A COMPARATIVE STUDY OF DOCUMENT CORRELATION TECHNIQUES FOR TRACEABILITY ANALYSIS

Anju G. Parvathy, Bintu G. Vasudevan and Rajesh Balakrishnan
SETLabs, Infosys Technologies Ltd., Bangalore

Keywords: Traceability matrix, impact analysis, document correlation, TFIDF, LSI, LDA, CTM.

Abstract: One of the important aspects of software engineering is to ensure traceability across the development lifecycle. Traceability matrix is widely used to check for completeness and to aid impact analysis. We propose that this computation of traceability can be automated by looking at the correlation between the documents. This paper describes and compares four novel approaches for traceability computation based on text similarity, term structure and inter-document correlation algorithms. These algorithms base themselves on different information retrieval techniques for establishing document correlation. Observations from our experiments are also presented. The advantages and disadvantages of each of these approaches are discussed in detail. Various scenarios where these approaches would be applicable and the future course of action are also discussed.

1 INTRODUCTION

Requirements traceability can be defined as "An explicit tracing of requirements to other requirements, models, test requirements and other traceability items such as design and user documentation". A traceability item in turn is "Any textual or model item, which needs to be explicitly traced from another textual, or model item, in order to keep track of the dependencies between them" (Spence and Probasco, 1998). A general definition of a traceability item would be thus, a "project artifact". In the life cycle of large software projects several artifacts like Requirements specification, Architecture, Design specification, Test Plans, Use Case documents etc. get created. Any change in the requirements or design would imply change in many other artifacts as well. The ability to assess the impact of a change, in any one of the artifacts, on the rest of the project artifacts and on the project execution itself is a critical aspect of software engineering.

Traceability links is one popular mechanism used to perform impact analysis. This allows a user to follow the life of a requirement both forwards and backwards, from origin through implementation (Gotel and Finkelstein, 1994). Thus the user can keep track of the requirement's development, specification, its subsequent deployment and use.

We start with a discussion on related works and proceed to present our hypotheses. Following this we explain the four different implementation schemes we experimented with. The first scheme involves Cosine Similarity (CosSim), the second employs Latent Semantic Indexing (LSI) (Deerwester et al., 1990), the third uses Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and the fourth is based on topic modeling using Correlated Topic Models (CTM) (Blei and Lafferty, 2007). We also explain how we classified the dependent artifacts into high, medium and low match categories. This is followed by a comparative analysis of the recall and precision inferred from their respective traceability matrices.

2 RELATED WORKS

Traceability is essential for many purposes, including assuring that systems conform to their requirements, that terms are defined and used consistently, and that the structures of models correspond to requirements (Alexander, 2002). A variety of software engineering tasks require tools and techniques to recover inter document traceability links, particularly the ones be-

G. Parvathy A., G. Vasudevan B. and Balakrishnan R. (2008).
A COMPARATIVE STUDY OF DOCUMENT CORRELATION TECHNIQUES FOR TRACEABILITY ANALYSIS.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - ISAS*, pages 64-69
DOI: 10.5220/0001676100640069

tween documentation and source code. These tools enable general maintenance tasks, impact analysis, program comprehension and reverse engineering for redevelopment and systematic reuse. Several such tools are currently available, some of which are discussed here. There are different types of traceability techniques like cross reference centered, document centered and structure centered (Gotel and Finkelstein, 1994). We use document correlation to build the traceability links.

Different methods to improve the overall quality of dynamic candidate link generation for requirements tracing have been recently studied. (Dekhtyar et al., 2006). A novel approach to automate traceability when one artifact is derived from the other has been illustrated(Richardson and Green, 2004). 'Qua Trace' is yet another tool to aid impact analysis that allows traces to be established, analyzed, and maintained effectively and efficiently (Knethen and Grund, 2003). Some other tools that support traceability between artifacts are (RDD-100, 2006), (RequisitePro, 2006), (DOORS, 2006), RETH (Ebner and Kaindl, 2002),CORE (Vitech Corp.), icCONCEPT RTM (Integrated Chipware), SLATE (TD Technologies, Inc.) and XTie-RT (Teledyne Brown Engineering) .

