A NOVEL STUDY FOR SUMMARY/ATTRIBUTE BASED BUG TRACKING CLASSIFICATION USING LATENT SEMANTIC INDEXING AND SVD IN DATA MINING

Ketki¹, Mr. Sanjay Kumar², Mr. RajKumar Singh Rathore³

¹PG Scholar, Galgotias College Of Engineering & Technology, Greater Noida(up), (India)  
²Assistant Professor, ³M.Tech Coordinator, Dept of CS/IT,  
Galgotias College of Engineering & Technology, Greater Noida(up), (India)

ABSTRACT

This paper presents a Latent Semantic Indexing (LSI) method for learning bug tracking concepts in document data. Each attribute in a vector provides the mark of participation of the document in data or term in the parallel concept. The objective to describe the concepts summary based, but to be capable to signify the documents and relations in a combined way for showing document-similarity, document-term, and term-term similarities or semantic relationship. Many technique for the implementing our research i.e. NLP, STEMMING, LUD, & SVD to the relevant similarity like bug report bug title, report summary etc., since every bug report, and mined developer’s name who fixed the bug reports from the developers activity data. We processed the mined textual data, and got the term-to-document matrix. Our proposed model different machine learning methods for on the basis of summary and attribute based classification of bug reports which produce the outlier bugs from large amount of data that improved accuracy and efficiency bug classification and relevant base bug indexing which is similar in meaning by using LSI and SVD Technique.

Keyword: BTS, LSI, SVD, LUD etc

I. INTRODUCTION

A bug tracking system is used for justification and sharing of bug reports to the maximum appropriate designers. An LSI based bug tracking scheme may decrease the software preservation time and advance its value by correct and timely task of new bug reports to the suitable reporters.

Figure 1: Diagram Shows Some use Cases Related to Bug Tracking and Fixing
In this paper, we present the current techniques overdue an LSI bug tracking system, which is based on the labeling of summary bug reports. In order to get an attribute bug tracking system we used these techniques and performed comparative experiments.

Latent Semantic Analysis arose from the problem of how to find relevant documents from search words. The fundamental difficulty arises when we compare words to find relevant documents, because what we really want to do is compare the meanings or concepts behind the words. LSA attempts to solve this problem by mapping both words and documents into a "concept" space and doing the comparison in this space.

Most NLP applications such as information extraction, machine translation, sentiment analysis and question answering, require both syntactic and semantic analysis at various levels. The motivation for LU decomposition is based on the observation that systems of equations involving triangular coefficient matrices are easier to deal with.

1.1 Indexing for Documents
Indexing is an essential part of the IR systems for two reasons. First, it optimizes the query performance and improves the respond times considerably by storing terms in an inverted file structure.

1.2 Searching
In this phase, the query terms are searched against the inverted index. All the documents that contain the occurrences of the query terms are retrieved. Depending on the application, the retrieval can be done even for the partially matched documents.

1.3 Ranking
The documents retrieved in the previous step are given scores according to the matching quality between the query terms and the documents. The documents are sorted according to this score, so that the most relevant documents are presented to the user on top of the retrieval list. Ranking process is highly dependent to the IR model. Semantic analysis at various levels. Traditionally, NLP research has focused on developing algorithms that are either language-specific and/or perform well only on closed-domain text.

II. LITERATURE REVIEW

Literature reviews are the maximum basic, yet very significant concept to set a theoretical basis. Their quality and usefulness greatly depends on the literature research process:

Turney [2, 9] proposed the Latent Relational Analysis (LRA) by covering VSM-based methods in three ways: a) lexical patterns are mechanically extracted from a corpus, b) the Singular Value Decomposition (SVD) is used to smooth the frequency data, and c) synonyms are used to travel variants of the word-pairs.

Danushka Bollegala et al. [11] planned the method which accepted web text wastes to mechanically extracts lexical patterns that describe the relation obscure by the two words in a word-pair and calculates the relational similarity between two word-pairs using a machine learning approach.

Shuji [10] shape a model to detect defect correction effort based on protracted association rule mining. They defined defect fixing effort as a variable and appropriate association rule mining to treat with such variables.

Data used are supported from Japan's Ministry of Economy, Trade and Industry (METI).
Emad and Walid [13] have developed an approach for predicting re-opened defects through Eclipse projects, their study depend upon some factors such as work habits dimension like: day which issue is closed, the bug report features dimension like: components, the bug correction dimension like: time needed to fix bug.

Weiß et al. predict the “fixing effort” or person-hours spent addressing a defect [7]. They leverage existing bug databases and historical context information. To make their prediction, they use pre-recorded development cost data from the existing bug report databases.

III. PROBLEM STATEMENT

The Software development corporations have to face a lot of problems while preserving physically all the preservation of the projects, their bugs and their status. This type of problem kinds the whole system an incompetent one and thus making a poor and unorganized working. In order to remove this type of problem, So that the paper is deliberate to develop. The LSI mechanism that resolves the problem contains simply of helpful a very large number of local Data. It also permits the manager of the system to modify the tracking procedure so that unnecessary documentation on the part of the problem solvers does not develop a waste of time.

