Configuring Latent Semantic Indexing for Requirements Tracing

Sebastian Eder, Henning Femmer, Benedikt Hauptmann, Maximilian Junker
Technische Universität München, Germany

***Abstract*—Latent Semantic Indexing (LSI) is an accepted technique for information retrieval that is used in requirements tracing to recover links between artifacts, e.g., between requirements documents and test cases. However, configuring LSI is difficult, because the number of possible configurations is huge. The configuration of LSI, which depends on the underlying dataset, greatly influences the accuracy of the results. Therefore, one of the key challenges for applying LSI is finding an appropriate configuration. Evaluating results for each configuration is time consuming, and therefore, automatically determining an appropriate configuration for LSI improves the applicability of LSI based methods.**

We propose a fully automated technique to determine appropriate configurations for LSI to recover links between requirements artifacts. We evaluate our technique on six sets of requirements artifacts from industry and academia and show that the configurations selected by our approach yield results that are almost as accurate as results from configurations based on a ground truth like known links or expert knowledge. Our approach improves the applicability of LSI in industry and academia, as researchers and practitioners do not need to determine appropriate configurations manually or provide a ground truth.

I. Introduction

When change requests have to be implemented for a software system, not only the software itself has to be changed, but also its documentation. It is often unclear which parts of a software system have to be changed and which other artifacts are impacted by the change. Furthermore, if one documentation artefact is changed, e.g., a use case, other artifacts also have to be changed, for example test cases. In both cases, we face the problem to decide which artifacts have to be changed in order to keep a consistent state of the software itself and the documentation as a whole.

In the fields of requirements tracing and trace link recovery, there are several methods of finding artifacts that are semantically connected to each other. Many of these approaches rely on Latent Semantic Indexing (LSI) [1] to retrieve links between artifacts that are semantically connected, e.g. [2]–[4]. However, LSI has to be configured differently for every artifact corpus [5].

Problem: There is a huge number of possible configurations for LSI [6] heavily influencing the resulting links and their accuracy [2], [3], [5], [7]–[10]. Existing approaches determine the configuration of LSI by using the configuration that reflects the links best that have to be created by system experts manually first [2]–[4], [11], [12]. Therefore, experts have to invest efforts to gain initial results, and may introduce incorrect links [13], which limits the applicability of LSI in requirements tracing.

Contribution: We propose a fully automated technique that estimates configurations for LSI. Our technique only relies on heuristic metrics that are calculated on the similarities between documents computed by LSI. In contrast to existing techniques, it does *not* rely on system experts' knowledge, existing known links between documents, or a ground truth, to recover semantic links between artifacts written in natural language. This yields the advantage, that experts do not have to invest efforts to gain just first results.

We evaluate our technique on six sets of requirements artifacts from industry and academia. We show that our approach determines configurations for LSI that yield, compared to the best possible configurations and to configurations chosen with a ground truth, accurate results for different kinds of artifacts.

Outline: The remainder of the work is structured as follows: First, we give an overview of related work and then introduce the most important terms and definitions. We then outline the procedure how we derived the proposed approach. We describe our pre-study, the approach and its validation. Afterwards, we describe topics for future work and summarize our work.

II. Related Work

Automated Requirements Tracing: Falessi et al. present seven principles that aim on improving the validity of studies that compare techniques for requirements tracing based on natural language processing [14]. We adhere, as far as possible, to these principles, however, we do not compare different techniques for requiremenents tracing (as we are considering just LSI) but present an approach for finding configurations for LSI that yield accurate results. Furthermore, Falessi et al. present a case study that compares different techniques to requirements tracing. As they state that LSI, not considering certain configurations, yields well results, we are motivated to also use this technique.

Requirements Tracing using LSI: A common approach to requirements tracing is to use LSI for generating links between semantically connected artifacts. Existing approaches use rules of thumb and best practices based on knowledge about correct links between artifacts, as also shown in Section VII, to select initial configurations of LSI, e.g., [2]–[4], [12], [15]. These works aim at the accuracy of recovered trace links, or at improving their accuracy. As optimal parameters for LSI vary

for every artifact corpus [5], we propose to select parameters more carefully in a systematic way, and to focus on finding an initial configuration for LSI to recover trace links.

