Towards Semantically Guided Traceability

Yalin Liu, Jinfeng Lin, Qingkai Zeng, Meng Jiang, Jane Cleland-Huang
University Of Notre Dame
Notre Dame, IN
yliu26@nd.edu, jlin6@nd.edu, qzeng@nd.edu, mjiang2@nd.edu, JaneClelandHuang@nd.edu

***Abstract*—In many regulated domains, traceability is established across diverse artifacts such as requirements, design, code, test cases, and hazards – either manually or with the help of supporting tools, and the resulting trace links are used to support activities such as impact analysis, compliance verification, and safety inspections. Automated tracing techniques need to leverage the semantics of underlying artifacts in order to establish more accurate trace links and to provide explanations of links that have been created in either a manual or automated fashion. To support this, we propose an automated technique which leverages source code, project artifacts and an external domain corpus to generate a domain-specific concept model. We then use the generated concept model to improve traceability results and to provide explanations of the results. Our approach overcomes existing problems with deep-learning traceability algorithms, as it does not require a training set of existing trace links. Finally, as an initial proof-of-concept, we apply our semantically-guided approach to the Dronology project, and show that it improves over other tracing techniques that do not use a concept model.**

***Index Terms*—Traceability, Repository Mining, Concept Model, Semantic Analysis, Domain Specified Ontology**

I. Introduction

Requirements traceability is recognised as an essential property in many software domains, where links are established across diverse artifacts such as bug reports, feature requests, commit messages, design documents, code, test cases, and release documents. Establishing and maintaining trace links among these artifacts helps stakeholders to perform diverse requirements engineering activities such as change impact analysis, compliance analysis, and safety assurance [5]. Despite the importance of traceability, numerous studies have observed that achieving traceability in large industrial projects can be labor intense, arduous and error-prone [14]. To mitigate these problems, automated traceability approaches have been proposed based on diverse techniques such as information retrieval, code analysis, deep learning, and topic modeling to generate candidate trace links [8], [10], [14].

Recent solutions based on deep-learning techniques (e.g., [13]) are specifically designed to overcome the common problem of term mismatches between related artifacts. These approaches can improve trace accuracy when very large corpus of data are available; however, they currently suffer from a lack of *explainability*. A concept model is a semantic network which connects phrases based on their semantic relationships. Researchers have previously attempted to leverage concept models to improve trace link generation, although most of these techniques rely on pre-built general purpose models, such as WordNet [20], ConceptNet [17], or BabelNet [21]. As software projects tend to include technical, domain-specific terminologies, we need to develop and leverage domain specific concept models. In this short paper we therefore propose an automated approach for generating domain-specific concept models, using them to improve the accuracy of generated trace links and also to provide explanations for both manually and automatically generated links. The domain concept model created in this paper includes 4 types of relationships including synonyms, acronyms, ancestors-descendent and sibling.

Our approach starts by collecting a corpus of domain-specific documents. We first create project-related search queries which we use to retrieve white papers, academic papers, and open source systems that are relevant to the domain. We dynamically analyze the textual information and the source code class hierarchy to create an concept model, and then leverage this concept model to establish traceability links . Finally, as an initial proof-of-concept study, we evaluate our approach in the domain of Unmanned Aerial Vehicles (UAV).

The remainder of the paper is organized as follows. Section II describes the concept model building process. Section III then introduces the tracing techniques that we use throughout the remainder of the paper. These include the traditional Vector Space Model (VSM), Phrase-based VSM (PVSM), and finally the Generalized VSM (GVSM) which utilizes the concept model to generate trace links. Sections IV and III apply our approach to the UAV domain – discussing the quality of the generated model and the application of our approach for trace link generation and trace link explainability. Finally, in sections VI and VII, we discuss related work, and our conclusions which lay out the prominent future research challenge.

II. Automated Concept Model Construction

To generate a domain-specific concept model we follow the three steps of (1) domain repository mining, (2) concept detection, and finally (3) hierarchical concept model construction.

