Leveraging Intermediate Artifacts to Improve Automated Trace Link Retrieval

Alberto D. Rodriguez
Computer Science and Software Eng.
California Polytechnic State University
San Luis Obispo, CA, USA
arodr243@calpoly.edu

Jane Cleland-Huang
Computer Science and Eng.
University of Notre Dame
Notre Dame, IN, USA
JaneHuang@nd.edu

Davide Falessi
Civil Eng. and Computer Science
University of Rome Tor Vergata
Rome, Italy
falessi@ing.uniroma2.it

***Abstract*—Software traceability establishes a network of connections between diverse artifacts such as requirements, design, and code. However, given the cost and effort of creating and maintaining trace links manually, researchers have proposed automated approaches using information retrieval techniques. Current approaches focus almost entirely upon generating links between pairs of artifacts and have not leveraged the broader network of interconnected artifacts. In this paper we investigate the use of intermediate artifacts to enhance the accuracy of the generated trace links – focusing on paths consisting of source, target, and intermediate artifacts. We propose and evaluate combinations of techniques for computing semantic similarity, scaling scores across multiple paths, and aggregating results from multiple paths. We report results from five projects, including one large industrial project. We find that leveraging intermediate artifacts improves the accuracy of end-to-end trace retrieval across all datasets and accuracy metrics. After further analysis, we discover that leveraging intermediate artifacts is only helpful when a project's artifacts share a common vocabulary, which tends to occur in refinement and decomposition hierarchies of artifacts. Given our hybrid approach that integrates both direct and transitive links, we observed little to no loss of accuracy when intermediate artifacts lacked a shared vocabulary with source or target artifacts.**

***Index Terms*—software traceability, links, traceability network**

I. Introduction

Software traceability links capture associations between diverse artifacts such as requirements, design, code, and test cases [1], [2], and provide support for many different development activities, such as safety-assurance, impact analysis, and software reuse. However, manually creating and maintaining trace links requires non-trivial effort, and often does not provide immediate benefits to the team members who need to create the links [1]. As a result, developers and other project stakeholders, often postpone the task of creating or maintaining links, resulting in incomplete, inaccurate, and even conflicting trace links [1], [3], [4], [5]. Researchers have addressed this problem through the use of automated tracing techniques based on information retrieval, machine learning, repository mining [6], [7], [8], and more recently deep-learning approaches [9], [10] which have increased the accuracy achievable by automated approaches.

Project artifacts are organized according to an artifact schema, often referred to as a Traceability Information Model (TIM). The trace links created between individual artifacts form an artifact network, and individual trace queries often include multiple artifacts, creating the notion of a *transitive trace* in which multiple atomic traces are connected in a sequence, where the target artifact for one atomic trace becomes the source artifact for the next trace [11], [12]. For example, given the TIM of the TrainController TC dataset depicted in Fig. 1, we can trace from System Requirements (SR) to SubSystem Requirements (SSR) directly or intermediately through System Designs (SD). The network of artifacts introduces the opportunity to leverage intermediate artifacts in order to improve the accuracy of an automatically generated end-to-end trace link. In this paper we explore the benefits of leveraging intermediate artifacts whilst generating trace links.

Fig. 1: TrainController's traceability information model (TIM) showing artifacts and traceability paths.

Consider the example provided in Figure 2 which includes a system requirement (SR), a system design (SD), and a subsystem (SSR) requirement artifact taken from the TC System, and shows two possible traceability paths. The *direct* trace path goes from SSR $\rightarrow$ SR, while the indirect path goes from SSR $\rightarrow$ SD $\rightarrow$ SR. The dashed boxes provide semantic explanations for these associations. For example, the SR includes the term "Highway Wayside Segment" while the SD includes the associated acronym 'WIU'. Tracing techniques that include project glossaries are easily able to leverage this information to help establish a link between SD and SR. On the other hand, the semantic distance from SSR to SR is much greater with fewer direct connections. By leveraging the intermediate SD component, we see that 'WCP' is a component of the 'WIU' which, as previously stated, represents the Highway Wayside Segment described in the SR.

The question we explore in this paper is whether leveraging intermediate artifacts can improve the accuracy of trace links generated using different automated techniques. Nishiwaka et al., [12] previously explored this question and proposed a technique which they referred to as "Connecting Links Method" (CLM). They compared the results from a transitive trace from $A \rightarrow B \rightarrow C$, against a direct trace from $A \rightarrow C$ and reported improvements for only a small subset of the trace

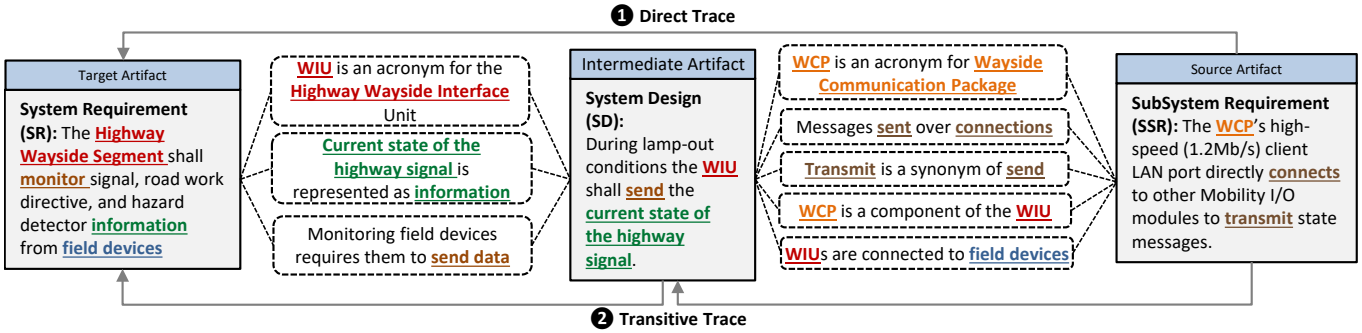


Fig. 2: An illustrative example inspired by our Train Controller system illustrates the concept of a direct versus a transitive trace. The implicit semantic connections between the artifacts are depicted in the dashed nodes.

links. However, they evaluated their approach using only one dataset based on only one technique. We therefore replicate their experiment on multiple datasets, and expand it to include diverse techniques for generating links and aggregating results. We address the following research question:

Can the accuracy of trace links generated between end-to-end artifacts be improved by incorporating intermediate artifacts in the trace path, and under what conditions do intermediate artifacts improve this accuracy?

Results from our experiments show that effectiveness of using intermediate artifacts to generate trace links is impacted by the terminology used across the artifacts and the presence of a strong hierarchical relationship between the artifacts.

The remainder of the paper is laid out as follows. Section II presents the techniques we used to incorporate intermediate artifacts in end-to-end trace retrieval. Section III describes the series of experiments that we conducted, while Sections IV and V report and analyze the results. Finally in Sections VI to VIII we describe threats to validity, related work, and present conclusions and ideas for future work.

II. Leveraging Intermediate Artifacts

This section describes how we leverage intermediate artifacts in automated trace retrieval techniques. There are three technique families in our study. The first, *Direct*, uses one of two algebraic models to compute textual similarities between source and target artifacts, subsequently, this family does not leverage intermediate artifacts. The second family, *Transitive*, was first proposed by Nishiwaka et al. [12] as the 'Connecting Links Method' (CLM) and only uses intermediate artifacts. We evaluate 12 transitive techniques using combinations of two algebraic models, two scaling methods, and three path aggregation functions. We refer to this approach as *transitive traceability*. Finally, the third family, *Hybrid*, aggregates results from a *Direct* and *Transitive* technique using one of three aggregation functions. Figure 3 illustrates the differences between these three techniques.

A. Computing Artifact Similarity

Several different information retrieval techniques have been proposed and evaluated for generating links between two

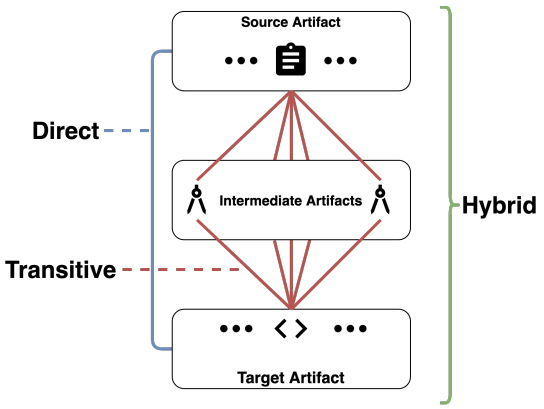


Fig. 3: Artifact relations used by each technique to calculate similarity scores between a source and target artifact pair.

artifacts. Of these, the Vector Space Model (VSM) has consistently performed well in previous studies even though it fails to take semantic similarity into consideration. We also include the Latent Semantic Indexing (LSI) approach for comparison purposes, as this considers underlying semantic concepts.