There are several approaches for recovering links between semantically similar requirements artifacts. Some approaches use expert feedback on some generated links to adjust the whole set of links [3], [15]. However, expert feedback might worsen the results [13] and forces practitioners to invest efforts, which limits the applicability in practice. Therefore we do not consider expert feedback to adjust the results produced by LSI, but develop a fully automated approach for the estimation of appropriate configurations of LSI for a given artifact corpus without using a ground truth, like expert knowledge or known links.

Parameter Estimation for LSI: Kontostathis gives insights in factors that influence the performance of LSI itself [16]. However, the work focuses on understanding the inner mechanisms of LSI and does not consider the accuracy of the results. In contrast, the accuracy is in the center of our work, since it improves the applicability of LSI in requirements tracing. Ali et al. investigate the impacts of the inputs to tracing approaches when recovering links between source code and requirements artifacts [7]. They concentrate on experts' knowledge and on properties of the given artifact corpus. In contrast, we focus only on the configuration of LSI to produce accurate results independently of the actual artifact corpus or experts' feedback, because often, both are not changeable nor available. Binkley at el. give an approach to estimate parameters for Latent Dirichlet Allocation (LDA) [17], but not for LSI. We focus on LSI.

III. Background and Terms

Latent Semantic Indexing: LSI calculates the similarity between artifacts contained in an artifact corpus based on the content of the contained artifacts. Similarity is usually expressed by a value between -1 and 1, where a greater value means the compared artifacts are more similar[1]. LSI identifies words belonging to a common concept (e.g., 'car' and 'automobile'), enabling it to deal with synonyms to a certain extent.

LSI starts with creating a term-document matrix ($terms \times artifacts$) containing entries for each word in each artifact. The entries are calculated by a global weighting function w_g and a local weighting function w_l for each word, determining a value depending on the occurences of a term in the containing document (local) and on the ocurrences of a term in the whole artifact corpus (global). On this matrix, singular value decomposition is performed and the result is truncated to a smaller matrix, given the number of desired dimensions k (which can be interpreted as the number of concepts). This results in a reduced matrix where words are replaced by their concepts. This matrix represents every artifact as a vector in the space of concepts. The similarity of artifacts is calculated by comparing their vectors, e.g., cosine similarity[2].

We apply LSI to calculate similarity values between source artifacts and target artifacts. For example, if we want to know which use cases are impacted by change requests, the source artifacts represent the change requests and the target artifacts represent the use cases. Therefore, we generate similarity values between all change requests and all use cases. Afterwards, the target artifacts (use cases) are sorted by their similarity value to the source artifact (change requests, in descending order), resulting in a ranking of all target artifacts by similarity to the source artifact.

Configuration options of LSI: A configuration of LSI consists of three items: the local weighting function w_l, the global weighting function w_g, and the number of dimensions k. Table I shows common options for each item, and supplemantary variables. We use these options in the remainder of the work. All combinations of the options for w_l, w_g, and k are applicable, leading to $3 \cdot 3 \cdot (N-4)$ combinations.

TABLE I
Configuration options of LSI considered

Variables	
df_i	Number of artifacts containing ith term
tf_{ij}	Number of occurences of ith term in jth artifact
gf_i	Number of occurences of ith term in all artifacts
p_{ij}	tf_{ij}/gf_i
N	Total number of artifacts
Options for the Local Weighting Function w_l	
Term Freq.	$tf = tf_{ij}$
Log Term Freq.	$ltf = log(tf_{ij} + 1)$
Binary	$bin = 1$
Options for the Global Weighting Function w_g	
Entropy	$H = 1 - \sum_j \frac{p_{ij} \ log \ p_{ij}}{log \ n}$
Inverse Doc. Freq.	$idf = log \frac{n}{1+df_i}$
Binary	$bin = 1$
Options for the Number of Dimensions k	
Range from 5 to N, considering all possibilities (step width is 1)	

Quality Measurement: To measure the quality of a configuration, we calculate the average precision (AP) [18] on the rankings for every source artifact and then take the mean over all these rankings, which results in the mean average precision (*MAP*) [6]. A *MAP* of 1 means for a ranking, that all relevant artifacts are ranked to the top of the list, without irrelevant artifacts in between[3]. We use *MAP* only for evaluating our approach (see Section VII), and to develop our approach (see Section V). The approach itself does not rely on *MAP*, since for this, it would need a ground truth.

[1] Can be scaled to the range between 0 and 1.

[2] We use only cosine similarity, since our pre-study (see Section V) showed that it outperforms other distance measures by large.

[3] *MAP* implicitely measures recall as well, since a greater number of irrelevant artifacts between relevant artifacts reduces the *MAP*.