A. Domain Repository Mining

An initial search for domain-specific document corpus is seeded with a set of relevant search queries. When sufficient domain expertise is available, the search queries can be created manually; otherwise a snowballing technique can be exploited [28] in which a small number of initial search terms are used to retrieve an initial set of results, and then commonly occurring domain-specific phrases are identified and utilized to seed further search queries [13]. Domain documents include online

materials such as white-papers, product descriptions, training materials, academic literature, and OSS project repositories.

B. Concept Detection

We use AutoPhrase [26], a tool that was developed to automatically extract concepts from general knowledge bases (KB) such as Wikipedia. AutoPhrase uses a concept known as distance training and part-of-speech (POS) tag-guided phrasal segmentation, to extract phrases and then to evaluate their quality. It leverages phrases found in titles, keywords, and internal hyperlinks as training examples for finding additional phrases. The approach has been shown to produce relatively high quality phrases that are representative of the domain under analysis. A complete description is provided in [26]. We also leverage the Stanford Parser [9] to produce POS (Part-of-Speech) tag annotations and to identify Universal Dependencies (UD) [23] within each sentence. From these, we extract noun phrases by identifying tokens which are bounded through the *compound* relationship and annotated with NN (Noun-Noun) POS tags. Unlike AutoPhrase, this approach uses the Stanford Parser's pre-built language model and therefore requires no project-specific training. Whereas AutoPhrase is well suited to extract concepts from complete sentences found in requirements and other textual artifacts, the Universal Dependency based approach is able to identify noun phrases from short and grammatically incomplete sentences such as code comments, itemized lists, and web pages.

C. Hierarchical concept model construction

Finally, we leverage HiGrowth [30] to identify synonyms, acronyms, siblings, and ancestor-descendant (i.e., whole-part, or is-a) relations. Each of these relations indicates a specific type of node structure in the concept hierarchy. HiGrowth uses textual pattern mining to extract tuples. For example, from the sentence *"... [classification] methods such as [SVM], [Decision Trees]..."* it infers that (1) "classification" is an ancestor of "SVM" and "Decision Trees", that (2) "SVM" and "Decision Trees" are siblings, and (3) the context is "methods". It also infers the parent-child relations from the set of synonyms, siblings, and ancestor relations, and finally addresses conflicts and redundancies by carefully maintaining structure when adding new relations into the hierarchy. This approach has been demonstrated to be effective in other domains, and we therefore investigate its effectiveness in the Software Engineering domain. In addition to the established technique we also leverage the internal structure of OO source code to extract hierarchical relationships specified as super-classes or interfaces associated with a sub- or concrete class. Furthermore, we infer additional ancestor relationships from declaration statements – for example a variable name used to name an object of a class, could suggest a child relationship (e.g. ILogger LOGGER and JSONMissionPlanReader-jsonReader).

III. Trace Link Generation Using Concept Model

There are many different information retrieval techniques for generating trace links; however, the Vector Space Model (VSM) [25] is simple and also effective. Software artifact document is preprocessed through tokenizing every sentence, removing stop words, splitting complex terms, and stemming the terms into their morphological roots. The VSM model represents artifacts as a vector of term weights. These weights are determined using a standard term frequency-inverse document frequency ($TF-IDF$) [14] to produce a vector $\vec{d} = (w_{1,d}, w_{2,d}, ..., w_{n,d})$, where $w_{i,d}$ is the term weight for the i-th term in document d. The similarity between two documents, d_1 and d_2, is computed as follows:

$$sim(d_1, d_2) = \frac{(\sum_i^n w_{i,d_1} w_{i,d_2})}{\left(\sqrt{\sum_i^n w_{i,d_1}^2} \cdot \sqrt{\sum_i^n w_{i,d_2}^2}\right)} \quad (1)$$

The VSM-nGram model is similar to VSM, except that it represents document as a vector of n-gram weights, where an n-gram is a sequence of n words in the document. The use of fixed-length n-grams provides little flexibility to take various lengthed phrases into consideration. Mao *et al.*. therefore proposed a phrase-based VSM (PVSM) that replaces n-grams with multi-length phrases.