• *VSM* computes term similarity between two artifacts and has been applied to many traceability tasks [13], [14], [15], [16]. Despite its simplicity it has been shown to consistently outperform other information retrieval techniques [17], [10]. In VSM, terms are represented as an indexed linear vector, while individual artifacts are depicted as weighted vectors, with weights commonly assigned using term frequency-inverse document frequency (tf-idf). In tf-idf, the importance of each term is based on its occurrence and distribution across the text corpus. More precisely, let A_S and A_T represent the collection of source and target artifacts respectively, then each artifact $a_i \in A_S \cup A_T$ is represented by the terms $\{t_1 ... t_n\}$ it contains, and transformed into a numeric format $a_i = \{w_1, w_2, \ldots, w_n\}$ where w_n indicates the tf_idf score for t_i. Artifact similarity is then measured by computing the distance between vector representations, often using cosine similarity between the source and target vectors as follows:

$$Similarity(a_i, a_j) = \frac{a_i^T \cdot a_j}{\sqrt{a_i^T \cdot a_i}\sqrt{a_j^T \cdot a_j}} \quad (1)$$

• *Topic modeling approaches* have also been used for trace link generation [18], [17], [19], [16] as they are capable of discovering hidden semantic structures as abstract concepts and representing each artifact as a distribution over these concepts. The most common techniques are Latent Dirichlet Allocation (LDA), Latent Semantic Indexing (LSI), and Probabilistic Latent Semantic Indexing (PLSI). LSI represents each artifact a_i as a vector of word counts c_n such that each word is represented as $a_i = \{c_1, c_2, ..., c_n\}$ and the artifact corpus A is represented as a matrix $A = \{a_1, a_2, ..., a_m\}$ where m refers to the total number of all artifacts in A. LSI applies matrix decomposition, e.g Singular Value Decomposition (SVD) in order to learn the latent topics [15], [20], [21]. In a probabilistic variant of LSI, known as [22], SVD is replaced by a probabilistic model of latent topics. Finally, LDA is a Bayesian version of PLSI in which dirichlet priors are introduced for the topic distribution. Given a topic distribution vector of both source and target artifacts, the similarity, or affinity between two artifacts is estimated by using either Cosine similarity or Hellinger distance [23] to quantify the similarity between two probability distributions. In this paper, we have selected LSI as a representative topic modeling approach.

Both VSM and LSI produce a similarity score for each pair of artifacts. Scores range from 0 to 1, with higher numbers representing greater likelihood of a link. Scores are stored in a **similarity matrix** where a score at indices (i,j) stores the similarity between $a_i \in A_S$ and $a_j \in A_T$.

B. Aggregating Results across Trace Paths

The vast majority of research in trace link generation has focused on generating links directly between two artifacts without considering alternate paths, even though case studies of industrial projects have shown that multiple, potentially redundant paths often exist between artifacts [4], [24]. In this work we focus on trace queries involving three different artifact types (e.g., requirements $\rightarrow$ design $\rightarrow$ code). We define a *direct link* as a **link** between a *source artifact* and a *target artifact*, and an *indirect link* (otherwise referred to as a **transitive trace link**) as a link comprising multiple *atomic trace links* strung in sequence, such that a *target artifact* for one *atomic trace link* becomes the *source artifact* for the next *atomic trace link* [11], [12]. For the remainder of this discussion we assume three artifact types A_S (source), A_T (target), and A_I (intermediate).

All direct links are generated from A_S to A_T using a tracing algorithm (e.g., VSM and LSI) to compute artifact similarity and to produce a traceability matrix as previously described.

The goal of using a transitive approach is to generate links from A_S to A_T by leveraging an intermediate artifact type A_I. We first generate a trace matrix of direct links between A_S and A_I and refer to this as TM_{SI}, and then a second matrix of direct links from A_I to A_T, referred to as TM_{IT}. To generate transitive links, we need to perform two additional tasks to *normalize* and *aggregate* TM_{SI} and TM_{IT} in order to produce a trace matrix between A_S and A_T. There are several possible approaches for normalization and aggregation of links.

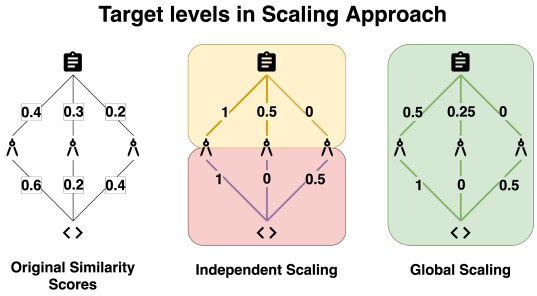


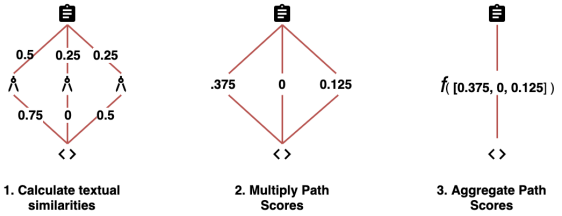
Fig. 4: The groups of similarity scores scaled to the range [0, 1] within the *Independent* and *Global* scaling approaches

• *Link Score normalization:* Tracing algorithms, such as VSM and LSI, are sensitive to the characteristics of the artifacts – typically impacted by attributes such as document length and vocabulary size [25]. In practice, they often produce higher scores on more textually rich documents than on sparser ones, which creates a problem when aggregating results from two or more trace matrices (e.g. TM_{SI} and TM_{IT}) where the matrices with higher scores could overly influence the final trace results. We therefore evaluated the following scaling approaches as summarized in Figure 4:

- *Independent Scaling*: Each matrix is scaled **separately**, normalizing its scores to span the range [0,1]. The scope of each score is noted in Figure 4 by the orange and red rectangles.
- *Global Scaling*: Matrices are scaled **conjointly**, such that each score is scaled within the global minimum and maximum score of all matrices, and so the global range spans [0,1]. Figure 4 illustrates this with the encompassing green rectangle highlighting that the range of all scores in considered when scaling.

• *Link Aggregation:* Given two matrices TM_{SI} and TM_{IT} we could imagine multiple paths from source artifact $a_j \in A_S$ to target $t_k \in A_T$ via different intermediate artifacts (see red paths in Figure 3). Each of these individual, intermediate paths includes two distinct links from a_j to i_l and from i_l to t_k, each with their own similarity score (or probability score). We compute the product of these two scores to produce the final strength of a single intermediate path. Furthermore, given N intermediate artifacts, our approach generates N intermediate paths from a_j to t_k each associated with its own similarity score. In Figure 5 we see an example with three intermediate artifacts between a source and target artifact pair. In the first intermediate path we see that the similarity score between the source and intermediate artifact is *0.5*; similarly, the score between the intermediate and target artifact is *0.75*. Therefore, the source and target artifacts have a similarity score of *0.375* through the first intermediate artifact as depicted in the second step of Figure 5. Finally, the final score describing the similarity between source and target artifacts will be the aggregate of the scores for each intermediate path (e.g. *0.375*,

Fig. 5: The process of creating single transitive relation score from textual similarities.



0, *0.125*).

We aggregate the relation scores using the following functions in order to produce a single score representing the likelihood of a link from $a_j \rightarrow t_k$:

- *Max*: The maximum relation score is used with the rationale that a single strong connection is sufficient to establish a trace link.
- *Sum*: The sum of all transitive relation scores with the rationale that a set of related intermediate nodes provides evidence for a link. While there is a risk that a large number of intermediate artifacts with low similarities could incorrectly inflate the score, we include this method for replication purposes as it was the original method used by Nishikawa et al., [12].
- *PCA*: A weighted sum of intermediate similarity scores is used. The weights represent the percentage of variance explained by relation scores through some intermediate artifact calculated using principal component analysis (PCA). The underlying assumption here is that intermediate artifacts whose relations scores explain the variance among all transitive relation scores deserves more weight because they are more able to discern relatedness between artifact pairs. This method has previously been used to integrate orthogonal information in retrieval techniques [19].

C. Exploring a Hybrid Approach

We also evaluate a hybrid approach in which results from the direct and transitive approaches are aggregated. The idea behind this approach is to augment benefits of the direct links with supporting data provided by the transitive links. This is also illustrated in Figure 3. We evaluated three different ways of aggregating direct and transitive results as follows:

- *Max*: The maximum value between the direct and multi-hop similarity scores.
- *Sum*: Direct and transitive scores are summed and scaled.
- *PCA*: A weighted sum of the direct and transitive relation scores. The weights represent the proportion of variance explained by each technique in the aggregate of all relation scores calculated using PCA. The underlying assumption being that techniques that better discern between link and non-links deserve a greater weight. [26].