IV. SYSTEM MODEL

The system model in linear algebra that decomposes a matrix into three factor matrices. SVD is a much more complex approach than other; however, once the original matrix has been moldy, operations on the matrix are rather quick.

\[
\text{SVD}\text{ is an extension of Latent Semantic indexing (LSI). LSI is a technique of investigating a group of terms and documents to find relationships between the two [7].}
\]

SVD takings as input a matrix of size \( m \times n \). This matrix is spoiled into three different matrices: \( U \Sigma V_T \). The output of these three matrices is a relationship to the original matrix.

\( S \) and \( V_T \) are both orthogonal matrices and \( \Sigma \) is a diagonal matrix. These three matrices are added identified as:

- \( S \) is the right singular vectors
  - Sized: \( m \times r \)
- \( V_T \) is the left singular vectors
  - Sized: \( r \times r \)
- \( \Sigma \) is the singular values
  - Sized: \( n \times r \)

\[\text{Fig 2: Singular Value Decomposition is a Mathematical}\]

SVD is an extension of Latent Semantic indexing (LSI). LSI is a technique of investigating a group of terms and documents to find relationships between the two [7].

SVD takings as input a matrix of size \( m \times n \). This matrix is spoiled into three different matrices: \( U \Sigma V_T \). The output of these three matrices is a relationship to the original matrix.

\( S \) and \( V_T \) are both orthogonal matrices and \( \Sigma \) is a diagonal matrix. These three matrices are added identified as:

- \( S \) is the right singular vectors
  - Sized: \( m \times r \)
- \( V_T \) is the left singular vectors
  - Sized: \( r \times r \)
- \( \Sigma \) is the singular values
  - Sized: \( n \times r \)
For SVD, “r” is measured to be the rank of the matrix, which is the minimum of the original matrix dimensions. In overall, all matrices necessity is full rank, meaning r is equal to either m or n. In this case, we can accurately rebuild the original matrix given the three decomposed matrices.

SVD has a stimulating property that permits for less than full rank of the matrices to approximate the original matrix. For the purposes of this project and LSA, we don’t want the original data back (perfect reconstruction of the original matrix), but rather we want underlying relationships in the movie data.

V. PROPOSED IMPLEMENTATION

To complete Latent Semantic Indexing on a collection of documents, this paper performs the following steps:

- Main in first, change each document in your index into a vector of word occurrences. The number of dimensions your vector occurs in is equivalent to the number of exclusive words in the whole document set. Most document vectors will have great empty patches, some will be quite full. It is optional that common words (e.g., “this”, “him”, “that”, “the”) are removed.
- Next, scale each vector so that every term imitates the frequency of its occurrence in context. *I’ll column the math for this step when I get home.* Seems he didn’t ever get home ;-
- Next, combine these column vectors into a large term-document matrix. Rows represent terms, columns represent documents.
- Execute SingularValueDecomposition on the term-document matrix. This will result in three matrices usually called U, S and V. S is of specific interest, it is a diagonal matrix of singular values for your document system.
- Recombine the terms to form the original matrix (i.e., U * S’ * V(t) = M’ where (t) signifies transpose).
- Break this reduced rank term-document matrix back into column vectors. Assistant these with their corresponding documents.
- Currently you have a Latent Semantic Index.
5.1 Indexing Process

A stemming algorithm is a process of language standardization, in which the different procedures of a word are summarized to a common form, for example,

Connection
Connections
Connective --> connect
Connected
Connecting

It is important to grow that we use stemming with the meaning of improving the concert of LSI systems. It is not an exercise in etymology or grammar. In fact from an etymological or grammatical viewpoint, a stemming algorithm is accountable to make many mistakes. In adding, stemming algorithms - at least the ones obtainable here - are appropriate to the written, not the spoken, form of the language.

For some of the world's languages, Chinese for example, the concept of stemming is not applicable, but it is certainly meaningful for the many languages of the Indo-European group. In these languages words tend to be constant at the front, and to vary at the end:

ion
ions
connect-ive
ed
ing

The adjustable part is the `ending', or `suffix'. Taking these endings off is called `suffix stripping' or `stemming', and the residual part is called the stem.

VI. RESULT

The performance evaluation in the thesis is being carried out by using standard Bug tracking of recall and precision for each word & sentence, interpolated average percentage is computed.

![Figure 1: Semantic Search Sentence Based Similarity](image)

In Figures 2 and 3, we have presented the results on evaluation of interpolated semantic search for each of the content queries in the database. Figure 3 represents the highlight of sentence for each query with respect to SVD and LUD at, where we have recorded the maximum average percentage in Figure 3.
VII. CONCLUSION

Current bug tracking systems effectively elicit all of the information needed by user. Without this material developers cannot resolve bugs in a timely detect. We analyze the same bug summary attributes in different
ways. First we analyze the bug summary with the help of LSI Indexing & SVD classification which is simple to use and efficient to other classification algorithm. Natural language processing enables us to implements a more automated and more efficient bug training process. By analyzing same BUG summary with different methods it increase the efficiency of the system. While implementing a range of improvements from these areas may be ideal, bug tracking systems may instead prefer to specialize, thus providing a rich set of choices.

REFERENCES


