A Comparison of Stemming Techniques in Tracing

David Farrar, Jane Huffman Hayes
Computer Science Department
University of Kentucky
Lexington, Kentucky, USA
david.farrar@uky.edu, hayes@cs.uky.edu

***Abstract* – We examine the effects of stemming on the tracing of software engineering artifacts. We compare two common stemming algorithms to each other as well as to a baseline of no stemming. We evaluate the algorithms on eight tracing datasets. We run the experiment using the TraceLab experimental framework to allow for ease of repeatability and knowledge sharing among the tracing community. We compare the algorithms on precision at recall levels of [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0], as well as on mean average precision values. The experiment indicated that neither the Porter stemmer nor the Krovetz stemmer outperformed the other on all datasets tested.**

Keywords – traceability, stemming, empirical research

I. INTRODUCTION

Requirements tracing, also called traceability, is an ongoing area of research in software engineering that affects many different industries. An oft-used definition of traceability is "the ability to describe and follow the life of a requirement in both the forwards and backwards direction [1]." Tracing is useful in many tasks in software engineering, and for many artifacts beyond just requirements. Tracing is beneficial in the verification and validation (V&V) and independent verification and validation (IV&V) processes [2]. Tracing has also shown its usefulness in performing change analysis, especially in the case of large projects.

There has been ongoing work to formulate the tracing problem as an information retrieval (IR) task [2]–[4]. By formulating the tracing activities as an IR problem, we can leverage the extensive research and knowledge acquired over the years by that community. This approach showed significant gains over prior techniques in terms of the precision and recall.

Stemming has become a common part of the IR process when attempting to correlate a query to documents in a document collection. The process of stemming reduces similar words to their morphological root or "stem." This reduction allows for simplified matching between similar words. The Porter stemmer [5] appears to have become the most prevalent stemming technique applied in information retrieval.

This study focuses on how diverse stemming techniques influence the performance of information retrieval applied to requirements tracing. We compare two stemming techniques, Porter [5] and Krovetz [6], against each other and against a baseline of no stemming. The study attempts to determine if the standard suffix stemming approach of the Porter stemmer outperforms the morphologically focused Krovetz stemmer in the domain of requirements tracing. The experiment was undertaken using the TraceLab workbench [7], aimed to assist in the development of tracing experiments and in the transfer of tools for research reproduction and further innovation. We leveraged an existing TraceLab experimental package for this experiment and existing TraceLab components were used where possible.

The experiment showed that for three of the eight datasets, the tracing using no stemming outperformed either stemming technique. On two of the datasets, the Krovetz stemmer outperformed the other two methods. On the remaining three datasets, the Porter stemmer outperformed the other techniques.

We organize the paper as follows: Section II discusses related work and helps pertinent background concepts. Section III discusses the experimental design and the execution of the experiment. Section IV analyzes the data gathered from the experiment. Section V presents the conclusions and future work.

II. RELATED WORK

This section provides information on work related to tracing, IR, and stemming.

A. Tracing

Requirements tracing has been a focus of interest and research for decades. Pierce [8] is commonly referenced in regards to historical examples of the development of traceability tools. Pierce developed "The Requirements Tracing Tool" to assist with V&V and change analysis by tracing requirements throughout the phases of software development. The tool was initially used on a naval undersea sensor system, and was then repurposed to support the Cruise Missile Mission Planning Project. Pierce's tool focused on the construction of a database that leveraged keywords from the requirements.

More recently, a variety of approaches to the requirements tracing problem have been applied, including keyword detection and matching [9], reference models [10], and information retrieval [2], [3], [11]. As discussed above, this work is targeting the enhancement of requirements tracing using information retrieval techniques. The IR approach is largely bring used for performing tracing after-the-fact, as opposed to the upfront construction of requirements tracing artifacts.

B. *Information Retrieval*

Information retrieval attempts to identify relevant information from a large dataset generally composed of natural language text. An IR problem is formulated such that it is composed of a document collection and a query. The document collection is a large dataset, or a representation of the dataset, that a user wishes to query for relevant data. The query is a representation of the information that the user desires to locate in the document collection. Sanderson and Croft [12] provide a history of information retrieval beginning with librarianship, progressing through computerization, and finishing with descriptions of techniques used at the time of its publishing in 2012. The paper discusses topics such as the move towards ranked retrieval and the introduction of relevance feedback. Sanderson and Croft also describe the important concept of term weighting schemes such as term frequency-inverse document frequency (tf-idf).