III. Experimental Design

We evaluated our approach using five different datasets. Our experiments require datasets with at least three different types of artifacts and trace links provided along at least two distinct trace paths. Many publicly available traceability datasets include two artifact types or are extremely small. We favored the use of larger datasets (e.g., TrainController and Dronology) as these are more representative of targeted industrial projects, but due to lack of available datasets we also used three smaller ones available at COEST.org. We established the additional criterion that intermediate artifacts must represent at least 5% of the total artifacts. We did not use OSS as in prior traceability experiments (e.g., reported by Rath et al., [27]) because of their lack of sufficient intermediate traces. For example, SEAM2, had 27 source artifacts, 63,207 intermediate artifacts, 21,069 target artifacts; but, did not include any intermediate trace links. Similarly, PIG included three artifact types, but one of the paths only included actual links for about 0.01% of the artifact pairs.

The selected datasets are shown in Table I. Four of these datasets are available as community resources, while one (TrainController) was provided by our industrial collaborators under a non-disclosure agreement. Table I provides a brief description of each dataset, the three artifact types used in our study for each dataset, the trace link paths, the number of actual links, candidate links, and subsequent percent of true links out of all candidate links for that path.

A. Dataset Preparation

All artifacts were processed as follows. Method names such as *receiverManager.log()* were split into their constituent parts, non-alpha numeric characters were removed, and CamelCase or Snake_case phrases were split apart. Finally, all characters were converted to lower case, stop words were removed, and remaining words were stemmed to their morphological roots using the Porter Stemmer (e.g. *receiv manag log*).

In most datasets links were only provided between a subset of artifact types. We therefore used *transitive* inferencing, to establish ground truth for the end-to-end tracing path or its missing segment – depending upon which subset of links were provided. For example, as shown in Table I, Dronology provided transitive links from FRs $\rightarrow$ Design, and from Design $\rightarrow$ to Java Classes. By using the Connecting Links Method (CLM) described by Nishikawa et al. [12], we were able to establish ground truth for the direct links from FRs $\rightarrow$ Java Classes. Table I differentiates between manually created links (bolded diamonds and triangles), and inferred ones (hollowed diamonds and triangles). Diamonds represent the direct trace path, while triangles represent either the path from source to intermediate artifact (i.e., TM_{SI} ▲) or from the intermediate artifact to the target (i.e., TM_{IT} ▼).

B. Trace Link Generation

To construct the trace matrices needed for our evaluation, we first used VSM to generate trace links for trace paths with artifact indices $0 \rightarrow 1$, $1 \rightarrow 2$, and $0 \rightarrow 2$ for each dataset and stored the subsequent results in a trace matrix. This produced three initial trace matrices for each of our five projects in which the $0 \rightarrow 2$ matrix represented the *direct* links, while the

TABLE I: Dataset descriptions, artifacts, and defined trace links.

Dataset	Description	Refs	Artifacts	Paths		Links	Candidates	Percent
Dronology	A system for managing and coordinating multiple semi-autonomous Unmanned Aerial Vehicles for emergency response missions.	[28][29]	0: FRs (30)	$0 \rightarrow 2$	◊	2,985	25,025	11.9%
			1: Design (20)	$0 \rightarrow 1$	▲	1,749	5,445	32.1%
			2: Java Classes (455)	$1 \rightarrow 2$	▼	7,100	45,045	15.8%
TrainControl	An industrial system for the Onboard Unit of a positive train control (PTC) system.	Hidden for DBR	0: SRS (218)	$0 \rightarrow 2$	◊	26,857	127,094	21.1%
			1: SDD (519)	$0 \rightarrow 1$	▲	31,303	113,142	27.7%
			2: SSRS (583)	$1 \rightarrow 2$	▼	104,976	302,577	34.7%
EasyClinic	A small, student-created dataset in both English and Italian. The English version was used for this project.	[30]	0: Use Cases (30)	$0 \rightarrow 2$	♦	373	1,410	26.5%
			1: IDs (20)	$0 \rightarrow 1$	▲	210	600	35.0%
			2: Class Descriptions (47)	$1 \rightarrow 2$	▼	254	940	27.0%
EBT	Small dataset created to support event-based traceability using a publish-subscribe model.	[30]	0: Requirements (41)	$0 \rightarrow 2$	♦	1,086	2,050	53.0%
			1: Test Cases (25)	$0 \rightarrow 1$	▲	740	1,025	72.2%
			2: Source Code (50)	$1 \rightarrow 2$	▽	652	1,250	52.2%
WARC	A tool suite for handling files in the WARC format including command line tools, server plug-ins, and documentation for file handling.	[30]	0: NFRs (21)	$0 \rightarrow 2$	◊	45	882	5.1%
			1: SRs (89)	$0 \rightarrow 1$	▲	90	1,869	4.8%
			2: FRs (42)	$1 \rightarrow 2$	▼	192	3,738	5.1%

Legend: FR=Functional Requirement, NFR=Non-Functional Requirement, SR=Software level requirement, ID=Interaction Diagrams.
Trace paths provided by dataset: ♦=Direct, ▲=Higher Level, ▼ = Lower Level. Inferred paths are unfilled (e.g., ◊, △, ▽)

other two matrices comprised the building blocks needed to generate the transitive links. We then applied each combination of previously described techniques for normalizing link scores and aggregating links. This produced six additional trace matrices for each project, representing the *transitive* matrix for each combination of scaling scores (independent, global) and link aggregation methods (Max, Sum, PCA) applied to trace matrices for trace paths $0 \rightarrow 1$ and $1 \rightarrow 2$. Finally, each unique combination of direct and transitive matrices was combined using the link aggregation methods (Max, Sum, and PCA) to produce a total of 72 *hybrid* matrices. Regardless of whether direct, transitive, or hybrid techniques had been used, the final result was the end-to-end trace matrix (TM_{ST}), representing the $0 \rightarrow 2$ path depicted by a black diamond for each dataset in Table I. The entire process was then repeated using LSI in place of VSM.

C. Accuracy Metrics

Accuracy was evaluated by comparing the generated TM_{ST} against its corresponding answer set. Candidate links within TM_{ST} were ranked in descending order, and the following metrics, in alignment with standard benchmarking guidelines for traceability experiments [31], were computed.

- *MAP*: the mean average precision (AP) assigns a non-proportional higher weight to correct links ranked at the top of the result list rather than the bottom [31].
- *AUC*: area under ROC Curve, measures how well a technique distinguishes between linked and non-linked artifacts. It assigns the same weight to each correct link [31].
- *LAG*: the number of non-links that a human analyst would have to review in a ranked list of candidate traces before finding all actual links. It assigns a non-proportionally higher negative weight to a correct link ranked at the bottom of the result list [31][32]. Note, a higher LAG score means a lower performing technique. It is convenient to adjust LAG so that greater scores mean better performances as well as having it fit within the range between 0 and 1, like the other metrics. For this reason we apply a transformation to LAG to be able to visualize and compare metrics. This transformation scales scores to fit between 0 and 1 and inverts $(1 - score)$ so that higher scores mean more accurate techniques. LAG was normalized across each project, using all generated scores to establish the scaling factor. We refer to this metric as *LagNormInverted* through the remainder of this study.

The best techniques are identified by random sampling (without replacement) 75% of the artifacts of each artifact level (e.g. requirements, designs, code) into a training set and the remaining into a test set. Then, all techniques are evaluated over 20 independent training splits by taking the mean metric score (after standardization) over 20 iterations. The best performing techniques are determined via the sum of metrics scores (e.g. MAP, AUC, LagNormInverted).

Finally, after having identified the best techniques we tested whether a technique could be considered more accurate than another technique by using the non-parametric Wilcoxon test applied to our three accuracy metrics over the entire dataset. We selected this test because our trace query data does not follow a normal distribution, as noted when performing Shapiro–Wilk tests [33]. Therefore, our approach is compliant to the suggestion to avoid using ScottKnottESD in case of non normal distributions [34]. We set alpha at 0.01; we have chosen this conservative value due to the high number of performed tests.

IV. Results

We report accuracy of all direct and transitive techniques over the entire dataset in Table II. The direct family includes two techniques, the transitive family 12, and the hybrid family 72. Given so many combinations of hybrid techniques, we report only the best and worst cases using the test-train splits defined in Section III-C to select these techniques. The best techniques vary by dataset and are reported in Table IV. The best direct, transitive, and hybrid techniques are identified by the training splits and evaluated on the testing splits.

As reported in Figure 6, the best transitive technique outperformed the best direct techniques in all dataset's except WARC's MAP. **The best hybrid technique outperformed**

TABLE II: Performance of direct and transitive techniques in each dataset over all artifacts. LAG scores are represented by *LagNormInverted* metrics defined in Section III-C. A score of 0 means it performed the worst among all techniques and 1 the best. Best and worst hybrid techniques are selected via a test-train split defined in Section III-C.