The user interface supported by most commercial tools detailed above, introduces overhead on software development process. This is pronounced by the tracing tools adding a lot of extra work on the users like Business Analyst, Architect, Designer, Developer, Tester, which is distinctly different from what their corresponding job function demands. These overheads include maintenance of the database, storing the concept definitions and relations, maintaining access rights and permissions to different kinds of users etc. Further the user is supposed to mark requirements, build the requirement hierarchy and maintain it. Moreover, it is necessary to start using the tool from project initiation as it would be virtually impossible to capture the required trace information from different project artifacts otherwise. These tools also provide limited support for impact analysis. Most of the tools do not give a quantitative relationship between two concepts, either. Further the links established by such tools are rarely based on the semantic structure of the documents. Hence some sort of semantic analysis should be incorporated in the tools to eliminate 'false positive links' established by the tools.

Some approaches for semantic analysis can be used for information retrieval for document correlation. Probabilistic Latent Semantic Analysis is a probabilistic variant of Latent Semantic Analysis which defines a proper generative model of data (Hoffman, 1999). Aspect model (Hoffmann et al., 1999) and semi discrete matrix decomposition (Kolda and O'Leary, 1998) are other models for semantic analysis. An algorithm used for classification of words into semantic classes (or concepts) is 'Unsupervised Induction of Concepts' (Lin and Pantel, 2001).

The four approaches for requirements tracing discussed in this paper reduce the burden on the user by automatically identifying concepts and trace links between these concepts. Moreover, approaches based on LSI, LDA and CTM aid in semantic analysis.

3 AUTOMATING TRACEABILITY

In this section, we look at the details of the approaches towards automating requirements tracing. They are based on two key hypotheses: The first, is usage of common terminology. We believe that the application-domain knowledge that programmers process when writing the code is often captured by the mnemonics for identifiers; therefore, the analysis of these mnemonics can help associate high-level concepts (eg:use case) with program concepts (eg: class design specification) and vice-versa. The second, hypothesis is the usage of connected words as in class/design specification documents. A connected word contains two or more words in mixed case. The beginning of a new token within a word is marked by an upper case character. Often prefixes are also attached to these words. The prefixes which begin with a lower case character are neglected as they do not contribute to the meaning of the word. Appropriate rules are set to tokenize the connected word depending on the document type in use (eg: of a connected word is 'getCostParameter'; 'get' is the prefix and the tokens are 'Cost' and 'Parameter'). Accordingly, if a particular document has the words 'cost' and 'parameter' and another has the connected word 'getCostParameter' then the two documents are related.

Formally, a project consists of a set of artifacts like Business Process Model, Requirements Specification, Class/Design Specification, Test Case document etc. Each Artifact could contain multiple concepts having different concept types. A concept type can be a requirement, use case, test case, class design or some other relevant topic. A set of "Concept Extraction Algorithms" can be used to specify how to extract a particular concept type from an artifact. A simple CEAL for extracting requirement concept type from a requirement document could be to look for some key word like "Requirement" in the different headings in the document. In some extreme cases,

we may have to apply a combination of text segmentation and text abstraction algorithms to extract concepts from artifacts (Goldin and Berry, 1997). Our approaches propose to derive the relationships, between concepts, through different "Relationship Extraction Algorithms" (REAL) that specify how to derive relationships between two different types of concepts. For eg:, if a project has a use case artifact and a class design artifact and the template used for documenting use cases have a section called "Related Classes"; an REAL to extract relationships can look for the items under the "Related Classes" section. Another REAL could be based on text similarity between use case description and class design specifications. If the "Related Classes" section is well maintained, the first algorithm is the more efficient one.

The approaches may not help in deriving all the concepts and relations in the documents. However, even if an automated mechanism based on the above approach is able to derive 60-70 % of the concepts and relations, we believe that there would be an overall increase in productivity through out the lifecycle of a software project. The approaches themselves are not tied to any specific set of project artifacts or concept types or any specific document format. Further, we can use any of the four approaches in any stage of software development lifecycle for inter artifact tracing.

We used sample documents of "Use Case" (UC)s and related "Class Specification" (CS)s as input corpora for all the four approaches. The outputs are quantified relationship matrices and classified traceability matrices. The input corpora are preprocessed and freed from 'stop words' which are very common words, that will adversely effect the relationship matrix computation. Every new word is also checked for being a connected word. The four approaches based on cosine similarity, LSI, LDA and CTM are discussed subsequently.

