|D_{ret}|} \quad (4)$$

where D_{rel} represents the collection of documents which are relevant to the source requirement, and D_{ret} is the collection of retrieved documents.

MAP, as one of the most frequently used IR measures, considers the rank of the retrieved links, meaning that the higher the MAP score is, the better quality of the retrieved ranking list of trace links is in terms of requirements relevance. In particular, given a collection of queries Q, a set of related documents D_a contained in the answer set, the MAP score of the ranking list L of retrieved documents for the given query q, is defined as below:

$$MAP = \frac{1}{|Q|} \sum_{q=1}^{Q} \frac{1}{|D_a|} \sum_{d=1}^{D_a} SCORE_{rank}(d,L) \quad (5)$$

where $SCORE_{rank}(d,L)$ represents the rank score of the document d in the list L. The higher the rank of d is, the larger rank score the document has.

B. The Chosen Datasets and Implementation

We apply our method to analyze two chosen datasets eTour [20] and iTrust [21], which are one electronic tourist guide application and one medical application, separately developed at Center of Excellence for Software Traceability and North Carolina State University in United States. Each of the datasets consists of requirements, testing documents, source code, and a traceability matrix which is comprised of 308 (for eTour) and 139 (for iTrust) pre-determined true trace links between requirements and source code, individually.

In the two chosen datasets, the requirements are introduced as *use cases*, of which one example is shown in Figure 2. In this work, there are 57 and 44 use cases in eTour and iTrust respectively. Our target is to build trace links from the sub-flows of each use case to the source code files, by using the context of the use case. In this work, we assume that the sub-flows in the same use case have the same discoursal context, which is the title of the use case. Before applying the standard VSM, we should also pre-process all the documents in a pipelined fashion by using *tokenizer*, *stop-words remover* and *stemmer*, as introduced by Kong in [5]. According to the situational context analysis, the Java code files of iTrust can be divided into two categories, i.e., "action" and "DAO", which gives two types of search space of the documents. However, for eTour, we cannot perform the situational context analysis, due to the lack of clear information available that should be given by well-structured source code files (while it is not the case based on the current implementation of eTour). We also use the Lucene library [22] in our implementation, which is the well-known VSM with tf-idf weighting scheme, and has been considered as the default IR model by many work [].

Our testbed is running Mac OS X, version 10.6.8, and the computer is equipped with the Intel Core Duo CPU i7 processor, 4GB RAM and a 256KB L2 Cache. The processor has four cores and one frequency level: 2.2 GHz.

C. Evaluation Results

In this section, we discuss the evaluation results, by firstly presenting the improvement from the perspective of the retrieved true trace links, which is followed by the improvement in the ranking lists of traceability links given by our VSM-based context analysis and experiments summary.

1) Improvement in True Trace Links Retrieved: Here we evaluate the improvement in our VSM-based context analysis method in terms of the number of obtained true trace links, comparing the number derived with the standard VSM. In Table I, the values in Column *True trace links*, *VSM*, *VSM-SA* and *VSM-CA* are the values of the number of the true trace links for each chosen dataset, the number of the true trace links retrieved by the VSM, our situational context analysis and the

```
UC Authenticate Users Use Case
  Sub-flows:
    [S1] If the security question/answer has been set (it is not null), present security question and obtain answer.
    [S2] If answer to security question is correct, allow user to change their password. An email notification is sent.
```

Fig. 2. An example shows one use case in ITrust.

TABLE I. OUR PROPOSED VSM-BASED CONTEXT ANALYSIS CAN RETRIEVE MORE TRUE TRACE LINKS, COMPARING THE STANDARD VSM, IN TERMS OF HAVING THE 3.01% INCREASE AT MOST.

Dataset	True trace links	VSM	VSM-SA	VSM-CA	Imprv. VSM-CA
eTour	308	290	–	290	0%
iTrust	139	133	137	137	**3.01%**

VSM-based context analysis methods. In Column *Imprv. VSM-CA*, the inherent improvement over the standard VSM given by our method is expressed as percentages which are calculated in the following way, i.e., $\frac{Links_{vsm-*} - Links_{vsm}}{Links_{vsm}} \times 100\%$, where $vsm - *$ represents either the VSM-based situational context analysis or the VSM-based context analysis (as a whole). As shown in the table, the results given by our method can find at most 3.01% more true trace links compared with the standard VSM. Recall that we cannot perform the situational context analysis of the dataset eTour, due to the lack of clear information caused by the not well-structured source code files based upon its current implementation. Therefore, such improvement is not applicable in Table I.

It is also interesting to note that after looking at the result of eTour, though there are no more true trace links given by our method comparing the standard VSM, our method can still guarantee that such true trace links retrieved by the standard VSM, can also be discovered in the requirements traceability recovery process.