Approach	Technique Variant								Project													
		Model		Scaling		Aggregation			Drone			TC			EC			EBT			WARC		
	ID	VSM	LSI	Ind	Glob	Sum	PCA	Max	MAP	AUC	LAG	MAP	AUC	LAG	MAP	AUC	LAG	MAP	AUC	LAG	MAP	AUC	LAG
Direct	D1	•							0.341	0.613	0.237	0.503	0.545	0.358	0.705	0.758	0.779	0.755	0.685	0.729	0.674	0.835	0.624
	D2		•						0.334	0.594	0.000	0.503	0.536	0.264	0.686	0.750	0.719	0.755	0.655	0.577	0.634	0.800	0.327
Trans	T1	•		•		•			0.384	0.677	0.940	0.514	0.662	1.000	0.642	0.731	0.588	0.853	0.733	1.000	0.621	0.885	0.975
	T2	•		•			•		0.362	0.652	0.653	0.448	0.597	0.621	0.475	0.667	0.148	0.835	0.691	0.774	0.563	0.852	0.725
	T3	•		•				•	0.376	0.671	0.876	0.536	0.615	0.723	0.781	0.790	1.000	0.810	0.707	0.86	0.617	0.889	1.000
	T4	•			•	•			0.384	0.677	0.940	0.514	0.662	1.000	0.632	0.726	0.554	0.853	0.733	1.000	0.621	0.885	0.975
	T5	•			•		•		0.362	0.652	0.653	0.448	0.597	0.621	0.464	0.662	0.116	0.835	0.691	0.774	0.563	0.852	0.725
	T6	•			•			•	0.376	0.671	0.876	0.536	0.615	0.723	0.780	0.790	0.996	0.810	0.707	0.86	0.617	0.889	1.000
	T7		•	•		•			0.420	0.670	0.860	0.484	0.65	0.932	0.578	0.700	0.375	0.861	0.704	0.846	0.387	0.758	0.009
	T8		•	•			•		0.357	0.608	0.166	0.342	0.501	0.056	0.433	0.646	0.004	0.828	0.582	0.182	0.585	0.787	0.229
	T9		•	•				•	0.362	0.662	0.772	0.497	0.575	0.489	0.767	0.778	0.917	0.791	0.686	0.746	0.625	0.867	0.837
	T10		•		•	•			0.424	0.682	1.000	0.480	0.655	0.961	0.575	0.699	0.370	0.861	0.675	0.686	0.387	0.757	0.000
	T11		•		•		•		0.355	0.615	0.244	0.340	0.492	0.000	0.432	0.645	0.000	0.831	0.548	0.000	0.585	0.785	0.214
	T12		•		•			•	0.365	0.663	0.781	0.509	0.586	0.557	0.766	0.778	0.916	0.791	0.667	0.641	0.625	0.869	0.848
Hybrid	Best Performing Hybrid Technique (H1)								0.399	0.681	1.000	0.563	0.634	1.000	0.770	0.797	0.954	0.838	0.739	1.000	0.749	0.894	1.000
	Worst Performing Hybrid Technique (H2)								0.342	0.605	0.000	0.488	0.516	0.000	0.686	0.754	0.000	0.792	0.602	0.000	0.603	0.808	0.000

Note: For the hybrid approach we show only the best (H1) and worst (H2) results

TABLE III: The relative gain in average accuracy achieved by the best technique(s) of each family for each dataset over 20 test splits. The Transitive and Hybrid families are compared to Direct, and Hybrid to Transitive.

Dataset	**Metric**	**Transitive over Direct**	**Hybrid over Direct**	**Hybrid over Transitive**
Drone	MAP	22.0%	16.7%	-4.4%
	AUC	9.7%	10.3%	0.5%
	LAG	15.6%	16.6%	1.2%
TrainController	MAP	3.4%	12.4%	8.7%
	AUC	20.2%	15.9%	-3.6%
	LAG	22.8%	18.0%	-6.3%
EasyClinic	MAP	5.4%	7.2%	1.7%
	AUC	0.9%	3.5%	2.5%
	LAG	2.8%	10.7%	8.1%
EBT	MAP	11.6%	10.4%	-1.0%
	AUC	6.1%	7.3%	1.2%
	LAG	12.6%	15.2%	2.9%
WARC	MAP	-6.4%	9.3%	16.8%
	AUC	5.0%	6.5%	1.4%
	LAG	28.4%	36.0%	10.6%

the best direct technique in all datasets across all metrics. Interestingly, the best transitive technique outperformed the best hybrid technique in Drone across all metrics as well as for EasyClinic and EBT on MAP. These results are summarized in Figure 6 which shows the best results obtained for direct, transitive, and hybrid techniques. In three cases (Drone, TC, and EC) the hybrid approach performed best, followed by the transitive approach, and then the direct one.

In addition, Table III displays the relative gain in accuracy of the best performing hybrid technique over the best performing direct technique for each dataset-metric combination. Relative gain is calculated for a target (T) and baseline (B) score where the gain is of the target *over* the base calculated by:

$$RelativeGain = \frac{T-B}{B}$$

Our results so far has focused on top performing techniques for each dataset; however to understand when and where the

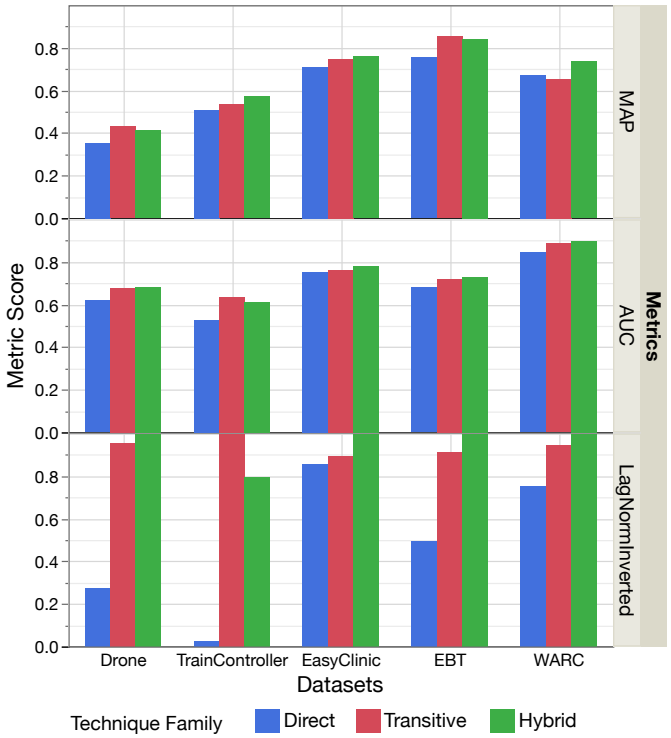


Fig. 6: The average accuracy metric scores over the 20 test splits for the best technique(s) of each direct, transitive, and hybrid family. As *LagNormInverted* is normalized, a score of 1 represents the best lag score and not perfect achievement.

transitive approach works, we now investigate its performance across individual trace queries. Figure 7 reports results for individual queries and Table V reports the significance levels of the non-parametric Wilcoxon method for identifying differences in distributions of rankings.

Figure 7 shows a large variance in accuracy scores across all techniques meaning that there are some queries that are

TABLE IV: Top performing hybrid techniques of each dataset.

Dataset	Direct Algebraic Model	Transitive Algebraic Model	Scaling Method	Path Aggreg.	Direct-Transitive Technique Aggregation
Drone	VSM	LSI	GLB	SUM	SUM
TC	VSM	LSI	GLB	SUM	SUM
EC	VSM	VSM	IND	MAX	SUM
EBT	VSM	VSM	Tie(GLB,IND)	SUM	SUM
WARC	VSM	VSM	Tie(GLB,IND)	MAX	SUM

TC = TrainController, EC = EasyClinic
GLB = Global Scaling, IND = Independent Scaling

TABLE V: Significance values of non-parametric Wilcoxon Method for differences in accuracy scores on individual trace queries for the best direct, transitive, and hybrid techniques.