Cosine Similarity. Here the documents are preprocessed and represented in the vector space using $TFIDF$ of terms, as weights. The popular text similarity algorithm of cosine measure is applied to every pair of document vectors to get the relationship index RI_{ij}, which quantifies the relationship between them. The greater the cosine value the higher is the correlation. Further, we classify the dependencies based on their degree of match into three categories high, medium and low matches respectively. For classification the *RI*s with CSs are first sorted in descending order for each UC. Then the average values of highest three *RI*s are used as threshold values for classification. Thus corresponding to each UC, we have three classes of CSs, one or more of which could be empty, as per the degree of correlation between the documents.

Latent Semantic Indexing. Documents are preprocessed as before and subsequently represented in vector space. The term by document matrix is then subject to singular value decomposition for LSI ((Deerwester et al., 1990) and (Dumais, 1991)). Choosing optimal value for rank k for dimensionality reduction produces a reduced term by document matrix. Different values of k result in different links being recovered. The *RI* here is the cosine of angle between a pair of document vectors (of documents to be correlated). This *RI*s are used for classification of CSs as in the earlier approach.

Latent Dirichlet Allocation. LDA is a general probabilistic model for collections of discrete data such as text corpora (Blei et al., 2003). The input corpus size is normalized. The maximum document size amongst the documents in the corpora is obtained. Then every document with size less than 50% of the maximum document size is appended with copies of itself until the size exceeds the maximum document size. This normalization is done to ensure uniform length of documents so that LDA captures importance words from smaller documents and quantizes their relationships.

The documents are preprocessed as in the earlier cases. The vector form of every document(d) is determined by the unique words(w) and their frequencies(f) in it. The j^{th} document vector is : $d_j = \langle n \rangle \langle w_i : f \rangle ...$, where n is the total number of unique words in d_j and $i = 1, 2, ...n$.

We assign the number of topics for which LDA pools the terms to be equal to the total number of dependent documents in the entire corpora. LDA processes the document vectors with predefined settings and produces the bag of words and log beta file which is a matrix of word to topic probabilities. We define a text similarity algorithm where RI_{ij} is computed as:

$$RI_{ij} = \sum_{k=1}^{N} p_{ki} \times f_{ki} \times f_{kj} \times \beta_k$$

where, N is the total number of unique words in LDA bag of words, p_{ki} is the weight with respect to maximum number of tokens in connected words, f_{ki} and f_{kj} are the frequencies of $word_k$ in $document_i$ and $document_j$ respectively and β_k is the sum of beta probabilities of $word_k$ across all topics as given by LDA. The classification of the CSs is done as in the previous cases, the only difference being a different form of *RI*.

Correlated Topic Model. CTM is a fast variational inference procedure for carrying out approximate inference that can be used for semantic analysis (Blei

and Lafferty, 2007). The procedure for traceability is exactly the same as the one which involves LDA, except that the bag of words is obtained using CTM techniques.

4 EXPERIMENTAL RESULTS

Based on the above mentioned approaches the experiments were carried on sets of two distinct corpora, each one containing documents belonging to a particular concept type in the artifact - specifically 22 UCs[1] and 21 CSs[1]. Similar experiments were conducted on a project of 26 UCs and 235 related CSs for actions dealing with waste water management, recall and precision of which are discussed later.

4.1 Quantified Relationship Matrices

In this section, we provide the quantified relationship matrices (of dimension 22 X 21) for four different approaches mentioned.

Table 1 shows a portion of the matrix of normalized *RI* computed using cosine similarity. In the example, RI_{14} is greater than RI_{11}, RI_{12}, RI_{13}, RI_{14}, RI_{15}, RI_{16} and RI_{17}. This means that the highest correlation is between CS_4 and UC_1 , amongst the seven retrieved relations for UC_1 as in the table.

Table 1: Relationship matrix of *RI* for CosSim.