Quality of Configurations: The best possible *MAP* among all configurations for one dataset is denoted as MAP_B, and the average over all configurations as MAP_A.The *MAP* produced by a selected configuration C is denoted as MAP_C.

As an acceptence criterion, we consider a configuration C as *appropriate*, if $MAP_C > MAP_B - 0.1$. This means, compared to the best configuration, precision@rank [10] drops by less than 0.1 in average over all possible ranks and over all source artifacts with an appropriate configuration. Therefore, we expect configurations that are *appropriate* to produce rankings that reflect actual semantic links almost as good as the best configuration does.

Heuristic metrics: The proposed approach aims at finding appropriate configurations of LSI by not using a ground truth or expert knowledge. However, it uses heuristic metrics that are calculated solely on the ranking produced by LSI with a certain configuration, and *not* on the ground truth. *MAP* cannot be such a heuristic metric, since it demands a ground truth, like a set of known links between artifacts, to be calculated.

IV. Study Overview

The goal of our study is to develop an approach to select appropriate configurations for LSI without using a ground truth in terms of expert knowledge or known links. We developed the approach in a structured way and in three steps, as illustrated in Figure 1. The next sections are organized along this structure.

In our first step, we conduct a pre study to build hypotheses about which heuristic metrics are suitable to determine appropriate configurations. We do this based on our experience gathered in earlier work [19], [20], and on the dataset MR0. Additionally, we test the hypotheses for validity to determine whether the heuristic metrics are suitable to determine appropriate configurations. This step is detailed in Section V.

The second step is developing a fully automated approach that is based on the hypotheses and heuristic metrics from the first step by operationalizing them, as explained in Section VI.

The last step is validating our approach by applying it to all datasets available. In this step, we determine whether the proposed approach is suitable for finding appropriate configurations for LSI without using a ground truth or expert knowledge. The approach will be described in detail in Section VI, and the validation well be described in Section VII.

We use the ground truth for the datasets only to test our hyptheses from the first step, to develop our approach and to validate the approach. The ground truth is not used for applying the approach.

V. Pre-Study

We summarize the first step of our study in this section, the pre-study: First, we develop our heuristic metrics and build hypotheses based on them. Second we test the hypotheses. The goal of the pre-study is to identify reliable heuristic ranking metrics (that do not depend on a ground truth) we can base our apporach on.

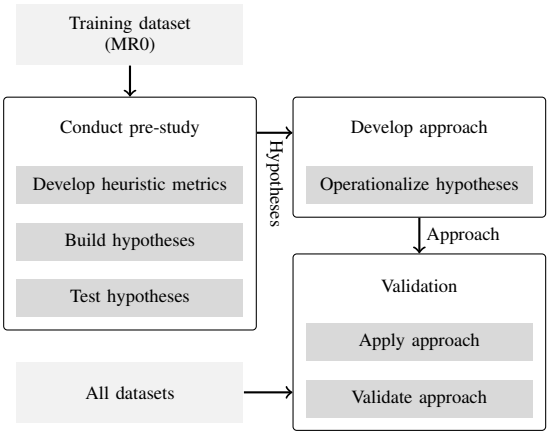


Fig. 1. Schematic overview of the studies. Light boxes are input artifacts, while dark boxes are steps in the approach. Arrows indicate data flow.

A. Heuristic Ranking Metrics

Our approach relies on two heuristic metrics that are calculated based on the rankings a configuration yields. We developed these metrics in a pre-study with the dataset MR0 (see Section VII) for which all correct links were documented by system experts. We use knowledge about correct links only for developing and evaluating our approach, whereas the approach itself does *not* rely on it.

We determined two heuristic metrics to select configurations from our experience: *pos*, and *range*. The following paragraph explains these metrics and the rationale behind them.

Metric *pos*: The position of the largest distance between the similarity values of two consecutive[4] artifacts in the ranking. *Rationale:* Often, only few target artifacts are actually linked to the source artifact. The lower *pos* is, the less artifacts are considered to be linked by LSI following the approach of Zhao et al. [21] that considers artifacts ranked above *pos* to be actually linked to the source artifact. Thus, we expect rankings to be better, if *pos* is lower.

Metric *range*: The difference of the highest and lowest similarity value in a ranking. *Rationale:* With a higher *range*, the ranking gets more expressive, since artifacts with higher similarity are ranked higher with more confidence, and low ranked artifacts have a more distinctive similarity.