2) Improvement Given by the VSM-based Context Analysis: Here we firstly evaluate the performance of the proposed method, in terms of introducing the improvement (as percentages) in the obtained precision score of each individual use case, comparing the standard VSM. Typically, the percentages of such improvements (i.e., $\frac{Precision_{vsm-*} - Precision_{vsm}}{Precision_{vsm}}$, where $vsm - *$ is either the VSM-based situational context analysis or the VSM-based context analysis as a whole) are shown in Figure 3 and 4. As we can see from both figures, for the most of the use cases of the two chosen datasets, our VSM-based context analysis can obtain higher precision scores with the maximum improvement 360.54% and 1 900% for eTour and iTrust respectively, compared with the standard VSM. In addition, for the situational context analysis of iTrust, such improvement is 1 037.88%.

Next we present the improvement of our method concerning the MAP score of two chosen datasets, in response to the standard VSM. Typically, the percentages of such improvement (i.e., $\frac{MAP_{vsm-*} - MAP_{vsm}}{MAP_{vsm}}$, where $vsm - *$ has the same meaning as we introduced previously) are shown in Column *Imprv. VSM-SA* and *Imprv. VSM-CA* in Table II. As shown in the table, the significant improvements made by our method, compared with the ones obtained by the standard VSM, are 35.04% and 62.51% increase in finding higher MAP scores for eTour and iTrust separately.

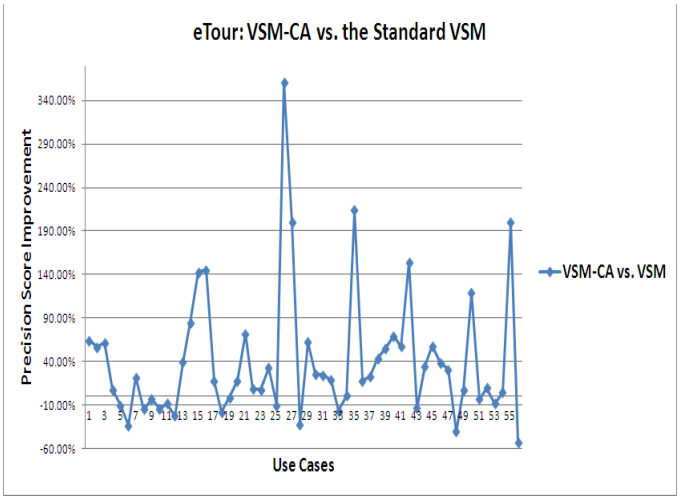

Fig. 3. The improvement (as percentages) in the precision score of each use case in eTour given by our VSM-based context analysis method, comparing the standard VSM.

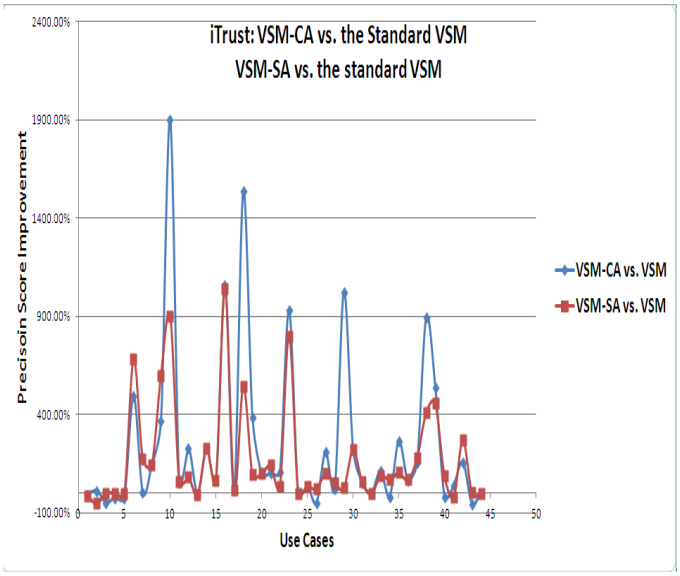

Fig. 4. The improvement (as percentages) in the precision score of each use case in iTrust given by our VSM-based context analysis (as a whole) and its VSM-situational context analysis methods, comparing the standard VSM.

3) Experiments Summary: Summarizing the above observations, our evaluation results have confirmed the following points:

- Our proposed VSM-based context analysis method for post-requirements traceability recovery process, corresponding to two different chosen datasets in practice, can retrieve more true trace links (at most 3.01% increase) and obtain higher MAP scores (in terms of 62.51% higher at

TABLE II. COMPARED WITH THE STANDARD VSM, OUR PROPOSED VSM-BASED CONTEXT ANALYSIS CAN OBTAIN THE HIGHER MAP SCORES OF THE GENERATED RANKING LIST OF TRACEABILITY LINKS, IN TERMS OF HAVING 35.04% AND 62.51% INCREASE FOR ETOUR AND ITRUST RESPECTIVELY.

MAP	VSM	VSM-SA	Imprv. VSM-SA	VSM-CA	Imprv. VSM-CA
eTour	0.24	–	–	0.33	**35.04%**
iTrust	0.36	0.57	59.72%	0.58	**62.51%**

most), compared with the standard VSM.