C. *Stemming*

Stemming, also referred to as suffixing, is a process of reducing words to their morphological root or word stem. The word stem may or may not be a "real" word, depending upon the stemming approach used. The purpose of word stemming in IR is to conflate words (combine into one) with the same stems, assuming that they have like meanings. The stemming technique may be as simple as removing pluralization suffixes from words (such as s or es), or more complicated approaches that attempt to maintain meanings and incorporate dictionaries.

Lovins [13] developed one of the earliest documented stemming algorithms in 1968. The Lovins stemming algorithm focused on longest match stemming for the English language. Martin Porter developed the Porter stemming algorithm [5] in 1980. The Porter stemming algorithm focuses on suffix removal to assist in the conflation of words.

In 1993, Robert Krovetz [6] described his work in determining if there is benefit to using word morphology in the stemming process. Krovetz attempted to maintain the meaning of words throughout the stemming process to avoid the over-stemming problem associated with strong stemmers, such as the Porter stemmer, which can cause faulty conflation of words. Krovetz incorporated a suffixing process that focused on repetitive minimal suffixing alongside a machine-readable dictionary. Throughout the repetitive suffixing steps, the results are checked against a dictionary of commonly mis-stemmed words. If a word is found in the dictionary, it is either stemmed to a known root or instructed to stop any further stemming. Krovetz's work led to the creation of what later became known as the Krovetz stemmer. The Krovetz stemmer is considered a lightweight stemmer, in that it is not as susceptible to over-stemming, but may suffer from under-stemming of words.

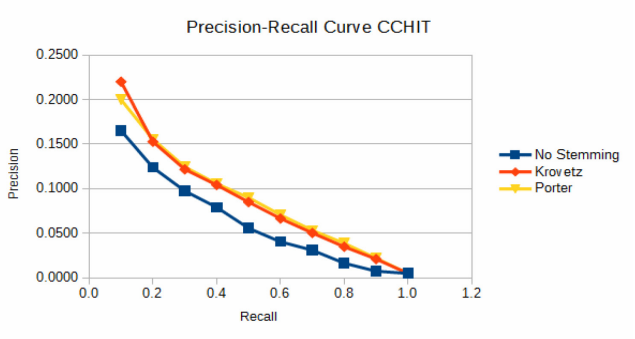

Fig. 2. Precision-Recall curve for CCHIT.

III. STUDY

The goal of this experiment is to determine if there is a difference in the performance of a tracing experiment when the stemming technique is varied. We chose the Porter and Krovetz stemming techniques for comparison as they are prevalent stemming techniques use in information retrieval. We evaluate the two stemming techniques against a baseline where no stemming is performed in the requirements tracing process.

A. *Analysis Method*

The standard metrics for evaluating information retrieval techniques of precision, recall, and mean average precision are used to compare the stemming techniques.

Precision indicates the quality of the links retrieved in the information retrieval process. Precision is a measure in the range of 0 to 1, calculated by:

$$Precision = \frac{|\#\ of\ Correct\ Links\ Returned|}{|\#\ of\ Returned\ Links|}.$$

Recall indicates the quality of the links returned on a scale of 0 to 1. Recall indicates the percentage of the total number of true links returned out of the total

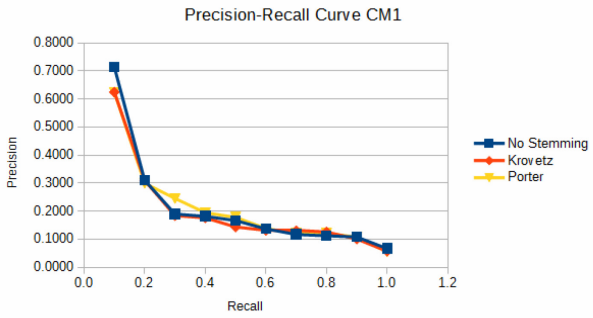

Fig. 3. Precision-Recall curve for CM1.

number of true links and is calculated by:

$$Recall = \frac{|\#\ of\ Correct\ Links\ Returned|}{|Total\ \#\ of\ Correct\ Links|}.$$

Another method of evaluating information retrieval techniques is the mean average precision (MAP), which is popular among the TREC IR community [14]. MAP is preferred in many cases, as it gives a clearer indication of where on the list of retrieved links the correct links are located. A higher MAP score indicates that more of the true links returned are near the beginning of the list of links. MAP is calculated on a scale of 0 to 1 by:

$$MAP = \frac{\sum\ (Average\ Precision\ per\ query)}{|\#\ of\ queries|}.$$

The MAP metric is the primary evaluation criteria, as it provides an indication of the precision across the recall levels. As this experiment does not focus on the quality of a specific recall value, or number of links, MAP provides a useful point of comparison between stemming techniques.