An intrinsic limitation of VSM and PVSM is that terms and phrases only contribute to the similarity score if they appear in both source and target artifacts; while semantically similar concepts such as synonyms, acronyms and ancestors-descendants are not considered. Generalized-VSM (GVSM) [29] addresses this limitation by forming connecting semantically-related concepts.

Using the same underlying vector representation as VSM, GVSM performs pairwise matching between each concept in the source and target artifacts. It uses an algorithm, $isRelated(t_{i,d_1}, t_{j,d_2})$, that we introduce to determine whether two concepts in d_1 and d_2 are semantically related or not. The similarity formula for GVSM can be therefore represented as follows:

$$sim(d_1, d_2) = \frac{\sum_i^m \sum_j^n w_{i,d_1} \cdot w_{j,d_2} \cdot isRelated(t_{i,d_1}, t_{j,d_2})}{\sqrt{\sum_i^m w_{i,d_1}^2} \cdot \sqrt{\sum_j^n w_{j,d_2}^2}} \quad (2)$$

Algorithm 1: isRelated function

Input: t_1: concept in source artifact
t_2: concept in target artifact
G: ontology network as a directed graph
η: threshold for concept lexical similarity
τ: maximal path length
Output: R: relevance of input concepts

$R \leftarrow 0$;
$G \leftarrow G$ only keep synonyms and child edges;
if *stem(t_1) == stem(t_2)* **then**
  $R \leftarrow 1$;
else
  $node_1 \leftarrow$ fuzz_match(G.nodes, t_1, η);
  $node_2 \leftarrow$ fuzz_match(G.nodes, t_2, η);
  path $\leftarrow$ Dijkstra($node_1$, $node_2$);
  if *exist(path) AND len(path)* $\leq \tau$ **then**
end $R \leftarrow 1$;
return R

Given a concept model, $isRelated()$ leverages paths between two concepts to evaluate their relevance. In this proof-of-concept study, we implement $isRelated()$ as a binary function, which returns 1 if a valid path is found and 0 otherwise. The pseudo code of the $isRelated()$ function is shown in algorithm 1. Based on initial experimentation, we remove all links from the network except for synonyms, acronyms, and ancestor-descendants (line 2). We then utilize fuzzy matching to search for all unique, non-redundant paths that connect nodes between concepts t_1 and t_2 and use a shortest path algorithm to identify the best one. We consider two concepts to be related if they are connected via a path of length less than τ. We use fuzzy matching with the aim of normalizing the Levenshtein Distance [16] between a concept in the artifact and a node in the concept model, to increasing the number of matches. This is especially important for smaller concept networks, which is the case in most software project domains. However, it introduces the risk of incorrect matches between artifact concepts and ontology nodes. We therefore apply a threshold η, such that when $\eta = 1$, it matches concepts and nodes with identical text, and when $\eta = 0$, it randomly match concepts to nodes. We configured our system with $\eta = 0.95$.

IV. Building a Concept Model: A Domain Application

To evaluate the efficacy of our approach, we applied it to the domain of Unmanned Aerial Vehicles (UAV). We selected this domain due to the the availability of project artifacts [6], the abundance of documents describing UAV systems, and the availability of several OSS UAV projects. We address two specific research questions about the quality and nature of the generated concept model.

- **RQ1:** Which types of document are most useful for retrieving concepts?
- **RQ2:** What types of concepts are retrieved?

A. Creating a UAV Domain Repository

We built a repository of domain-related terms using text retrieved from websites, OSS, and academic papers.

• **Websites:** We seeded our search with the query terms shown in Table I and implemented a screen scraper using GoogleScraper [12] and other libraries. We retrieved $37,006$ distinct URL links; however, we limited response time to 10 seconds, and as a result retrieved 8364 html files, from which we extracted plain text using a java parser.