To calculate *pos* and *range* for a configuration C, we take the mean of these metrics over the rankings C produced for all source artifacts.

B. Hypotheses

With our two heuristic metrics, we constitue two hypotheses. Since we measure the accuracy of the ranking produced by one configuration by its *MAP*, both hypotheses relate our heuristic metrics to *MAP*.

$\mathbf{H_{pos}}$: Rankings with low *pos*, exhibit a high *MAP*. So we expect configurations that produce rankings with a low *pos* to produce more accurate results.

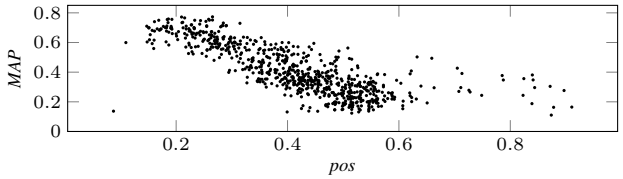
[4]Rankings are sorted by similarity value in descending order (Section III).

$\mathbf{H_{range}}$: Rankings with high *range*, exhibit a high *MAP*. This hypothesis states that we expect configurations that yield rankings with a high *range* to produce more accurate results.

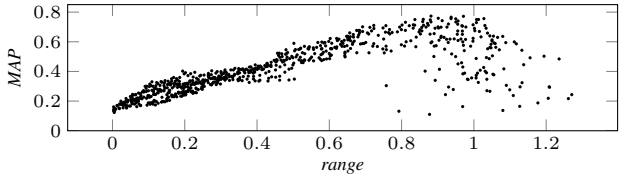
C. Test of Hypotheses

Figure 2 visualizes the correlation between *pos*, *range*, and *MAP* for the dataset MR0, where one dot is one configuration: Configurations with a high *MAP* exhibit a low *pos* and a high *range*. The visual impression is confirmed by the correlation of the two heuristic metrics to *MAP*: The Pearson correlation coefficient between *pos* and *MAP* is -0.84, consituting a strong correlation, and between *range* and *MAP* it is 0.87, also constituting a strong correlation. Both with a p-value that is greater than 95%.

Due to these correlations, we confirm our hypotheses. Therefore, we use these metrics to determine appropriate configurations. However, picking configurations with the highest *pos* or the highest *range* is not sufficient, since there are outliers, which do not yield optimal results, but exhibit a low *MAP*, as illustrated in Figure 2.



(a) Correlation of *pos* and *MAP*



(b) Correlation of *range* and *MAP*

Fig. 2. Correlations of the metrics *pos* and *range* with *MAP* for the study object MR0. Every point represents one configuration.

VI. Approach

Due to the outliers shown in the last section, we develop a more sophisticated approach to determine proper configurations for LSI.

The input to the first step of our approach are the sets of source and target artifacts (the dataset). The output of the approach is one configuration. For all other steps, the inputs are the outputs of the previous step. Note that we select different configurations for every dataset. The approach is illustrated in Figure 3. Note that, as illustrated, the proposed approach does *not* use any kind of ground truth as input, but only the source and target artefacts.

① **Perform LSI** to compare every source artifact to every target artifact for all possible configurations.
Output: For every configuration: Rankings for each source artifact to all target artifacts sorted by similarity.

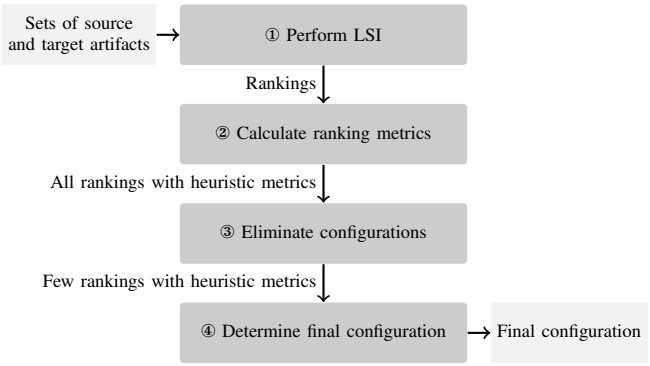


Fig. 3. Schematic illustration of the approach. Light boxes are input/output artifacts, while dark boxes are steps in the approach. Arrows indicate data flow between the steps.

② **Calculate ranking metrics** for the rankings produced by the previous step. For every configuration, we calculate *pos* and *range*.
Output: Ranking metrics for all configurations.