B. *Hypotheses*

Null Hypothesis (H0): There will be no difference between the precision (P), recall (R), and mean average precision (MAP) when different stemming techniques are incorporated into the tracing process. The stemming techniques compared will be no stemming (SN), Porter stemming (SP), and the Krovetz stemmer (SK).

Null Hypothesis

1. H0: $MAP_{SN} = MAP_{SP} = MAP_{SK}$

Alternate Hypothesis:

1. $MAP_{SN} > MAP_{SP} > MAP_{SK}$
2. $MAP_{SN} > MAP_{SK} > MAP_{SP}$

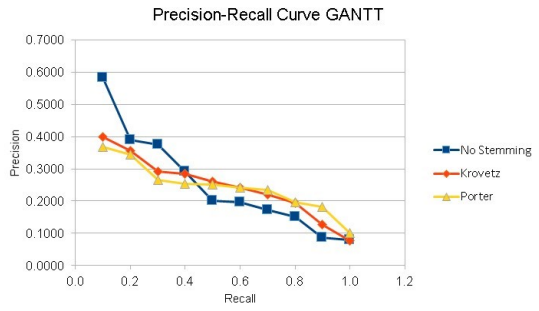

Fig. 4. Precision-Recall curve for GANTT.

3. $MAP_{SK} > MAP_{SN} > MAP_{SP}$
4. $MAP_{SK} > MAP_{SN} > MAP_{SP}$
5. $MAP_{SK} > MAP_{SP} > MAP_{SN}$
6. $MAp_{SP} > MAP_{SK} > MAP_{SN}$
7. $MAP_{SP} > MAP_{SN} > MAP_{SK}$

The independent variable for this experiment is the stemming technique used in processing the source and target tracing artifacts. The stemming techniques studied are:

1. No Stemming (SN)
2. Porter Stemmer (SP)
3. KStem/Krovitz Stemmer (SK)

The dependent variables evaluated are precision, recall, and MAP. The tracing process is static, excluding the varying of the stemming technique.

C. *Datasets*

We evaluate the stemming techniques on several existing requirements tracing datasets. The datasets are available at www.coest.org [15], with the addition of the Pine dataset previously used in tracing studies at the University of Kentucky [16]. The datasets vary in both size and application domain.

The datasets include:

1. CCHIT [17] - A project from the healthcare domain consisting of 116 World Vista requirements as source artifacts, 1064 CCHIT healthcare regulatory codes as target artifacts, and 587 true links.
2. CM-1 [2] - A NASA Instrument project consisting of 235 requirements as source artifacts, 220 design elements as target artifacts, and 361 true links.
3. GANTT [18] - A software project to generate Gantt charts consisting of 17 high level requirements as source artifacts, 69 low level requirements as target, and 68 true links.

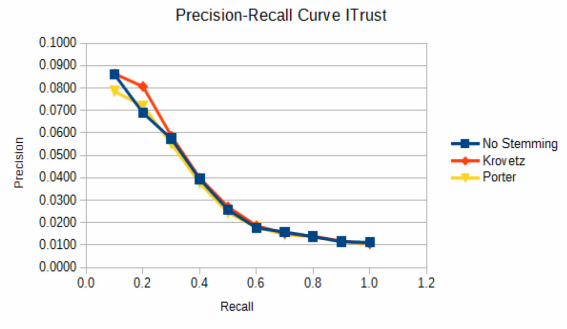

Fig. 5. Precision-Recall curve for ITrust.

4. ITrust [19] - Another healthcare related project consisting of 124 use cases as source artifacts, 367 java source files as target artifacts, and 400 true links.
5. Pine [16], [20] - An email program consisting of 49 requirements as source artifacts, 133 test cases as target artifacts, and 247 true links.
6. SMOS [21] - A School Monitoring software project consisting of 67 use cases as source artifacts, 100 code modules as target artifacts, and 1044 true links.
7. WARC Functional Requirements Specifications (FRS)

to System Requirements Specifications (SRS) [22], [23] - A web archive tool consisting of 42 source artifacts , 89 target artifacts, and 78 true links.