TABLE I: Search Terms for Constructing Domain Repository

Unmanned-Aerial Vehicles	FAA Small UAVs
Unmanned-Aerial Systems	Middleware
Pixhawk	Runtime monitoring
Drone Collision Avoidance	Control Software
Drone Coordination	Drone Simulation
Drone Traffic Management	Drone Swarm
Intelligent Drone systems	Dronekit Python
MavLink	Messaging system
Small Unmanned Aerial Systems	Publish Subscribe
UAV Ground Control Station	Software Architecture

TABLE II: OSS Projects used in the construction of the UAV domain ontology

Project	Description	Pkg
Ardupilot	OS autopilot for multicopters, planes, rovers,.....	240
Dronecode	Flight-controller, autopilot, ground control station	53
Dronekit	SDK and web API for drone applications	22
Libre Pilot	Multicopter control and other RC-models	505
MavLink	message-marshalling library for UAVs –	23
MavProxy	Ground control station for UAVs	16
Msn Plan.	Ground station for ArduPilot	366
OpenDrone	Command line aerial image procs.	15
Paparazzi	Hardware & Soft. for multirotors and fixed wings.	290
PX4	Autopilot hardware for UAVs	376
QGrnd Ctrl	Flight control & mission planning	105

• **Open-Source Repositories:** We included ten OSS for UAVs that were described in a blog [3] and listed in Table II. We downloaded all available source code and documentation from each of these OSS, then extracted comments and textual descriptions. We did not use the raw source code to build the concept model although this will be included in future work.

• **Academic Papers:** Based on some initial trial queries, we used the query terms "drone | unmanned aerial | UAV" to search for papers against the XML metadata of the DBLP [7] repository. We retrieved $1,661$ papers (dated from $1997-2019$). Of these, 154, were excluded for one of the following reasons: (1) the publication was not available as a pdf, (2) the paper was not written in English, (3) the paper was a duplicate of an already included paper. This produced $1,507$ papers.

B. UAV Concept Model

Applying the AutoPhrase and HiGrowth algorithms against the UAV repository produced a total of 3,847 concepts, organized into 3,024 associations.

C. Source of Concepts

To answer RQ1 "Which types of document are most useful for retrieving concepts?" we analyzed the contributing source of each concept, and visualized this in Fig. 1. We observe that the vast majority of terms were uniquely derived from academic papers (61.8%), a significant number (21.0%) from the OSS, and an unexpectedly low fraction (5.6%) from documents retrieved from the web. An analysis of these results showed that the terms retrieved from the web tended to be high-level terms and phrases, and somewhat redundant across websites. In addition we observed that many terms derived from websites described specific applications, for example, the term 'forrest canopies' was discovered from websites describing the use of UAVs to map deforrestation.

D. Type of Concepts

To address RQ2 we examined the concepts and relationships in the generated concept model, and summarize their occurrence as shown in Table. III. A total of 7,895 concepts were

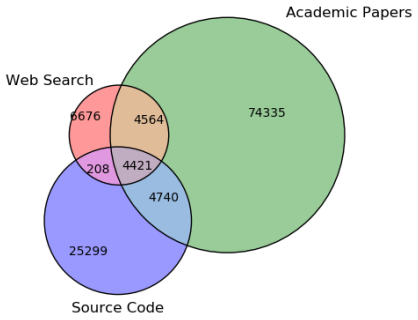


Fig. 1: Source of unique concept tokens used to build the concept model

discovered and 10,182 relationships were identified. Of these, 4,699 represented synonyms, 2,856 acronym, 784 ancestors, and 1,843 siblings. As shown, the majority of discovered relationships represented synonyms. It is also interesting to note that many synonymic relations that are particularly important for traceability purposes (e.g., GCS $\equiv$ Ground Control Station, UAV $\equiv$ Unmanned Aerial Vehicle) are not specified in general purpose thesauri such as WordNet.