③ **Eliminate configurations** with a mean value of *pos* higher than the median or a mean value of *range* lower than the median, because we want a configuration to satisfy both criteria: low *pos* (only a small number of target artifacts to be linked to the source artifact), and high *range* (a high difference in similarity between highly and lowly ranked target artifacts). We repeat this step until the number of kept configurations is lower than ten[5]. The rationale behind this step is not to select appropriate configurations, but to eliminate the bad ones. We do this incrementally not to loose either configurations with a low *pos* or a high *range*.
Output: Few configurations.

④ **Determine the final configuration** by taking the configuration with the median of *pos* among the remaining configurations, because we do not want to select outliers in terms of *pos* or *range*. The reason for this becomes apparent in Figure 2: The best configurations in terms of *MAP* exhibit a high *range* and a low *pos*, but there are configurations, which are worse, that also fulfil these conditions.

VII. Validating Study

We conduct a case study to evaluate our approach by answering the following research questions. The goal of the validating study is to check whether the proposed approach selects appropriate configurations for LSI. We used knowledge about correct links between artifacts just for designing and evaluating our approach, but not as an input to it.

RQ1: Does the algorithm find appropriate configurations? By answering this question, we validate the ability of the proposed approach to find configurations that produce accurate trace links measured against the best possible configuration.

RQ2: How does the automated approach compare to approaches used in literature? Other approaches in literature

[5]This is an arbitrary choice, but yielded the best results for MR0.

use knowledge about the semantic similarity between the artifacts in their datasets to gain appropriate configurations. We compare our approach that does not take any knowledge about a (partial) set of correct links between artifacts, to the configurations used by other researchers.

A. Study Objects

We use six datasets with varying artifact types to evaluate our approach. Three datasets, MODIS, CM-1, and EasyClinic[6] were already used by other researchers to evaluate their approaches, and can therefore be used for answering RQ1 and RQ2. The other three datasets, MR0, MR1, and MR2, originate from the reinsurance company Munich Re and are confidential. Therefore, they can only be used for answering RQ1, since no other researchers could validate their approaches on them. The characeristics of the datasets are shown in Table II.

B. Study Execution

We execute the study along our approach. We use an own implementation of LSI in Java, which is optimized for running LSI with different configurations and was already used in prior work [19], [20]. We calculate ranking measures in R[7]. Also the elimination and selection of the final configuration is done in R. Running the analyses took 11 minutes on a computing server with 16 processing cores (2.0 GHz) and 64 GB of RAM.

C. Results

RQ1: Figure 4 shows the *MAP* resulting from the configuration C selected by our approach (MAP_C), the *MAP* resulting from the best possible configuration (MAP_B), and the average *MAP* over all configurations (MAP_A), which shows that it is not trivial to find an appropriate configuration. The difference from MAP_B to MAP_C lies below 0.1 in every case, and therefore, we consider all found configurations as *appropriate*, as explained in Section III.

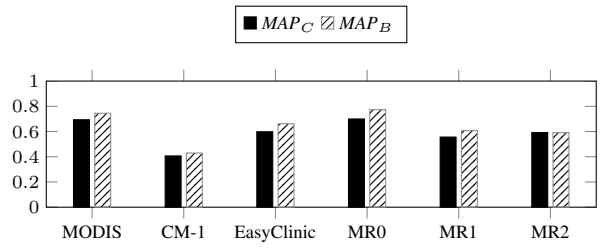


Fig. 4. *MAP* of the selected, best and average configuration for each dataset

Discussion: For all datasets we considered, the approach finds configurations that produce rankings with a *MAP* that is close to MAP_B. In the case of MR2, we find the optimal configuration. The difference of MAP_B to MAP_C lies below 0.1 in every case. This means, compared to the best configuration, precision@rank [10] drops by less than 0.1 in average over all possible ranks and over all source artifacts with the configuration selected by our approach. Therefore, the selected configuration produces rankings that reflect actual semantic links almost as accurate as the best configuration.

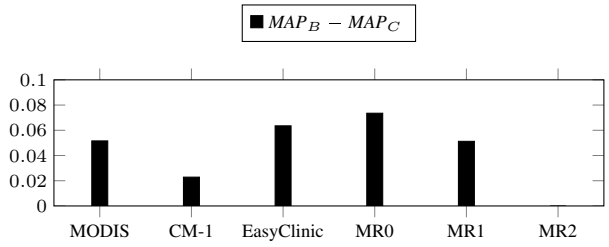


Fig. 5. Difference of *MAP* of the selected, and best configuration for each dataset. The higher the bars, the greater is the distance of the best possible configuration and the selected one.