Dataset	Metric	Transitive over Direct	Hybrid over Direct	Hybrid over Transitive
Dronology	MAP	0.001	0.001	0.003
	AUC	0.001	0.001	0.001
	LAG	0.001	0.001	0.004
TrainController	MAP	0.001	0.001	0.127
	AUC	0.001	0.001	0.053
	LAG	0.001	0.001	0.030
EasyClinic	MAP	0.058	0.070	0.620
	AUC	0.043	0.112	0.511
	LAG	0.036	0.081	0.506
EBT	MAP	0.001	0.001	0.074
	AUC	0.001	0.001	0.338
	LAG	0.001	0.001	0.399
WARC	MAP	0.562	0.637	0.293
	AUC	0.494	0.793	0.674
	LAG	0.494	0.833	0.600

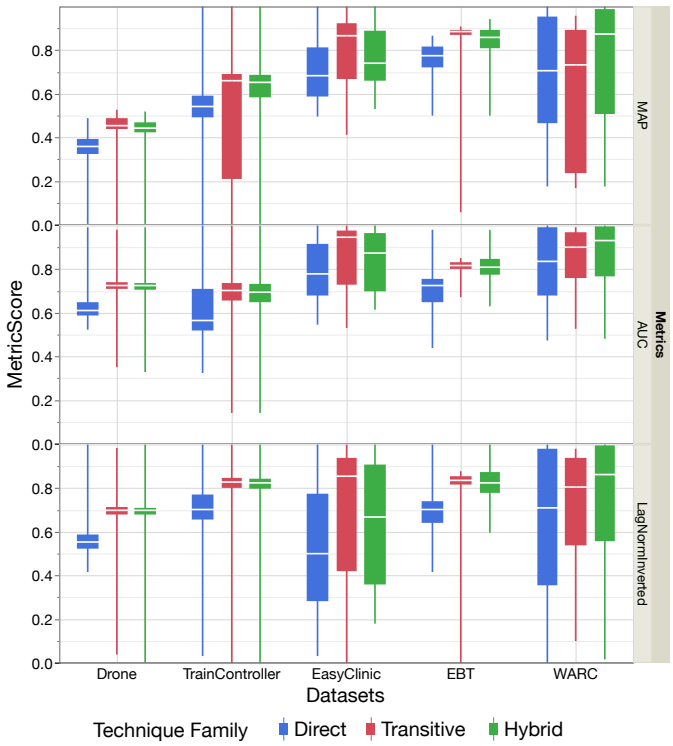


Fig. 7: Distribution of metrics scores of best performing Direct, Transitive, and Hybrid techniques for each individual project and family, requiring project-level configuration.

difficult for all techniques to perform well on. Further, there are some individual trace queries for which the best transitive technique performs significantly worse than the direct one. This is illustrated in the MAP scores for the EBT and TrainController datasets. We also notice that the best hybrid and transitive techniques were significantly more accurate than the best direct technique across 3 datasets. Interestingly, our analysis showed that the best transitive technique was only significantly better than the hybrid transitive technique in the case of AUC in Dronology.

Although the best technique using intermediate artifacts performed well, results in Table II indicate that the worst hybrid technique performed worse than the worst direct technique for TC, EC, EBT, WARC, implying that using the wrong technique would be counterproductive. This is likely the reason why Nishiwaka's CLM approach did not achieve more positive results [12]. For example, their CLM method resulted in a precision of 0.75 at a recall of 0.161 on the EasyClinic dataset; however, the best transitive technique we found in EasyClinic reaches a precision of 0.970 at a recall level of 0.165 resulting in a 29.3% relative gain. Therefore, it is important to investigate which hybrid technique is more useful and under which conditions.

From Table IV we observe that no single technique performed best across all datasets, which implies that the selection of an optimal technique will need to be tuned for each project and possibly for different trace paths within the project. On the other hand, we can observe trends that suggest a good baseline technique. For example, *VSM* outperforms *LSI* in all projects for the direct algebraic model and in 3/5 projects for the transitive algebraic model. Within the transitive technique, *global* scaling slightly outperforms *independent* as it provides best results in 4/5 datasets. Finally for link aggregation, *sum* performs best in 3/5 projects. For aggregating direct and transitive results in the hybrid approach, *sum* wins across all projects. In conclusion, *VSM* is favored over *LSI*, the choice of scaling method has little to no effect on the actual performance of a technique, and the *sum* of transitive paths performs best for aggregating direct and transitive techniques.

We refer to the aggregate of the most popular variants as the best performing *overall* technique as it returned generally good results across all projects. Note, however, that the best performing overall technique is only the top performer in EBT. Figure 8 presents the distribution of accuracy scores for the best direct technique and best *overall* technique across individual trace queries. Although this technique may not return top results for each individual project, it can be used without project-specific fine-tuning, and we still observe an increase to the median accuracy for the best *overall* technique over the best direct technique.

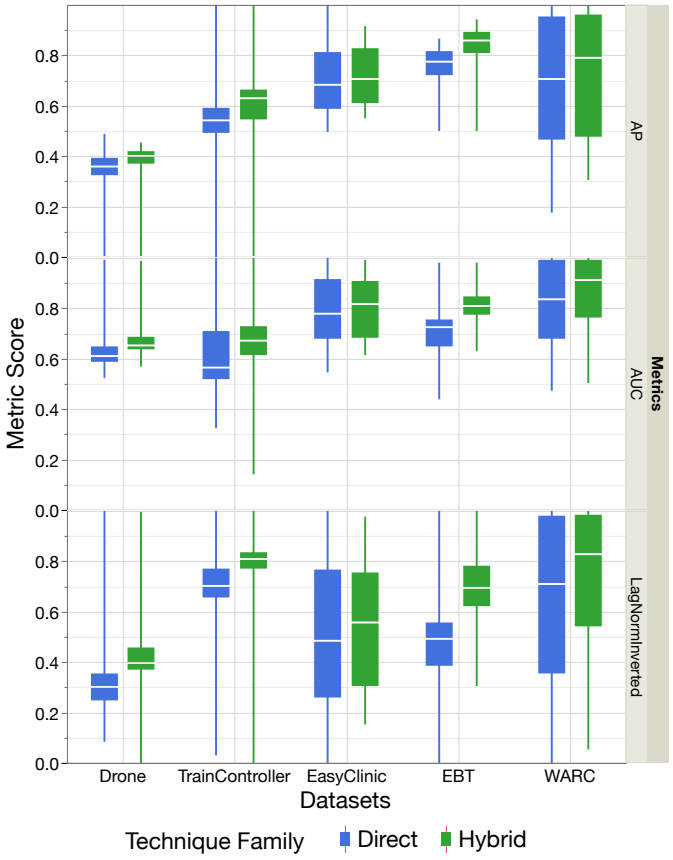


Fig. 8: Distribution of metric scores for the overall best direct and hybrid technique reported by query. This technique can be applied without project-level configuration as the best overall approach provides 'good' performance for all datasets.

V. Analysis of Results

Now that we have identified the best performing techniques we can examine where they work and where they do not.

A. Where does leveraging intermediate artifacts help?

To understand where leveraging intermediate artifacts works, we examined where the transitive technique component of each best performing hybrid technique (see Table IV) performed better than its direct counterpart. This revealed that transitive techniques help the most on queries where the source and target artifacts share little to no words in common, as illustrated by Figure 9. This figure is constructed by generating trace links using direct and transitive techniques, categorizing each traced artifact pair by the number of shared terms, and then computing the relative gain in ranking achieved using the transitive over direct technique.

We observe that the median gain is highest when the source and target artifacts share little or no words (e.g. 0 or 1). Consequently, a transitive technique is sometimes able to help a direct technique transcend the term mismatch problem between source and target artifacts. However, these term-matching problems [35] can only be alleviated when the intermediate artifacts include a common vocabulary used by both source and target artifacts as shown next.

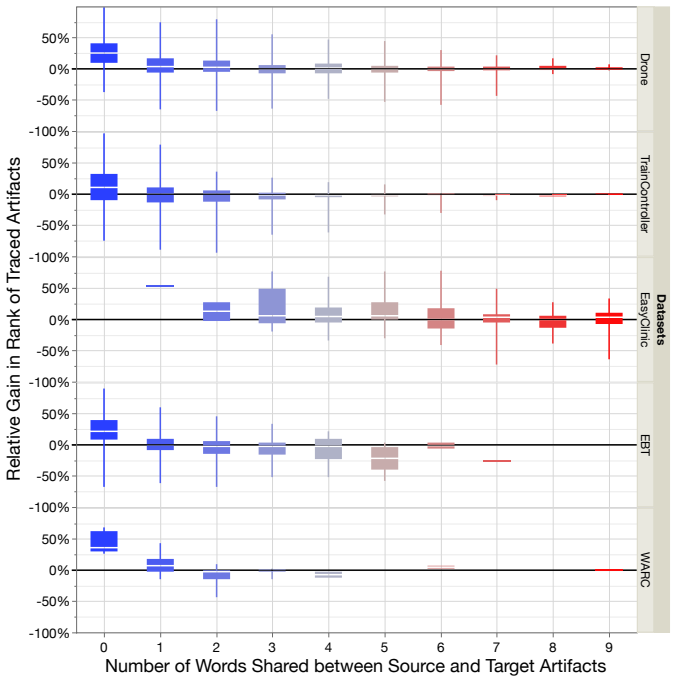


Fig. 9: Relative gain in ranking of traced artifacts for best transitive technique over best direct technique across number of intersecting words in artifacts.