TABLE III: Concepts and Relationships in UAV Ontology

Concepts	Relationships			
	Synonym	Acronym	Ancestor	Sibling
7,895	4,699	2,856	784	1,843

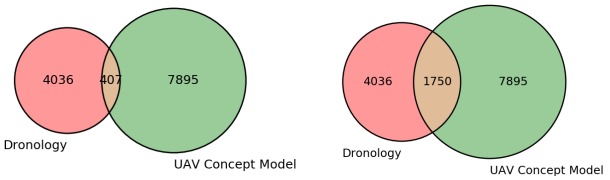
V. Evaluating Concept Models for Traceability

We evaluated the utility of the generated concept model for improving the accuracy of trace link generation and for explaining traceablity links through addressing the following research questions:

- **RQ3:** To what extent does the generated concept model overlap with artifact concepts in the targeted project?
- **RQ4:** Does the use of the generated concept model improve the accuracy of trace link generation?
- **RQ5:** Does the use of the generated concept model help explain traceability links?

A. Dronology System

We utilized artifacts from a UAV system that was not included in the construction of the domain concept model. The Dronology system [6] manages and coordinates the flights of semi-autonomous unmanned aerial vehicles (UAVs) for use in emergency response scenarios, and several datasets of artifacts including trace links are publicly available. We experimentally evaluated the use of our concept model for generating trace links between 151 Java Classes (SC) and 208 Design Definitions (DD). These artifacts create a search space of 31,408 candidate links, with 893 (2.84%) true links, provided by the Dronology developers, which served as ground truth.



(a) Overlap through direct match (b) Overlap through fuzz match

Fig. 2: Concept overlap for Dronology artifacts and UAV concept model

B. Concept Overlap

We address RQ3 by measuring the overlap of concepts between Dronology artifacts and the generated UAV concept model (i.e., those concepts that appear in the concept model and in the artifacts). We apply the same phrase extraction techniques described in Sec. II-B to the Dronology artifacts, and report overlap results in the Venn diagram depicted in Fig. 2. Part (a) of this figure shows that using direct string matching without any fuzzing resulted in concept overlap of only 407 concepts, representing only 5.16% of the total 7,895 concepts in concept model. However, when we use fuzzy matching with a threshold of η = 0.95, the match rate increased to 22.1% percent representing 1,750 matched concepts.

C. Improvement in Tracing Accuracy

To address RQ4, we generated trace links from Source Code to Design Definitions using VSM, PVSM, and GVSM as described in Sec. III. In addition, we combined GVSM with an enhanced preprocessor to create a GVSM+ model. GVSM+ includes both phrases and tokens within those phrases to formulate its vocabulary, while GVSM only includes phrases. Our rationale for taking this integrated approach is that GVSM+ is more robust in cases where specific phrases are missing from our concept model, and can compensate through the use of token-level matching as performed by VSM.

We report results using the commonly adopted metrics of F1, F2 (the harmonic mean of recall and precision), and MAP (mean average precision) [1]. Results are reported in Table. IV and show that GVSM outperformed both VSM and PVSM. PVSM returned the lowest score because direct phrase matching in VSM is less efficient than token-level matching. GVSM outperformed VSM from the perspective of both F measures and MAP, indicating that the use of a concept model can effectively support trace link generation and achieve higher link quality. By comparing GVSM+ and GVSM, we observed that GVSM suffered from the low-concept-coverage