Figure 6 shows the distribution of the *MAP* for the different study objects. The distributions show that finding a configuration that is that close to the best configuration is not trivial, since there are much more configurations with a *MAP* worse than the *MAP* of the selected configurations. The chance of selecting appropriate configurations according to our definition in Section III by random lies between 1% (MR1: 15 configurations out of 1422 are appropriate) and 23% (CM-1: 932 out of 4050 configurations are appropriate).

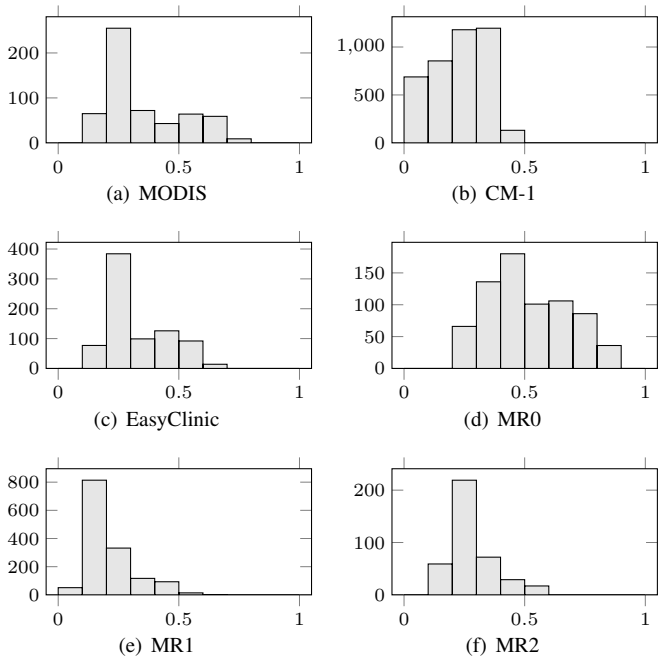


Fig. 6. Distribution of *MAP* over all configurations for the study objects

Thus, we conclude that the proposed approach finds configurations yielding accurate results, given that LSI is able to produce rankings with a desired quality.

RQ2: Table III shows the configurations from previous work [3], [4], [15] that were applied to our datasets and the

[6]These datasets, and others, can be acquired for study reproduction via CoEST at http://www.coest.org/index.php/resources/dat-sets

[7]http://www.r-project.org/

TABLE II
STUDY OBJECTS

Name	Source artifacts	Target artifacts	# possible Configurations	Source
MODIS	19 high-level requirements	49 low-level requirements	567	Industry (NASA)
CM-1	235 high-level requirements	220 low-level requirements	4,050	Industry (NASA)
EasyClinic	30 use cases	63 test cases	792	Academia
MR0	24 use cases	60 test cases	711	Industry (Munich Re)
MR1	135 defect reports	28 use cases	1,422	Industry (Munich Re)
MR2	28 change requests	21 use cases	396	Industry (Munich Re)

TABLE III
CONFIGURATIONS FOR ALL STUDY OBJETCS

	Configurations of other contributions					Selected		
Dataset	Paper	w_l	w_g	k (*C1*)	k (*C2*)	w_l	w_g	k
MODIS	[3], [15]	*tf*	*idf*	10	19	*ltf*	*bin*	7
CM-1	[3], [15]	*tf*	*idf*	100	200	*tf*	*idf*	66
EasyClinic	[4]	*ltf*	*H*	$0.1 \cdot N$	$0.2 \cdot N$	*tf*	*idf*	12

configurations selected by our approach. All three contributions propose two configurations *C1*, and *C2*. However, the configurations per contribution only vary in the number of dimensions k. In Figure 7, we illustrate the comparison of the configuration C selected by our approach, yielding MAP_C to the *MAP* of configurations used in previous work: MAP_{C1} and MAP_{C2}. In the case of CM-1 and EasyClinic, MAP_C is as high as the highest *MAP* achieved by the other configurations, and for MODIS, MAP_C lies between the two configurations proposed by the works we compare to.