8. WARC Non Functional Requirements (NFR) to Software Requirements Specifications (SRS) [22], [23] - Web Archive Tool: 21 source artifacts, 89 target artifacts, and 58 true links.

D. Setup

The experiment was designed to use the TraceLab framework [7], [24], [15], developed to facilitate tracing experimentation. The TraceLab community has created a number of modular components that can be leveraged to quickly setup and reproduce prior tracing experiments. TraceLab comes with the majority of the components necessary for this experiment. Existing components used in this experiment include several tracing artifact import components, a Porter stemmer component, a stopword removal component, a vector space model using tf-idf tracing component, and an export component.

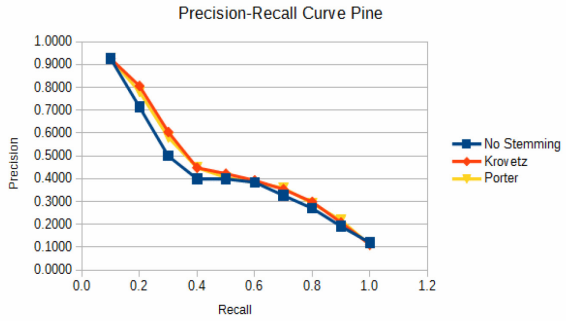

Fig. 6. Precision-Recall curve for Pine.

The existing TraceLab component contained a Porter stemming component, but did not contain a stemming component that incorporated the Krovetz stemmer. We created a Krovetz stemming component that used the Krovetz Stemmer Version 3.4 library developed by the Lemur Project [25], part of their Indri Toolkit. The Toolkit has been used and validated in a number of information retrieval research projects. We developed the component to use the same inputs and outputs as the Porter stemmer component. This allows the stemming algorithms to be swapped easily without any changes to the TraceLab experiments.

The included TraceLab components did not provide a metrics generation component that produced the metrics in a format that could be saved to disk. The source code for a series of components by the SEMERU group at the College of William and Mary was included in the GitHub repository for TraceLab [24]. The SEMERU components are covered by the GNU version 3 license and thus freely available for modification. The "SEMERU - OverallMetrics Computation" and the "SEMERU - Metrics Per Source Artifact" components were modified to store the computed tracing metrics as a TLSimilarityMatrix collection format that could then be saved to disk using an existing component. This allowed for the existing metrics calculations to be unaltered and allow for the data to be stored for review and processing.

The TraceLab experiment used for the evaluation of the stemming algorithms is shown in Fig. 1 (see Appendix, package available at selab.netlab.uky.edu/homepage/pages/TraceLabCatalogue/TraceLabCatalogue/Stemming_TraceLab_Project.7z). Each block in the experiment is a TraceLab component. The arrows between the components set constraints that force a component to wait on a prior component to complete. All experiments begin at the start block and end at the end block.

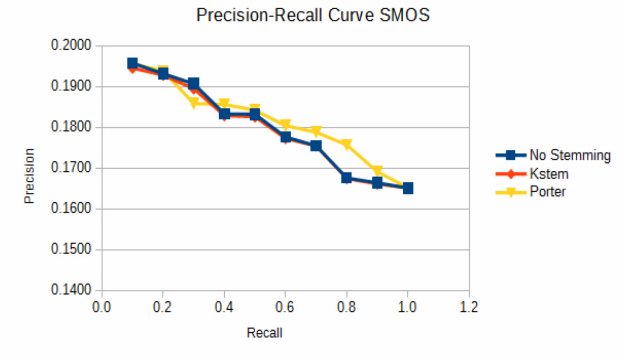

Fig. 7. Precision-Recall curve for SMOS.

In this experiment, the initial components load the source, target, stopword list, and oracle/answer/gold standard matrix from file. Next, the source and target artifacts are processed by the cleanup preprocessor, which converts all letters to lowercase and removes all non-alphanumeric letters. Stopwords, from the Fox stopword list [26], are removed from the source and target artifacts. This removes the most common English language words from the artifacts, to avoid matches on frequent words such as "the" or "a." At this point, the source and target artifacts are processed by the TF-IDF VSM Tracer component to generate a similarity matrix using the cosine between the inputs. This produces the non-stemmed similarity matrix which is processed by the "Overall Metrics - Matrix" component to determine the precision, recall, and MAP scores by comparing the similarity scores against the answer matrix for the dataset. The scores are saved to disk using the "CSV Similarity Matrix Exporter" component.