TABLE IV: Evaluation of PVSM, VSM, GVSM+ and GVSM

Model	F1	F2	MAP
PVSM	0.115	0.183	0.145
VSM	0.196	0.250	0.210
GVSM	0.220	0.272	0.237
GVSM+	**0.253**	**0.311**	**0.286**

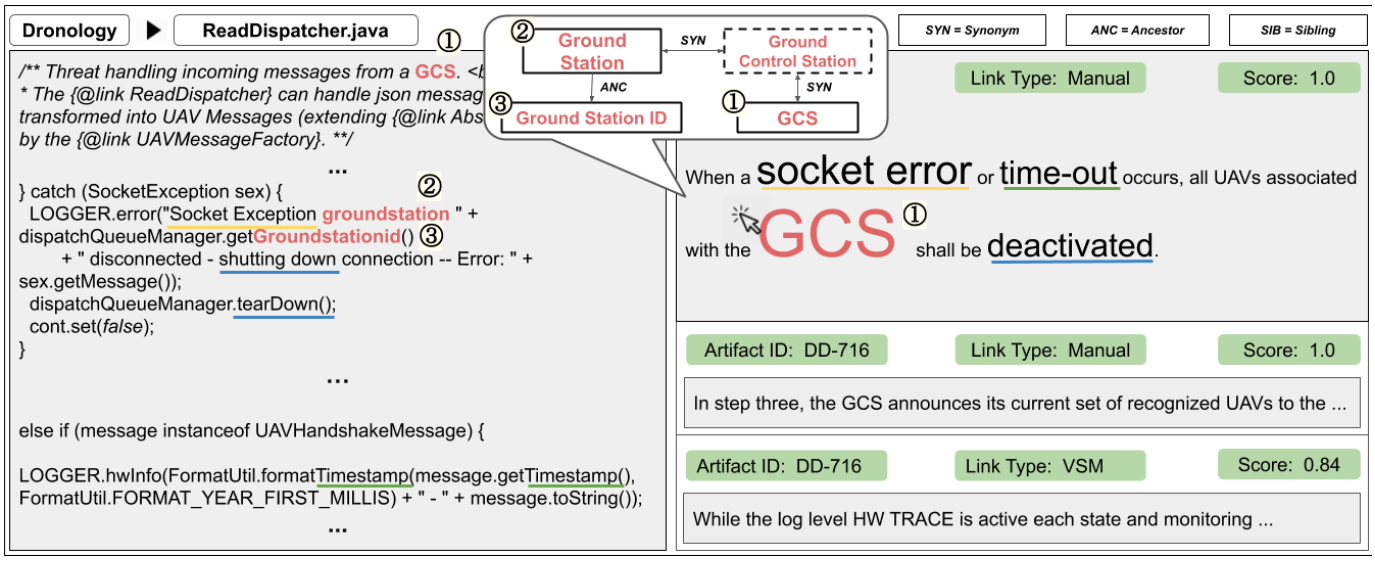


Fig. 3: Prototype GUI for explaining trace links.

issue; however, complementing it with token-level matching alleviated the problem and improved tracing performance.

D. Explaining Tracing Results

In this paper, we provide a preliminary discussion of how our approach could address RQ5. As a full user study was out of scope of this RE@Next! paper, we illustrate the way in which the UAV concept model and GVSM+ can be used to explain why two artifacts are connected via a traceability link. Fig. 3 depicts a design definition that is traced to a source code file. The code panel on the left displays the source code, while the artifact panel on the right hand side lists all potentially related design definitions. A user could click on a design definition to visualize the reason that the trace link was established. In both the Source Code and the Design Definition, the concepts recognized as semantically related are underscored with a common color (e.g. socket error and socket exception under underlined in 'yellow'. Furthermore, the terms are enlarged according to their idf (inverse document frequency) weights to reflect their contribution to the similarity score. By clicking on a concept, for example "GCS" as shown in the figure, we first highlight "ground control station" and "ground station" as related concepts, then display a rationale explaining why the GCS is related to these two concepts. This reasoning path is generated as a side product of isRelated() function, and in this case, it leverages specific parent-child associations derived from mining the source code.

VI. Related Work

In the context of IR, a thesaurus is a form of controlled vocabulary capturing the semantic relevance between pairs of terms. Several researchers have explored the use of a thesaurus for integrating semantics into software traceability. For example, Hayes *et al.*. showed that combining VSM with a manually created domain thesaurus could effectively improve tracing performance [15]. Tsatsaronis *et al.*. utilized a pre-built general-purpose thesaurus (e.g. WordNet) [27] to reduce manual efforts. In both cases the thesaurus was used as a dictionary providing the similarity coefficient between synonyms, and VSM integrated these as weightings in its similarity score computation. These approaches are limited in two primary ways: (1) manual effort is required to construct a domain specific thesaurus thus the thesaurus vocabulary is greatly constrained, (2) a general-purpose thesaurus based on WordNet lacks important terminology of many software engineering domains. Our method addresses these issues by automatically mining concept relationships from a domain corpus and OSS projects.