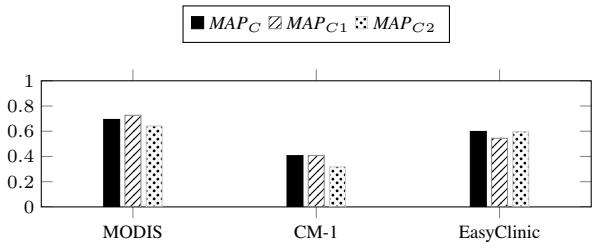


Fig. 7. *MAP* of the configuration selected by our approach (MAP_C) and the configurations proposed by other papers (MAP_{C1} and MAP_{C2})

Discussion: In all cases, our approach selected configurations that yield rankings with a *MAP* close to the configurations proposed by prior work. This means that our approach performs almost as well as approaches using knowledge about existing links between artifacts. We therefore conclude, that our approach is suitable for finding appropriate configurations. And in contrast to prior work, our approach does not rely on knowledge about existing links between artifacts. This shows the improvement of LSI's applicability by our approach.

D. General Discussion

The results show, that the proposed approach finds appropriate configurations among a great number of possible configurations (496 – 4050), where the number of appropriate configurations is small, without necessary knowledge about correct links between artifacts for several datasets. Therefore, it is suitable for estimating parameters for LSI.

The approach is computationally intensive, because LSI has to be performed on the datasets with all possible configurations. Analyzing the largest dataset (CM-1) took 7 minutes on a computing server with 16 processing cores (2.0 GHz) and 64 GB of RAM.

E. Threats to Validity

Even though we conducted the study on six datasets, we can still not generalize from the results. We adressed this threat by selecting study objects from industry containing heterogeneous artifacts from different companies.

We used manually developed lists of the correct links between artifacts that might contain errors. Three of the datasets (MODIS, CM-1, and EasyClinic) were used by other researchers in a plethora of studies and were examined for accuracy. Therefore, we consider this threat as minor for these datasets. For the datasets provided by Munich Re (MR0, MR1, and MR2), we performed manual inspections to mitigate this threat. As we could not find false or missing links, we consider this threat as minor for these datasets.

Measuring the *MAP* of the rankings produced by the configurations selected by our approach might not be suitable for all scenarios, since *MAP* captures whole rankings rather than just the first artifacts and thus implicates that the whole rankings are presented. Other approaches cut the rankings by thresholds of similarity values or after a fixed number of artifacts [14]. They measure, consequently, F-Measure to show the validity of the produced rankings. However, we examined the correlation between precision, recall, and *MAP*, and observed that *MAP* correlates strongly (in a statistical sense) with F-Measure.

VIII. FUTURE WORK

Since the approach is computationally intensive, we plan to employ more intelligent search strategies, e.g., genetic algorithms [6], to find configurations that match the given criteria in terms of the proposed metrics.

Furthermore, we only considered links between homogeneous sets of artifacts written in natural language. Therefore, we plan to evaluate our approach between more heterogeneous sets of artifacts like models or diagrams and source code.

As LSI is not the only approach for recovering semantic links between artifacts [22], we plan to apply similar heuristics for configurations of other tracing techniques. Examples are Latent Dirichlet Allocation (to compare to Binkley et al. [17]), the Jensen-Shannon Divergence, or the Vector Space Model.

Another research direction is, incorporating expert feedback as proposed in prior contributions, e.g., [3], [15]. We expect, as the initial rankings are produced automatically by our approach, the applicability of these approaches to improve.

IX. Summary and Conclusion

Changing requirements of a software system lead to changes not only to the software itself, but also to its documentation and tests, e.g., if a use case is adapted, we also have to update test cases. Therefore, we have to decide which artifacts have to be changed to keep a *consistent state* of the software itself and the documentation corpus.

Latent Semantic Indexing (LSI) is a common technique for requirements tracing, to recover links between artifacts, e.g., between requirements documents and test cases. However, configuring LSI is difficult, because the number of possible configurations is huge. The configuration of LSI, which depends on the underlying dataset, greatly influences the accuracy of the results. Therefore, one of the key challenges in applying LSI-based methods is finding an appropriate configuration producing accurate results.

We presented an approach to find configurations for LSI used in requirements tracing automatically. In contrast to other approaches, our approach does not rely on any kind of ground truth like expert knowledge or existing links between artifacts. The approach only uses two heuristic measures, *pos* and *range* that are calculated solely on the results LSI produces. Therefore, the only input to our approach are the documents for which traceability links have to be recovered.