The source and target artifacts are processed by the Krovetz stemming component to produce stemmed artifacts. The artifacts are then processed by the tracing component and the metrics are saved to disk. This process is repeated again by stemming the source and target artifacts with the Porter stemming component. This same process is repeated for each dataset analyzed. The only changes made between datasets had to do with exchange components used to import the artifacts, as there were several file formats used. It appeared best to leverage the existing import components rather than

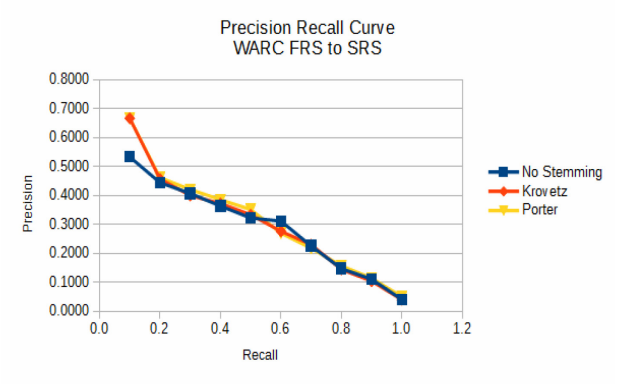

Fig. 8. Precision-Recall curve for WARC FRS to SRS.

modify the format of the datasets, which could have led to corruption of the data.

E. Threats to Validity

The main threat to construct validity is that there could be some fundamental flaw regarding the structure and formulation of the experiment. This was reduced in that any filtering and cleanup of the data has been consistent between all treatments. We used a popular general-purpose stopword list for the English language, and not customized for use with a particular dataset or stemming algorithm. We used standard IR analysis metrics.
A possible threat to internal validity is that the first author is both the designer of the experiment and the developer of the software. This could impose an unintended bias on the output of the experiment. This is somewhat limited by the use of TraceLab and many of its included components. The newly generated components for this experiment build on existing code developed by other institutions, which should limit an unintended bias toward a particular outcome.

Although the use of existing tools assists in reducing the possibility of unintended bias, it leaves open the possibility that there could be flaws in the metrics generation tools used for this experiment. As these tools were developed and used in other tracing experiments, it is unlikely that they are flawed, but it cannot be ruled out entirely. In the event that the analysis tools are flawed, the impact to the outcome is reduced in that all stemming techniques use the same analysis tools, and would most likely be impacted equally.

In regards to threats to external validity, there is the possibility that the datasets used are not representative. This risk is reduced by the fact that these datasets come from a range of sources and vary dramatically in size and domain. These datasets have been used in many other tracing experiments.

F. Execution

The datasets were loaded into the TraceLab experiment one at a time. The artifact import components were chosen to allow the import of the file format of the given dataset. The source and target artifacts were loaded such that the source and target artifact matched those used in the answerset. As tracing can be done in either direction, it was determined that the artifacts should be chosen such that the answerset did not have to be modified. This avoids possible corruption of the answerset, due to manipulation of its data, as well as simplifying the experiment.

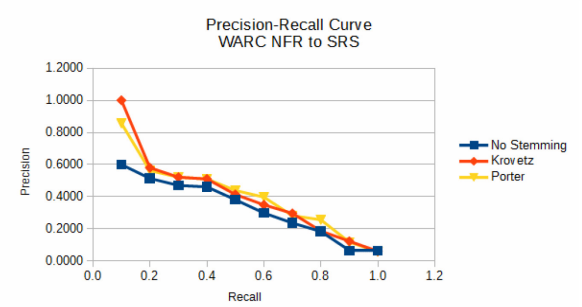

Fig. 9. Precision-Recall curve for WARC NFR to SRS.

IV. ANALYSIS

The metrics generated from the TraceLab components allow for comparison between the two stemming techniques and the baseline of no stemming. The precision values for the three methods were compared at recall values of [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]. The precision vs recall graphs are shown in Fig. 2 through 9. From the CCHIT dataset (Fig.2) it can be observed that the Krovetz stemmer outperforms the other methods at the 10% recall level, but then falls in line with the Porter stemmer from 20% - 100% recall. Both stemming techniques outperformed the baseline of no stemming across the range of recall levels. The CM-1 dataset (Fig.3) shows that the baseline technique outperforms the stemming techniques at the 10% recall, but the methods are roughly consistent with each other from 20% - 100% recall.