UC	CS_1	CS_2	CS_3	CS_4	CS_5	CS_6	CS_7
1	0.0	0.0	0.01	0.40	0.03	0.02	0.20
2	0.0	0.02	0.0	0.01	0.02	0.003	0.26
3	0.0	0.0	0.0	0.01	0.004	0.0	0.21
4	0.03	0.009	0.02	0.02	0.01	0.001	0.29
5	0.002	0.002	0.002	0.31	0.04	0.03	0.05

Table 2 shows the normalized *RI* computed after applying LSI with an optimal latent parameter $k = 10$. The higher the value of k, the lesser is the reduction in dimension. The highest possible value of k is equal to the total number of documents in the corpus in which case the results are the same as in the case of simple cosine similarity. In the table, the RI_{27} is the largest amongst all *RIs* for UC_2. Hence UC_2 is highly related to CS_7.

Table 2: Relationship matrix of *RI* for LSI.

UC	CS_1	CS_2	CS_3	CS_4	CS_5	CS_6	CS_7
1	0.0	0.02	0.004	0.46	0.36	0.01	0.20
2	0.007	0.02	0.007	0.09	0.17	0.003	0.17
3	0.004	0.02	0.005	0.04	0.13	0.0	0.15
4	0.005	0.02	0.008	0.10	0.18	0.009	0.17
5	0.001	0.01	0.003	0.53	0.30	0.03	0.09

Table 3 presents a part of the normalized matrix generated using the bag of words from LDA, with number of latent topics equal to 21 (the number of CSs). Here again RI_{14} of UC_1 with CS_4 is the highest in the list of *RIs* for UC_1 and quantifies a true positive 'traceability link'. The *RIs* of UC_2, UC_3 and UC_4 with CSs do not help in recovering true relationships as many dominant words from these UCs and CSs fail to be present in the LDA generated bag of words which consequently weakens the traceability process. As for UC_5, it was found that there was no CS in the CS corpora that was directly dependent on UC_5. Hence the *RIs* represent false positive relations for UC_5.

Table 3: Relationship matrix of RI for LDA.

UC	CS_1	CS_2	CS_3	CS_4	CS_5	CS_6	CS_7
1	0.02	0.02	0.01	0.16	0.02	0.02	0.03
2	0.01	0.01	0.01	0.01	0.01	0.004	0.01
3	0.005	0.006	0.006	0.006	0.005	0.002	0.006
4	0.04	0.05	0.03	0.04	0.02	0.02	0.03
5	0.01	0.01	0.007	0.11	0.01	0.01	0.01

The normalised matrix computed using CTM generated bag of words is as shown in Table (4). The number of latent topics is once again chosen to be 21. The highest *RIs* denote strongest relationships of UCs with CSs for UC_1, UC_2 and UC_3.

Table 4: Relationship matrix of *RI* for CTM.

UC	CS_1	CS_2	CS_3	CS_4	CS_5	CS_6	CS_7
1	0.01	0.01	0.006	0.16	0.02	0.01	0.02
2	0.0	0.0	0.004	0.0	0.006	0.0	0.005
3	0.0	0.0	0.002	0.0	0.003	0.0	0.003
4	0.007	0.01	0.007	0.01	0.01	0.008	0.01
5	0.005	0.003	0.002	0.09	0.01	0.004	0.01

4.2 Classified Traceability Matrices

The quantified relationship matrices generated previously are used to rank the CSs for every UC, in descending order of *RI*. After applying threshold measures on the *RI*s,the CSs that are dependent on the

[1]The results elaborated were generated from experiments on artifacts (UCs and CSs) pertaining to a shopping portal project. UC_1 elaborates the steps involved in making replacement costs available to regions, UC_2 elucidates the actions related to posting prices, UC_3 on publishing these posted prices, UC_4 on getting these posted prices and UC_5 explains the action sequence for entering parameters for quotes. CS_4 is a CS on cost parameter control within the shopping portal which also deals with quote prices, replacement costs and fixed price deals . CS_{10} sequences the process involved in searching cost parameters and replacement costs and CS_7 describes posted price control.

UCs are classified into the high, medium or low match categories to generate classified traceability matrices.