Mahmoud *et al.*. then compared these thesaurus-based approaches against other Semantic-augmented traceability methods such as Dirichlet Allocation (LDA), Latent Semantic Indexing (LSI), Explicit Semantic Analysis (ESA), and Normalized Google Distance [19], and showed that the thesaurus-based techniques outperformed other semantic augmented methods. However, they also reported that plain VSM achieved better MAP than thesaurus-based approaches across all three of their datasets. This supports prior findings show that VSM tends to outperform LDA and LSA [18]. We therefore adopted VSM as a baseline for our proof-of-concept experiments. In contrast to thesaurus-based approaches, our method integrates a relationship inference mechanism to find concept relevance through both "ancestor" and "synonym" relationships, while our use of GVSM+ alleviates problems caused by incomplete relationships. Since GVSM+ outperforms VSM with both MAP and F scores, it is likely to also outperform thesaurus-based techniques. A direct comparison between these two types of methods will be conducted as future work.

There are several studies investigating automated concept model construction. According to Toader *et al.*, concept model construction techniques can be categorized into structural techniques and contextual techniques [11]. The first approach relies on lexical relevance among concepts while the latter

analyzes the syntactic dependencies between a concept and its surrounding phrases. The HighGrowth model we use in this study belongs to the second category as it leverages patterns for relation discovery. Compared with structural techniques (e.g. Anh *et al.* [2]), Highgrowth tends to create concept models with higher precision and lower recall. Compared with other contextual techniques such as Poincaré Embeddings [22], On2Vec [4], and simple pattern matching, HighGrowth arranges relationships into a concept network structure, and performs a postprocessing step to reconcile inconsistencies. This further improves model precision.

We favor a concept model with high precision of generated relations in order to reduce the problem of errors propagated across relationship inferencing by the isRelated() function.

VII. Conclusion

This study represents an application of rich semantic analysis techniques to software traceability. Our approach leverages repository mining to build a domain corpus and uses an unsupervised approach for concept model construction. Although, our best results obtained using GVSM+ fall short of our goal of highly accurate traceability, they serve as a proof-of-concept that current state-of-the-art solutions based on concept modeling and the traceability algorithms that leverage semantic networks can improve the accuracy of traceability links. VSM and GVSM+ are näive models leveraging only contextual information during link generation, extra information could be leveraged along with feature based ML models such as Maxent, Random Forest, and Neural Networks to improve link quality. For example, Rath *et al.* combined VSM with features such as temporal relations, stakeholder-related information, and link relevance information to achieve high degrees of trace accuracy [24]. Replacing VSM with GVSM+ in their model could potentially lead to further improvements in accuracy.

Acknowledgments

This project has been funded by the US National Science Foundation Grants (CCF-1741781, CCF-1649448, and CCF-1513717) and Austrian Science Fund (FWF) (J3998-N319).

References

[1] Mean average precision. In L. Liu and M. T. Özsu, editors, *Encyclopedia of Database Systems*, page 1703. Springer US, 2009.

[2] T. Anh, J.-J. Kim, and S. Ng. Taxonomy construction using syntactic contextual evidence. pages 810–819, 01 2014.

[3] J. Baker. 8 open source drone projects, URL: https://opensource.com/article/18/2/drone-projects.

[4] M. Chen, Y. Tian, X. Chen, Z. Xue, and C. Zaniolo. On2vec: Embedding-based relation prediction for ontology population. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 315–323. SIAM, 2018.

[5] J. Cleland-Huang, O. Gotel, J. H. Hayes, P. Mäder, and A. Zisman. Software traceability: trends and future directions. In *Proc. of the on Future of Software Engineering*, pages 55–69, 2014.