We furthermore showed in a case study considering six objects from industry and academia that our approach finds appropriate configurations among hundreds or thousands of possible configurations. The found configurations produce rankings with similar accuracy (in terms of Mean Average Precision), as configurations chosen manually by other researchers with knowledge about existing links that had to be created manually beforehand.

The results we gained in our study indicate that our approach is suitable for automatically selecting appropriate configurations for LSI: The configurations selected are, in terms of accuracy, close to configurations selected with a ground truth. This improves the applicability of LSI in requirements tracing in research and practice, since no ground truth has to be established for determining an appropriate configuration.

Acknowledgment

The authors would like to thank Daniela Steidl, Veronika Bauer, Jonas Eckhardt, Daniel Méndez Fernández, and Andreas Vogelsang for their helpful comments on this work.

References

[1] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *JASIST*, vol. 41, no. 6, 1990.

[2] M. Lormans and A. van Deursen, "Can LSI help reconstructing requirements traceability in design and test?" in *CSMR*, 2006.

[3] S. K. Sundaram, J. H. Hayes, and A. Dekhtyar, "Baselines in requirements tracing," *SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, 2005.

[4] A. D. Lucia, F. Fasano, R. Oliveto, and G. Tortora, "Recovering traceability links in software artifact management systems using information retrieval methods," *TOSEM*, vol. 16, no. 4, 2007.

[5] R. B. Bradford, "An empirical study of required dimensionality for large-scale latent semantic indexing applications," in *CIKM*, 2008.

[6] S. Lohar, S. Amornborvornwong, A. Zisman, and J. Cleland-Huang, "Improving trace accuracy through data-driven configuration and composition of tracing features," in *ESEC/FSE*, 2013.

[7] N. Ali, Y.-G. Guéhéneuc, and G. Antoniol, "Factors impacting the inputs of traceability recovery approaches," in *Software and Systems Traceability*, J. Cleland-Huang, O. Gotel, and A. Zisman, Eds. Springer London, 2012.

[8] A. Garron and A. Kontostathis, "Applying latent semantic indexing on the trec 2010 legal dataset," in *TREC*, 2010.

[9] G. Bavota, A. De Lucia, R. Oliveto, A. Panichella, F. Ricci, and G. Tortora, "The role of artefact corpus in LSI-based traceability recovery," in *TEFSE*, 2013.

[10] A. Abadi, M. Nisenson, and Y. Simionovici, "A traceability technique for specifications," in *ICPC*, 2008.

[11] A. Kontostathis, "Essential dimensions of latent semantic indexing (LSI)," in *HICSS*, 2007.

[12] S. Eder, B. Hauptmann, M. Junker, E. Juergens, R. Vaas, and K.-H. Prommer, "Did we test our changes? assessing alignment between tests and development in practice," in *AST*, 2013.

[13] D. Cuddeback, A. Dekhtyar, and J. Hayes, "Automated requirements traceability: The study of human analysts," in *RE*, 2010.

[14] D. Falessi, G. Cantone, and G. Canfora, "Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques," *IEEE Trans. Softw. Eng.*, vol. 39, no. 1, 2013.

[15] J. Hayes, A. Dekhtyar, and S. Sundaram, "Advancing candidate link generation for requirements tracing: the study of methods," *IEEE Trans. Softw. Eng.*, vol. 32, no. 1, 2006.

[16] A. Kontostathis and W. M. Pottenger, "A framework for understanding latent semantic indexing (LSI) performance," *Information Processing & Management*, vol. 42, no. 1, 2006, formal Methods for Information Retrieval.

[17] D. Binkley, D. Heinz, D. J. Lawrie, and J. Overfelt, "Understanding LDA in source code analysis," in *ICPC*, 2014.

[18] S. Robertson, "A new interpretation of average precision," in *SIGIR*, 2008.

[19] S. Eder, B. Hauptmann, M. Junker, R. Vaas, and K.-H. Prommer, "Selecting manual regression test cases automatically using trace link recovery and change coverage," in *AST*, 2014.

[20] S. Eder, H. Femmer, B. Hauptmann, and M. Junker, "Which features do my users (not) use?" in *ICSME*, 2014.

[21] W. Zhao, L. Zhang, Y. Liu, J. Sun, and F. Yang, "Sniafl: Towards a static noninteractive approach to feature location," *TOSEM*, vol. 15, no. 2, 2006.

[22] M. Borg and P. Runeson, "IR in software traceability: From a bird's eye view," in *ESEM*, 2013.