Table 5 A shows the matrices computed using the measure of cosine similarity and the LSI technique. For cosine similarity method, it is observed that for UC_1 the high and medium matches agree with experts' trace. The relationships of UC_2, UC_3 and UC_4 with CS_7 fall in the medium, low and medium match categories respectively. But the experts' trace suggest that these are the strongest relations for these UCs. Relations for UC_5 can be ignored as it does not have significantly matching CSs as per experts' trace.The matches indicated by the traceability matrix for LSI approach are valid for UC_1, UC_2 and UC_3. Relations for UC_4 and UC_5 can be ignored owing to the reasons cited earlier. The classified traceability matrices generated using LDA and CTM are shown in Table 5 B. The trace measure indicated by the LDA approach validates the assertions by experts' trace for UC_1 only. The fallacy of the other links retrieved by LDA is due to inappropriate representation in bag of words.As for the CTM approach, UC_1, UC_2 and UC_3 are observed to have approvable matches.

Table 5: Classified traceability matrices for CosSim, LSI, LDA and CTM approaches.

UC A	CS					
	CosSim			LSI		
	High	Medium	Low	High	Medium	Low
1	4,10	14,17	7,21,13,9,12	4,10	12,5	11,13,14,9,17
2		7	13,11,9,16,2			5,7,13,9,19
3			7,11,13,17,16			7,13,5,19,9
4		7	9,13,11,20,16			13,12,9,11,5
5	14	4,10,17	21,12,9,7,13	4,10,14	17	21,12,11,5,13
B	LDA			CTM		
	High	Medium	Low	High	Medium	Low
1	10	4,14,17,21	19,12,15,11,7	10,4	14,17,21	19,12,15,5,11
2			15,3,2,12,11			15,5,7,3,12
3			10,19,2,3,7			7,5,15,11,3
4			19,2,11,4,10			15,19,10,4,11
5		10,4,14,17	21,19,12,15,11		10,4,14,17	21,12,15,19,5

4.3 Analysis of Experimental Results

The four different approaches were experimented on two case studies. The precision and recall shown in Table 6 were calculated without applying a threshold on *RI* values.

Table 6: The precision and recall table.

Approach	Precision%		Recall%	
	Case I	Case II	Case I	Case II
CosSim	73	72	94	90
LSI	55	48	71	63
LDA	50	40	65	50
CTM	50	40	65	50

For a given pair of documents, the *RI* computation using cosine measure involves all unique words in both the documents. But it provides a relatively small amount of reduction in description length and reveals little in the way of inter or intra document statistical structure. LSI does "noise reduction", precluding the term combinations which are less frequently occurring in the given document collection, from the LSI subspace used to calculate *RI*. The approaches that use LDA and CTM for computing the *RI* are confined to the bag of words that they generate after semantic analysis. The recalls offered by the later two methods are poorer than the cosine similarity and LSI based approaches because of the inability of dominant words from certain documents to figure in the bag of words. This could be the reason why the strength of traceability links are different when the different approaches are used.

In general, CTM approach scores over LDA approach in the fact that the words collected under one bag in CTM, is not confined to a particular document. So the inter document relationship is delivered by the bag of words as well. The LDA approach was found to extract words in a more document specific manner and hence the words with low frequency but of high importance in some documents didn't figure in the bag. However the two approaches yielded the same precision and recall in the experiments conducted.

The best traceability scheme as suggested by the result of experiments is thus 'document correlation using cosine similarity considering connected words'. However a great deal of its accuracy is attributed to the emphasis on connected words. When the same procedure was carried out ignoring such words, the recall was very poor and the result very erratic.

4.4 Advantages and Disadvantages

One of the most notable advantages is that they partially automate the task of concept identification and relationship extraction reducing the burden on the user for building and maintaining trace information. Further all of these approaches can be adopted during any phase of the project's lifecycle. Also, as the relationships are quantified, we can differentiate between strongly related and weakly related documents. This aids impact analysis and helps find the minimal set of design specifications that cover a given set of requirements. Moreover, the emphasis given to connected words by all four approaches is extremely advantageous for use in many technical domains of projects. Also, the techniques are programming language and paradigm independent, thus offering more flexibility and automation capability. Further, the relationship extraction using LDA and CTM bag of words greatly reduce the description length of a document and also

reveal inter or intra document statistical structure.