[6] J. Cleland-Huang, M. Vierhauser, and S. Bayley. Dronology: an incubator for cyber-physical systems research. In *International Conf. on Software Engineering: New Ideas and Emerging Results, ICSE (NIER) 2018, Gothenburg, Sweden, 2018*, pages 109–112, 2018.

[7] dblp. dblp computer science bibliography. https://dblp.org, 2019.

[8] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora. Enhancing an artefact management system with traceability recovery features. In *International Conference on Software Maintenance (ICSM)*, pages 306–315, 2004.

[9] M.-C. De Marneffe and C. D. Manning. Stanford typed dependencies manual. Technical report, Technical report, Stanford University, 2008.

[10] A. Dekhtyar, J. Huffman Hayes, S. K. Sundaram, E. A. Holbrook, and O. Dekhtyar. Technique integration for requirements assessment. In *15th IEEE International Requirements Engineering Conference (RE)*, pages 141–150, 2007.

[11] T. Gherasim, M. Harzallah, G. Berio, and P. Kuntz. *Methods and Tools for Automatic Construction of Ontologies from Textual Resources: A Framework for Comparison and Its Application*, volume 471, pages 177–201. 01 2013.

[12] GoogleScraper. https://github.com/NikolaiT/GoogleScraper, 2019.

[13] J. Guo, M. Gibiec, and J. Cleland-Huang. Tackling the term-mismatch problem in automated trace retrieval. *Empirical Software Engineering*, 22(3):1103–1142, 2017.

[14] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram. Advancing candidate link generation for requirements tracing: The study of methods. *IEEE Transactions on Software Engineering*, 32(1):4, 2006.

[15] J. Huffman Hayes, A. Dekhtyar, and S. K. Sundaram. Advancing candidate link generation for requirements tracing: The study of methods. *IEEE Transactions on Software Engineering*, 32(1):4–19, 2006.

[16] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[17] H. Liu and P. Singh. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226, 2004.

[18] S. Lohar, S. Amornborvornwong, A. Zisman, and J. Cleland-Huang. Improving trace accuracy through data-driven configuration and composition of tracing features. In *9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, pages 378–388, 2013.

[19] A. Mahmoud and N. Niu. On the role of semantics in automated requirements tracing. *Requirements Engineering*, 20(3):281–300, 2015.

[20] G. A. Miller. WORDNET: a lexical database for english. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, USA, February 23-26, 1992*. Morgan Kaufmann, 1992.

[21] R. Navigli and S. P. Ponzetto. Babelnet: Building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 216–225. Association for Computational Linguistics, 2010.

[22] M. Nickel and D. Kiela. Poincaré embeddings for learning hierarchical representations. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6341–6350, Red Hook, NY, USA, 2017. Curran Associates Inc.

[23] J. Nivre, M.-C. De Marneffe, F. Ginter, et al. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, 2016.

[24] M. Rath, J. Rendall, J. L. C. Guo, J. Cleland-Huang, and P. Mäder. Traceability in the wild: Automatically augmenting incomplete trace links. ICSE '18. Association for Computing Machinery, 2018.

[25] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[26] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han. Automated phrase mining from massive text corpora. *IEEE Trans. Knowl. Data Eng.*, 30(10):1825–1837, 2018.

[27] G. Tsatsaronis, I. Varlamis, and M. Vazirgiannis. Text relatedness based on a word thesaurus. *Journal of Artificial Intelligence Research*, 37:1–39, 2010.

[28] K. Wnuk and T. Garrepalli. Knowledge management in software testing: A systematic snowball literature review. *e-Informatica*, 2018.

[29] S. M. Wong, W. Ziarko, and P. C. Wong. Generalized vector spaces model in information retrieval. In *Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 18–25, 1985.

[30] Q. Zeng, M. Yu, W. Yu, J. Xiong, Y. Shi, and M. Jiang. Faceted hierarchy: A new graph type to organize scientific concepts and a construction method. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 140–150, 2019.