B. Where do intermediate artifacts not help?

To examine where intermediate artifacts did not help end-to-end retrieval we again examine the performance of the best *direct*, *transitive*, and *hybrid* for each dataset. Previously, Figure 7 had shown us the distribution of accuracy scores across individual trace queries for the best performing techniques in each family per dataset.

In Section IV we observed that there are some queries where the best transitive technique performs significantly worse than the direct one (e.g. MAP in EBT). In general, we observe that transitive techniques also suffer from the term-mismatch problem. This is due to the fact that the algebraic models evaluated in this paper are bag-of-word models (or derivatives) and rely on the assumption that two traced artifacts must share a set of words (or similarly topic distributions in the case of LSI). Conversely, there is also an assumption that artifacts that share words should be connected via a trace link. These assumptions begin to break down as we introduce transitive traces and more distant links are introduced.

In order to test our theory we calculated the performance of the best performing hybrid techniques across different trace queries paths in each dataset and report results in Figure 10. For example, for the WARC dataset we have only examined the trace query path between non-functional requirements (0) and functional requirements (2) through system requirements (1). The other query paths explored for WARC would be: non-functional requirements $\rightarrow$ functional requirements $\rightarrow$

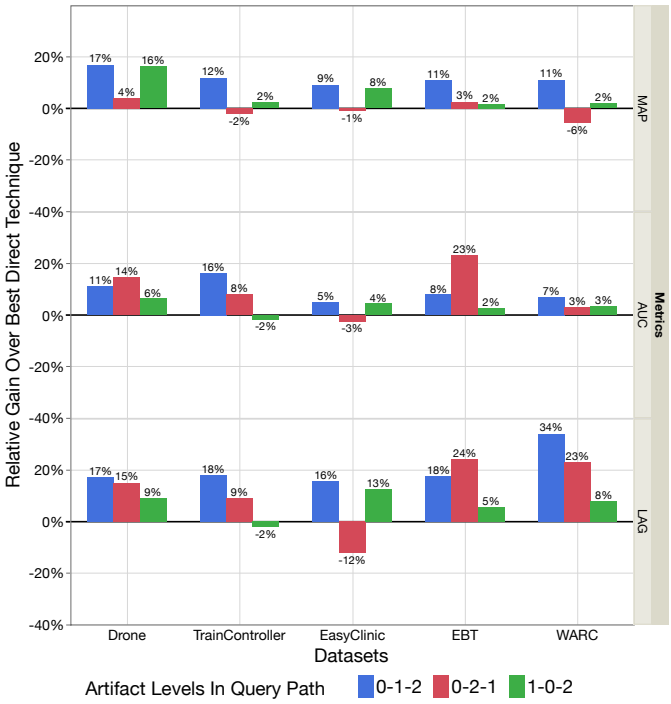


Fig. 10: Relative gain in accuracy for best hybrid technique over best direct technique across different trace query paths.

software requirements (0-2-1) and system requirements → non-functional requirements → functional requirements (1-0-2). Note, although there are technically 6 permutations of different query paths with three artifact types the results are identical when the source and target artifact types are swapped. This is due to our preprocessing step described in Section III-A in which all transitive traces are implied, so we expect to observe this symmetry in accuracy. This means that we can examine the performance of the best technique across all query paths (e.g. 0-1-2, 0-2-1, 1-0-2).

Examining Figure 10 we observe that some paths clearly benefit from the transitive approach while others do not. These results can be explained by dissecting the relationships between the artifacts and examining how they were created. Typically, projects begin by first specifying system features. Then, these features are refined by artifacts implicitly or explicitly teasing out the design and architecture of a system. Finally, once enough is known about how a system should look and behave the final refinement is made and the code is written. Note, that both agile and waterfall methodologies follow this progressive refinement with the difference being that in agile a smaller scope of features is followed through to the end per iteration. The interdependence between artifacts, as each level introduces more granularity, creates a hierarchy where each artifact is rooted to a less granular parent, ultimately creating a shared vocabulary across all child artifacts.

Transitive techniques leverage this common vocabulary to retrieve end-to-end trace links. There are three systems whose artifacts follow a clear hierarchy, namely, Dronology, TrainController, and EasyClinic. Notably, the query path following this hierarchical order (0-1-2) is the most improved path for each of these datasets across almost all metrics. WARC included non-functional requirements which are difficult to place into a hierarchy; however, we followed the path established by the developers of the dataset and observe that its use leads to the greatest accuracy across all metrics.

Queries experiencing less than 5% improvement (e.g. 1-0-2 in TrainController, 0-2-1 in EasyClinic) break the established hierarchy of the project. It is expected that differences in vocabularies between artifacts in a query path affect the performance of transitive techniques; namely, if there is a vocabulary mismatch between either source → intermediate artifacts or intermediate → target artifacts, and just one of their correspond similarity matrices is not able to compute *enough* similarity between artifacts (like when vocabularies mismatch) then the result is a near zero-valued matrix which wipes out any upstream or downstream relation scores.

VI. Threats to Validity

Our experiments are subject to several threats to validity [36]. A possible threat to internal validity is that different tracing techniques might influence our findings. To minimize this threat we adopted diverse, commonly adopted techniques that have performed well in past studies [37], [38], [26], [39], [6], [13]. However, we defer experimentation with deep learning solutions to future work as discussed in Section VIII. With respect to external validity and generalizability of our results, we experimented with only five datasets due to the lack of publicly available datasets containing three levels of artifacts. However, as suggested by Nagappan [40], our datasets spanned small, medium, and large projects, and contained different artifact types. Results indicated clear patterns of improvement across all datasets when artifacts are arranged in a hierarchical fashion, suggesting that our results extend beyond a specific type of artifacts. To support reproducibility, all materials needed for replication are made available [1]. The major threat to construct validity [41] is the accuracy and completeness of our five datasets. As described in Section III-A, we used the CLM method to infer missing trace links in datasets; however, three of the datasets were explicitly organized this way by the original engineers (e.g. Dronology, TrainController). , EasyClinic) in order to support transitivity. Finally, with respect to conclusion validity, our discussion in in Sections IV and V concluded that techniques leveraging intermediate artifacts only improved accuracy when artifacts have clear relations (e.g. hierarchy of artifacts). Given the analysis of Figure 10 there is reasonable evidence that our result holds under these circumstances.

VII. Related Work

Several areas of prior work on traceability are related to our research. Many researchers have used textual similarity to generate scores that are used to predict whether artifacts should

[1] https://github.com/thearod5/LeveragingIntermediateArtifacts

be connected through trace links [6], [14], [42], [43], [44], [35]. Additional work has leveraged supporting information to improve trace accuracy [7]. Recently researchers have proposed augmenting textual information from individual artifacts with diverse information from the project's environment [45], [46], [13], [47]. We have followed this line of research by combining transitive relation scores and textual similarities between artifacts to recover traceability links.

As we set out to combine direct textual similarities with transitive similarity scores it is unclear whether these methods are complementary; therefore, we include Gethers et al.'s recommended method of combining orthogonal IR methods in two places where this uncertainty may be true [26]. His work stemmed from the observation by Oliveto et al that some IR techniques are statistically equivalent while others seemed to provide orthogonal information [19]. Gethers et al. study explored an integration of these complementary or orthogonal methods by treating each method as an expert and using a weighted sum of the expert's opinion as the final opinion. Gethers recommends weighting each expert based on the ratio of explained variance given by principal component analysis; therefore, we refer to Gether's et al. method as *PCA*.

In the most closely related work, Nishikawa et al. first explored an automated method for calculating probabilities between source and target artifacts using intermediate artifacts named *Connecting Links Methods* (CLM) [12]. In this method, the similarity score between a source and target artifact is calculated by multiplying the textual similarities between source and intermediate artifact with that of the intermediate and target artifact. This method proved to be effective to cross language barriers, but in some cases would be outperformed by the direct textual similarity of the source and target artifacts. It was thus our motivation to investigate the benefits and methods of integrating both transitive relation scores and direct textual similarities. Nishikawa et al utilized EasyClinic, a dataset we included in our analysis too. They reported a precision of 0.75 while at a recall of 0.161. At similar or higher recall levels our technique reached a precision of 0.98. A later study by Tsuchiya et al took another look at the original methodology (connect the links) [48] and found a technique that reached 0.45 precision at a recall level of 0.72. Again, we were able to achieve a precision score of 0.59 at equal or greater recall levels.