However these approaches are not without their share of inadequacies. Their success depend upon the acceptance of standard templates for information capture by the project teams. Further, concepts and relations, at least sometimes, will have to be extracted using text segmentation or text similarity algorithms. Most of them work on large volumes of text. Hence success of our approaches will depend on whether these algorithms can be tuned to work with small amounts of text. Also, there are chances of the system coming up with wrong relations. The onus of maintaining the correctness of trace data still is on the user.

To improve the effectiveness of the traceability, we will consider the following improvements. A glossary or a Thesaurus which aids in resolving usage of similar words/terms can be devised. Ontology can be used to capture domain specific concepts. We can provide the user with a mechanism to control keywords importance and document granularity . Further improved text normalization methods can be adopted so that LDA and CTM bag of words uniformly contain dominant words from all documents in the corpora.

5 CONCLUSIONS

Document correlation is an important step towards establishing traceability. However inter-artifact traceability is often ignored due to the large overhead added by the current tracing mechanisms. Even partial automation of the trace building and maintenance would help the user to seriously consider this aspect through out the lifecycle of the project. The four different approaches for tracing discussed in this paper, can at least partially automate this process. The approaches can also be incorporated into the project at an advanced stage in project's lifecycle. To improve the efficiency of traceability 'tokenization of connected words' is done by all four approaches . Such words are extremely important for correlating technical documents like design specifications and actual codes with other documents. The result of experiments we conducted suggest that the best approach for traceability amongst the ones discussed is 'document correlation using cosine similarity considering connected words'. However the LSI, LDA and CTM based approaches emphasize on extracting semantic information from the text corpora. We also aspire to explore alternate methods to compute index of document similarity to improve the recall and precision.

REFERENCES

Alexander, I. (2002). Towards automatic traceability in industrial practice. In *Proc. of Workshop on Traceability*, pages 26–31.

Blei, D. M. and Lafferty, J. D. (2007). A correlated topic model of science. *Appl. Stat.*, 1(1):17–35.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. of MLR3*, pages 993–1022.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. (1990). Indexing by latent semantic analysis. *J. of ASIS*, 41(6):391–407.

Dekhtyar, A., Hayes, J. H., and Sundaram, S. K. (2006). Advancing candidate link generation for requirements tracing: The study of methods. In *IEEE Trans. on SE*, volume 32, pages 4–19.

DOORS (2006). Telelogic, http://www.telelogic.com.

Dumais, S. (1991). Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments, and Computers*, 23(2):229–236.

Ebner, G. and Kaindl, H. (2002). Tracing all around in reengineering. *IEEE Software*, 19(3):70–77.

Goldin, L. and Berry, D. M. (1997). Abstfinder, a prototype natural language text abstraction finder for use in requirements elicitation. *ASE*, 4(4):375–412.

Gotel, O. and Finkelstein, A. (1994). An analysis of the requirements traceability problem. In *Proc. of the IEEE Int'l. Conf. on Req. Engg.*, pages 94–101.

Hoffman, T. (1999). Probabilistic latent semantic analysis. In *Proc. of UAI*, pages 289–296.

Hoffmann, T., Puzicha, J., and Jordan, M. I. (1999). Learning from dyadic data. *ANIPS 11*.

Knethen, A. V. and Grund, M. (2003). Quatrace: A tool environment for(semi-) automatic impact analysis based on traces. In *Proc. of the Int'l. Conf. on Software Maintenance*, pages 246–255.

Kolda, T. G. and O'Leary, D. P. (1998). A semi - discrete matrix decomposition for latent semantic indexing in information retrieval. *ACM Trans. on Info. Sys.*, 16(4):322–346.

Lin, D. and Pantel, P. (2001). Induction of semantic classes from natural language text. In *Proc. of the seventh Int'l. Conf. on KDDM*, pages 317–322, California.

RDD-100 (2006). Holagent corporation, http://www.holagent.com/products/product1.html.

RequisitePro, R. (2006). Rational software, http://www.rational.com/products/reqpro/index.jsp.

Richardson, J. and Green, J. (2004). Automating traceability for generated software artifacts. In *Proc. of the 19th IEEE Int'l. Conf. on ASE*, pages 24–33.

Spence, I. and Probasco, L. (1998). Traceability strategies for managing requirements with use cases. *W. Paper*.