The GANTT dataset (Fig.4) indicates that the baseline technique outperforms the stemming techniques for the 10% - 30% recall levels, but underperforms the stemming techniques from 50% - 100% recall. The ITrust dataset (Fig.5) showed similar precision values for each technique throughout the range of recall levels. In the Pine dataset (Fig.6), the stemming techniques only show an improvement over the baseline for the 20% - 40% recall levels. In the SMOS dataset (Fig.7), the Porter stemmer shows improved precision over the range of 60% - 90% recall. In the WARC FRS-SRS dataset (Fig.8), the Krovetz stemmer shows an improved precision at the 10% recall level after which the three techniques closely match in precision values. Finally, in the WARC NFR-SRS dataset (Fig.9), both the Porter and Krovetz stemmers show a large improvement in precision over the baseline at the 10% recall level after which the stemming techniques maintain a slight improvement over the baseline.

As can be observed from the precision vs recall graphs, the range of precision values varies dramatically between datasets. It can also be observed that there is not one

approach that appears to outperform the other techniques on all the datasets. The same observation can be made when comparing the mean average precision (MAP) values for the different approaches on the datasets as shown in Fig.10 and Table 1. A review of the MAP scores show that the baseline method of no stemming outperformed the stemming techniques on the CM1, ITrust, and WARC NFR-SRS datasets. The Krovetz stemmer outperformed the other methods on the GANTT and the WARC FRS-SRS datasets. Finally, the Porter stemmer outperformed the other methods on the CCHIT, Pine, and SMOS datasets.

The precision scores at 100% recall of each source artifact for each dataset were analyzed for normality, and it was determined that the data did not meet the requirements for the T-Test. As such, the data were analyzed using the Wilcoxon method to determine if there was statistical significance to the results. On the CCHIT dataset, the Krovetz stemmer was shown to outperform the baseline with $p = 0.02202$, and the Porter stemmer outperformed the baseline with $p = 0.00338$. On the Pine dataset, the Krovetz stemmer was shown to outperform the baseline with $p = 0.00112$, and the Porter stemmer outperformed the baseline with $p = 0.002$. The other data showed no statistical significance with a significance level of $p \leq 0.05$.

TABLE I. MAP BY DATASET AND STEMMING ALGORITHM

	No Stemming	Krovetz	Porter
CCHIT	0.22	0.24	0.25
CM1	0.4	0.4	0.39
GANNT	0.46	0.48	0.45
iTrust	0.15	0.14	0.13
Pine	0.64	0.7	0.71
SMOS	0.23	0.23	0.23
WARC FRS-SRS	0.64	0.65	0.64
WARC NFR-SRS	0.61	0.6	0.62

V. CONCLUSION AND FUTURE WORK

We investigated how different stemming techniques influence the generation of similarity matrices for a set of disparate requirements tracing datasets. We compared the Porter and Krovetz stemming algorithms against a baseline of no stemming to determine how the generated similarity scores are impacted. From the results, it was observed that neither the Porter nor Krovetz stemmer appear to inherently outperform the other in all cases, or even outperform the non-stemming baseline approach. The benefits of stemming appear to vary greatly depending upon the underlying dataset.

This work can be extended by investigating the datasets in more detail to determine what characteristics may lend themselves to a particular stemming technique. Perhaps the stemming technique used in an information retrieval task could be determined by first observing the fundamental characteristics of a dataset.

It may also be beneficial to investigate the similarity matrices generated in more detail, to determine if the differing stemming techniques lead to higher scoring of differing links. It may be possible to develop a composite approach that incorporates the similarity matrices generated from two or more stemming approaches. A composite approach may be more computationally intensive than a single approach, but may lead to simplification of user tasks.

ACKNOWLEDGMENT

We thank NSF for partially funding this work under grants CCF-1511117 and CICI 1642134.

REFERENCES

[1] O. C. Z. Gotel and C. W. Finkelstein, "An analysis of the requirements traceability problem," in *Proceedings of IEEE International Conference on Requirements Engineering, 1994,* pp. 94-101.

[2] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, "Advancing candidate link generation for requirements tracing: the study of methods," *IEEE Trans. Software Eng.*, vol. 32, no. 1, pp. 4–19, Jan. 2006.