A recent paper tackled trace recovery by leveraging transitive traces in a probabilistic model named Hierar**C**hical Pr**O**babilistic **M**odel for Softwar**E** **T**raceability (COMET) [13]. Moran et al. utilized transitive relationships as one of a few inputs to tune their probabilistic model. Their paper intersects with ours on the integration of textual similarities from multiple information retrieval (IR) techniques as well as modeling transitive links in a hierarchy; however, we differ on the method in which we integrate IR techniques and transitive links in our trace predictions. First, COMET utilizes its various IR techniques to inform the parameter estimations of a beta-distribution representing all textual similarities in a project. We, in contrast, combine multiple IR techniques into a single technique that serves as the final technique determining the ranks of artifact pairs. Both COMET and our approach allow the integration of orthogonal techniques, supporting information, and relationships between the same types of artifacts which has been shown to improve accuracy when recovering trace links [8], [26], [7], [37]. Second, COMET models transitive relationships between artifacts by examining the execution trace of code to establish transitive trace links. We are not limited to transitive relationships that must include code artifacts since our model also predicts using the textual similarities between the artifacts in a transitive relationship. For example, recently Roll et al. observed benefits in using textual similarities between source artifact to improve trace retrieval [37]. Finally, other work has begun to incorporate multiple artifact types into IR models [49], [50], [51], [48] and to consider using the structural information about a project to help aid trace recovery [52], [53], [8].

VIII. Conclusion

Our study empirically evaluated several different techniques for leveraging intermediate artifacts to improve the accuracy of end-to-end trace retrieval. Our evaluation included two direct techniques, 12 transitive, and 72 hybrid. Our results showed that the best technique leveraging intermediate artifacts outperformed the best direct technique. We also evaluated a hybrid technique composed of the overall best performing transitive and direct techniques. The hybrid approach returned better median accuracy across all metrics on individual trace queries than the best direct technique. Our subsequent analysis of the top performing hybrid techniques discovered that transitive techniques help support direct techniques when few words exist between source and target artifacts, initially helping alleviate term-matching problems. However, transitive techniques suffer from their own term-matching issues, as evidenced by little to negative performance gain on trace queries containing semantically distant artifacts.

In future work we plan to explore two open research questions. First, whilst the experiments in this paper focused on IR-based tracing techniques, we expect that the findings will apply to the emergent set of deep-learning tracing algorithms too [10], [9], [35]. Even though DL approaches are able to link artifacts based on their semantics instead of relying upon shared terms and synonyms, we have made the initial observation that both the terminology mismatch and the semantic difference increases as artifacts are further apart in the development hierarchy. If this observation holds true, then the use of transitive trace links will also be useful with DL tracing solutions. Second, we plan to explore the benefits of leveraging transitive artifacts across broader graph-based networks of traceability links including the exploration of partial networks in which trace links are missing.

Acknowledgement

This work was partially funded under USA National Science Foundation grants CCF:1901000 and CCF:1901059.

REFERENCES

[1] O. C. Z. Gotel and A. Finkelstein, "An analysis of the requirements traceability problem," in *Proceedings of the First IEEE International Conference on Requirements Engineering, ICRE '94, Colorado Springs, Colorado, USA, April 18-21, 1994*, 1994, pp. 94–101. [Online]. Available: https://doi.org/10.1109/ICRE.1994.292398

[2] J. Cleland-Huang, O. Gotel, J. H. Hayes, P. Mäder, and A. Zisman, "Software traceability: trends and future directions," in *Proceedings of the on Future of Software Engineering, FOSE 2014, Hyderabad, India, May 31 - June 7, 2014*, 2014, pp. 55–69. [Online]. Available: https://doi.org/10.1145/2593882.2593891

[3] J. Cleland-Huang, B. Berenbach, S. Clark, R. Settimi, and E. Romanova, "Best practices for automated traceability," *Computer*, vol. 40, no. 6, pp. 27–35, 2007. [Online]. Available: https://doi.org/10.1109/MC.2007.195

[4] P. Rempel, P. Mäder, T. Kuschke, and J. Cleland-Huang, "Mind the gap: assessing the conformance of software traceability to relevant guidelines," in *36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014*, 2014, pp. 943–954. [Online]. Available: https://doi.org/10.1145/2568225.2568290

[5] B. Ramesh and M. Jarke, "Toward reference models of requirements traceability," *IEEE Trans. Software Eng.*, vol. 27, no. 1, pp. 58–93, 2001. [Online]. Available: https://doi.org/10.1109/32.895989

[6] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, "Recovering traceability links between code and documentation," *IEEE Transactions on Software Engineering*, vol. 28, no. 10, pp. 970–983, Oct. 2002. [Online]. Available: http://ieeexplore.ieee.org/document/1041053/

[7] J. Cleland-Huang, R. Settimi, Chuan Duan, and Xuchang Zou, "Utilizing supporting evidence to improve dynamic requirements traceability," in *13th IEEE International Conference on Requirements Engineering (RE'05)*. Paris, France: IEEE, 2005, pp. 135–144. [Online]. Available: http://ieeexplore.ieee.org/document/1531035/

[8] C. McMillan, D. Poshyvanyk, and M. Revelle, "Combining textual and structural analysis of software artifacts for traceability link recovery," in *2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering*. Vancouver, BC, Canada: IEEE, May 2009, pp. 41–48. [Online]. Available: http://ieeexplore.ieee.org/document/5069582/

[9] J. Guo, J. Cheng, and J. Cleland-Huang, "Semantically enhanced software traceability using deep learning techniques," in *ICSE*. IEEE / ACM, 2017, pp. 3–14.

[10] J. Lin, Y. Liu, Q. Zeng, M. Jiang, and J. Cleland-Huang, "Traceability transformed: Generating moreaccurate links with pre-trained bert models," in *ICSE*. IEEE / ACM, 2021.

[11] J. Cleland-Huang, O. Gotel, and A. Zisman, Eds., *Software and Systems Traceability*. London: Springer London, 2012. [Online]. Available: http://link.springer.com/10.1007/978-1-4471-2239-5

[12] K. Nishikawa, H. Washizaki, Y. Fukazawa, K. Oshima, and R. Mibe, "Recovering transitive traceability links among software artifacts," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. Bremen, Germany: IEEE, Sep. 2015, pp. 576–580. [Online]. Available: http://ieeexplore.ieee.org/document/7332517/

[13] K. Moran, D. N. Palacio, C. Bernal-Cárdenas, D. McCrystal, D. Poshyvanyk, C. Shenefiel, and J. Johnson, "Improving the Effectiveness of Traceability Link Recovery using Hierarchical Bayesian Networks," *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pp. 873–885, Jun. 2020, arXiv: 2005.09046. [Online]. Available: http://arxiv.org/abs/2005.09046

[14] J. Hayes, A. Dekhtyar, and J. Osborne, "Improving requirements tracing via information retrieval," in *Journal of Lightwave Technology*. Monterey Bay, CA, USA: IEEE Comput. Soc, 2003, pp. 138–147. [Online]. Available: http://ieeexplore.ieee.org/document/1232745/

[15] A. D. Lucia, F. Fasano, R. Oliveto, and G. Tortora, "Recovering traceability links in software artifact management systems using information retrieval methods," *ACM Trans. Softw. Eng. Methodol.*, vol. 16, no. 4, 2007. [Online]. Available: http://doi.acm.org/10.1145/1276933.1276934

[16] D. Falessi, G. Cantone, and G. Canfora, "A comprehensive characterization of NLP techniques for identifying equivalent requirements," in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '10*. Bolzano-Bozen, Italy: ACM Press, 2010, p. 1. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1852786.1852810

[17] S. Lohar, S. Amornborvornwong, A. Zisman, and J. Cleland-Huang, "Improving trace accuracy through data-driven configuration and composition of tracing features," in *Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE'13, Saint Petersburg, Russian Federation, August 18-26, 2013*, 2013, pp. 378–388. [Online]. Available: http://doi.acm.org/10.1145/2491411.2491432

[18] H. U. Asuncion, A. Asuncion, and R. N. Taylor, "Software traceability with topic modeling," in *32nd ACM/IEEE International Conference on Software Engineering (ICSE)*, 2010, pp. 95–104.

[19] R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia, "On the Equivalence of Information Retrieval Methods for Automated Traceability Link Recovery," in *2010 IEEE 18th International Conference on Program Comprehension*. Braga, Portugal: IEEE, Jun. 2010, pp. 68–71. [Online]. Available: http://ieeexplore.ieee.org/document/5521762/

[20] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, vol. 41, no. 6, pp. 391–407, 1990.