- The true trace links that are discovered by the standard VSM, have also been retrieved by our VSM-based context analysis.
- The computing time required by the trails of our method, on average took only a few minutes to compute. This is an important step toward handling real life-scale requirements traceability problems, exhibiting high degrees of variability.
- It is interesting to note that for some use cases, our proposed VSM-based context analysis method cannot achieve higher precision scores. More investigation on this is considered as part of our future work, which could be improved by employing some more advanced context extraction approach in the method.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a new Vector Space Model (VSM)-based technique which uses a novel context analysis to build trace links between requirements and other software development artifacts (i.e., documents in this work) retrospectively. Specifically, our approach uses both the situational and discoursal context analysis methods, to refine the search scope of the documents at first, and then improve the results obtained through the standard VSM by using both requirement and context queries. We have evaluated the approach by using two chosen datasets eTour and ITrust, through which the experiment results have shown that our approach can achieve better performance of finding more true trace links between requirements and documents as well as obtaining better quality of the retrieved list of traceability links, compared with the standard VSM technique.

For future work, we will investigate the reason why our method cannot obtain higher precision scores for some use cases in the chosen datasets, and improve the method e.g., by employing some advanced context extraction method as the hoped solution. We are also interested in discussing how the context of requirements and subsequent artifacts would impact the performance of VSM-based methods systematically. The comparison between our VSM-based context analysis and other automated information retrieval techniques, from the perspective of effective estimate of context similarity, as well as the completion of some extensive evaluation by using more datasets are also highly appreciated on our good side.

ACKNOWLEDGEMENTS

This work was partially supported by the Swedish Research Council (VR), and Mälardalen Real-Time Research Centre, Mälardalen University.

REFERENCES

[1] K. Lundqvist and L. Asplund, "A Ravenscar-Compliant Run-Time Kernel for Safety-Critical Systems," *The International Journal of Time-Critical Computing*, vol. 24, pp. 29–54, 2003.

[2] O. C. Z. Gotel and A. C. W. Finkelstein, "An Analysis of the Requirements Traceability Problem," in *Proceedings of the 2nd IEEE International Requirements Engineering Conference (RE' 94)*, 1994, pp. 94–101.

[3] N. Ali, Y.-G. Gueheneuc, and G. Antoniol, "Trust-Based Requirements Traceability," in *Proceedings of the 19th International Conference on Program Comprehension (ICPC' 11)*, 2011, pp. 111–120.

[4] C. Fautsch and J. Savoy, "Adapting the tf-idf Vector-Space Model to Domain Specific Information Retrieval," in *Proceedings of the 25th Symposium On Applied Computing (SAC' 10)*, 2010, pp. 1708–1712.

[5] W.-K. Kong and J. H. Hayes, "Proximity-based Traceability: An Empirical Validation using Ranked Retrieval and Set-based Measures," in *Proceedings of the 1st International Workshop on Empirical Requirements Engineering (EmpiRE' 11)*, 2011, pp. 45–52.

[6] A. Mahmoud, N. Niu, and S. Xu, "A semantic relatedness approach for traceability link recovery." in *Proceedings of the 20th IEEE International Conference on Program Comprehension (ICPC' 12)*, 2012, pp. 183–192.

[7] J. Kraft, Y. Lu, C. Norström, and A. Wall, "A Metaheuristic Approach for Best Effort Timing Analysis targeting Complex Legacy Real-Time Systems," in *Proceedings of the 14th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS' 08)*, 2008.

[8] G. Antoniol, G. Canfora, G. Casazza, A. D. Lucia, and E. Merlo, "Recovering Traceability Links between Code and Documentation," *IEEE Trans. Software Eng.*, vol. 28, no. 10, pp. 970–983, 2002.

[9] A. Dekhtyar, O. Dekhtyar, J. Holden, J. H. Hayes, D. Cuddeback, and W.-K. Kong, "On human analyst performance in assisted requirements tracing: Statistical analysis," in *Proceedings of the 19th IEEE International Requirements Engineering Conference (RE' 11)*, 2011, pp. 111–120.

[10] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and L. Beck, "Improving information retrieval with latent semantic indexing," in *Annual Meeting of the American Society for Info. Science 25*, 1988.

[11] T. Hofmann, "Unsupervised learning by probabilistic latent semantic analysis," *Machine Learning*, no. 42(1), pp. 177–196, 2001.

[12] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, no. 3, pp. 993–1022, 2003.

[13] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor, "Software Traceability with Topic Modeling," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE' 10)*, 2010, pp. 95–104.

[14] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, Nov. 1975.

[15] C.-H. Jane, G. Orlena, and Z. Andrea, *Software and Systems Traceability*. Springer, 2012.

[16] J. H. Connolly, "Context in functional discourse grammar," *Alfa : Revista de Linguística*, vol. 51, pp. 11–33, 2009.

[17] Twitter, https://twitter.com, 2013-02-20.

[18] D. POWERS, "Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation," *Journal of Machine Learning Technologies*, vol. 2(1), pp. 37–63, 2011.

[19] Wikipedia, http://en.wikipedia.org/wiki/Information_retrieval, 2013-02-20.

[20] Center of Excellence for Software Traceability, http://www.coest.org, 2013-02-20.

[21] N. C. S. University, http://agile.csc.ncsu.edu/iTrust/wiki/doku.php, 2013-02-20.

[22] E. Hatcher, O. Gospodnetic, and M. McCandless, *Lucene in Action*, 2nd ed., 2010.