[21] "sklearn.decomposition.TruncatedSVD — scikit-learn 0.23.2 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

[22] T. Hofmann, "Probabilistic latent semantic indexing," in *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA*, 1999, pp. 50–57. [Online]. Available: https://doi.org/10.1145/312624.312649

[23] T. Kailath, "The divergence and bhattacharyya distance measures in signal selection," *IEEE Transactions on Communication Technology*, vol. 15, no. 1, pp. 52–60, February 1967.

[24] P. Mäder, P. L. Jones, Y. Zhang, and J. Cleland-Huang, "Strategic traceability for safety-critical projects," *IEEE Softw.*, vol. 30, no. 3, pp. 58–66, 2013.

[25] Y. Shin, J. H. Hayes, and J. Cleland-Huang, "Guidelines for benchmarking automated software traceability techniques," in *SST@ICSE*. IEEE Computer Society, 2015, pp. 61–67.

[26] M. Gethers, R. Oliveto, D. Poshyvanyk, and A. D. Lucia, "On integrating orthogonal information retrieval methods to improve traceability recovery," in *2011 27th IEEE International Conference on Software Maintenance (ICSM)*. Williamsburg, VA, USA: IEEE, Sep. 2011, pp. 133–142. [Online]. Available: http://ieeexplore.ieee.org/document/6080780/

[27] M. Rath, J. Rendall, J. L. C. Guo, J. Cleland-Huang, and P. Mäder, "Traceability in the wild: Automatically augmenting incomplete trace links," in *SE/SWM*, ser. LNI, vol. P-292. GI, 2019, p. 63.

[28] Software and Requirements Engineering Center (SAREC), "Dronology: A System for Managing and Coordinating Unmanned Aerial Systems," accessed: 2018-5-4. [Online]. Available: https://dronology.info

[29] J. Cleland-Huang, M. Vierhauser, and S. Bayley, "Dronology: an incubator for cyber-physical systems research," in *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results, ICSE (NIER) 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, 2018, pp. 109–112. [Online]. Available: https://doi.org/10.1145/3183399.3183408

[30] "Center of Excellence for Software & Systems Traceability." [Online]. Available: coest.org

[31] Y. Shin, J. H. Hayes, and J. Cleland-Huang, "Guidelines for Benchmarking Automated Software Traceability Techniques," in *2015 IEEE/ACM 8th International Symposium on Software and Systems Traceability*. Florence, Italy: IEEE, May 2015, pp. 61–67. [Online]. Available: http://ieeexplore.ieee.org/document/7181530/

[32] S. K. Sundaram, J. H. Hayes, A. Dekhtyar, and E. A. Holbrook, "Assessing traceability of software engineering artifacts," *Requirements Engineering*, vol. 15, no. 3, pp. 313–335, Sep. 2010. [Online]. Available: http://link.springer.com/10.1007/s00766-009-0096-6

[33] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[34] S. Herbold, "Comments on scottknottesd in response to "an empirical comparison of model validation techniques for defect prediction models"," *IEEE Trans. Software Eng.*, vol. 43, no. 11, pp. 1091–1094, 2017. [Online]. Available: https://doi.org/10.1109/TSE.2017.2748129

[35] J. Guo, M. Gibiec, and J. Cleland-Huang, "Tackling the term-mismatch problem in automated trace retrieval," *Empirical Software Engineering*, vol. 22, no. 3, pp. 1103–1142, Jun. 2017. [Online]. Available: http://link.springer.com/10.1007/s10664-016-9479-8

[36] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln, *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012.

[37] D. Falessi, J. Roll, J. L. Guo, and J. Cleland-Huang, "Leveraging Historical Associations between Requirements and Source Code to Identify Impacted Classes," *IEEE Transactions on Software Engineering*, vol. 46, no. 4, pp. 420–441, Apr. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8423658/

[38] M. Rahimi, W. Goss, and J. Cleland-Huang, "Evolving Requirements-to-Code Trace Links across Versions of a Software System," in *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. Raleigh, NC, USA: IEEE, Oct. 2016, pp. 99–109. [Online]. Available: http://ieeexplore.ieee.org/document/7816458/

[39] D. Nir, S. Tyszberowicz, and A. Yehudai, "Locating regression bugs," in *Hardware and Software: Verification and Testing*, K. Yorav, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 218–234.

[40] M. Nagappan, T. Zimmermann, and C. Bird, "Diversity in software engineering research," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2013*. Saint Petersburg, Russia: ACM Press, 2013, p. 466. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2491411.2491415

[41] P. Ralph and E. Tempero, "Construct Validity in Software Engineering Research and Software Metrics," in *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018 - EASE'18*. Christchurch, New Zealand: ACM Press, 2018, pp. 13–23. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3210459.3210461

[42] X. Zou, R. Settimi, and J. Cleland-Huang, "Improving automated requirements trace retrieval: a study of term-based enhancement methods," *Empirical Software Engineering*, vol. 15, no. 2, pp. 119–146, Apr. 2010. [Online]. Available: http://link.springer.com/10.1007/s10664-009-9114-z

[43] 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*, vol. 1. Cape Town, South Africa: ACM Press, 2010, p. 95. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1806799.1806817

[44] M. Borg, P. Runeson, and A. Ardö, "Recovering from a decade: a systematic mapping of information retrieval approaches to software traceability," *Empirical Software Engineering*, vol. 19, no. 6, pp. 1565–1616, Dec. 2014. [Online]. Available: http://link.springer.com/10.1007/s10664-013-9255-y

[45] J. Zhou, Y. Lu, and K. Lundqvist, "A Context-based Information Retrieval Technique for Recovering Use-Case-to-Source-Code Trace Links in Embedded Software Systems," in *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*.

[50] J. Zdravkovic, M. Kirikova, and P. Johannesson, Eds., *Advanced Information Systems Engineering: 27th International Conference,* Santander, Spain: IEEE, Sep. 2013, pp. 252–259. [Online]. Available: http://ieeexplore.ieee.org/document/6619519/

[46] Y. Wang, Y. Yao, H. Tong, X. Huo, M. Li, F. Xu, and J. Lu, "Enhancing supervised bug localization with metadata and stack-trace," *Knowledge and Information Systems*, vol. 62, no. 6, pp. 2461–2484, Jun. 2020. [Online]. Available: http://link.springer.com/10.1007/s10115-019-01426-2

[47] H. Kuang, H. Gao, H. Hu, X. Ma, J. Lu, P. Mader, and A. Egyed, "Using Frugal User Feedback with Closeness Analysis on Code to Improve IR-Based Traceability Recovery," in *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*. Montreal, QC, Canada: IEEE, May 2019, pp. 369–379. [Online]. Available: https://ieeexplore.ieee.org/document/8813289/

[48] R. Tsuchiya, K. Nishikawa, H. Washizaki, Y. Fukazawa, Y. Shinohara, K. Oshima, and R. Mibe, "Recovering Transitive Traceability Links among Various Software Artifacts for Developers," *IEICE Transactions on Information and Systems*, vol. E102.D, no. 9, pp. 1750–1760, Sep. 2019. [Online]. Available: https://www.jstage.jst.go.jp/article/transinf/E102.D/9/E102.D_2018EDP7331/_article

[49] S. Baek, J.-W. Lee, and B. Lee, "Towards Recovering Fault Traceability Links by Using Information Retrieval Technique," in *Advances in Computer Science and Ubiquitous Computing*, J. J. Park, V. Loia, G. Yi, and Y. Sung, Eds. Singapore: Springer Singapore, 2018, vol. 474, pp. 1180–1185, series Title: Lecture Notes in Electrical Engineering. [Online]. Available: http://link.springer.com/10.1007/978-981-10-7605-3_188
CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, vol. 9097. [Online]. Available: http://link.springer.com/10.1007/978-3-319-19069-3

[51] R. Tsuchiya, H. Washizaki, Y. Fukazawa, T. Kato, M. Kawakami, and K. Yoshimura, "Recovering Traceability Links between Requirements and Source Code Using the Configuration Management Log," *IEICE Transactions on Information and Systems*, vol. E98.D, no. 4, pp. 852–862, 2015. [Online]. Available: https://www.jstage.jst.go.jp/article/transinf/E98.D/4/E98.D_2014EDP7199/_article

[52] A. Panichella, C. McMillan, E. Moritz, D. Palmieri, R. Oliveto, D. Poshyvanyk, and A. De Lucia, "When and How Using Structural Information to Improve IR-Based Traceability Recovery," in *2013 17th European Conference on Software Maintenance and Reengineering*. Genova: IEEE, Mar. 2013, pp. 199–208. [Online]. Available: http://ieeexplore.ieee.org/document/6498468/

[53] Y. Cao, Y. Zou, Y. Luo, B. Xie, and J. Zhao, "Toward accurate link between code and software documentation," *Science China Information Sciences*, vol. 61, no. 5, p. 050105, May 2018. [Online]. Available: http://link.springer.com/10.1007/s11432-017-9402-3