[3] J. H. Hayes, A. Dekhtyar, and J. Osborne, "Improving requirements tracing via information retrieval," in *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.* pp. 138–147.

[4] 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, Sep. 2007.

[5] M. F. Porter, "An algorithm for suffix stripping," Program: Electronic Library and Information Systems, 2006, Vol.40(3), p.211-218 , 2006.

[6] R. Krovetz, "Viewing morphology as an inference process," in *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '93,* pp.191-202.

[7] Ed Keenan, Adam Czauderna, Greg Leach, Jane Cleland-Huang, Yonghee Shin, Evan Moritz, Malcom Gethers, Denys Poshyvanyk, Jonathan Maletic, Jane Huffman Hayes, Alex Dekhtyar, Daria Manukian, Shervin Hossein, and Derek Hearn, "TraceLab: An experimental workbench for equipping researchers to innovate, synthesize, and comparatively evaluate traceability solutions," in *2012 34th International Conference on Software Engineering (ICSE)*, 2012, pp. 1375–1378.

[8] R. A. Pierce, "A Requirements Tracing Tool," in *Proceedings of the Software Quality Assurance Workshop on Functional and Performance Issues*, 1978, pp. 53–60.

[9] J. H. Hayes, "Risk reduction through requirements tracing," in Published in the *Conference Proceedings of Software Quality Week* 1990, San Francisco, California, May 1990, pp. 1-25.

[10] B. Ramesh and M. Jarke, "Toward reference models for requirements traceability," *IEEE Trans. Software Eng.*, vol. 27, no. 1, pp. 58–93, Jan. 2001.

[11] A. Mahmoud and N. Niu, "Using Semantics-Enabled Information Retrieval in Requirements Tracing: An Ongoing Experimental Investigation," in *2010 IEEE 34th Annual Computer Software and Applications Conference*, 2010, pp. 246–247.

[12] M. Sanderson and W. B. Croft, "The History of Information Retrieval

Research," *Proc. IEEE*, vol. 100, no. Special Centennial Issue, pp. 1444–1451, May 2012.

[13] J. B. Lovins, "Development of a Stemming Algorithm," Mechanical Translation and Computational Linguistics, Vol. 11, No. 1-2, 1968, pp. 22-31.

[14] C. D. Manning, P. Raghavan, H. Schütze, and Others, *Introduction to information retrieval*, vol. 1. Cambridge University press Cambridge, 2008.

[15] "CoEST Website," *CoEST.org Center of Excellence for Software & Systems Traceability*. [Online]. Available: www.coest.org. [Accessed: 30-Jan-2019].

[16] H. Sultanov, J. H. Hayes, and W.-K. Kong, "Application of swarm techniques to requirements tracing," *Requirements Eng*, vol. 16, no. 3, pp. 209–226, Sep. 2011.

[17] Y. Shin and J. Cleland-Huang, "A comparative evaluation of two user feedback techniques for requirements trace retrieval," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, 2012, pp. 1069–1074.

[18] E. A. Holbrook, J. H. Hayes, and A. Dekhtyar, "Toward Automating Requirements Satisfaction Assessment," in *2009 17th IEEE International Requirements Engineering Conference*, Atlanta, Georgia, USA, pp. 149–158.

[19] Meneely, A, Smith, B. and Williams, L., The iTrust Electronic Health Records System in *Software and Systems Traceability*, Huang, Jane, Gotel, Orlena, Zisman, Andrea (Eds.) 2011.

[20] "Pine Information Center." [Online]. Available: http://www.washington.edu/pine. [Accessed: 30-Jan-2019].

[21] 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, pp. 133–142.

[22] "Google Code Archive - Long-term storage for Google Code Project Hosting." [Online]. Available: http://code.google.com/p/warc-tools/. [Accessed: 30-Jan-2019].

[23] W.-K. Kong, J. H. Hayes, A. Dekhtyar, and O. Dekhtyar, "Process improvement for traceability: A study of human fallibility," in *2012 20th IEEE International Requirements Engineering Conference (RE)*, Chicago, IL, USA, pp. 31–40.

[24] CoEST, "CoEST/TraceLab," *GitHub*. [Online]. Available: https://github.com/CoEST/TraceLab. [Accessed: 30-Jan-2019].

[25] "Lemur Project Home." [Online]. Available: https://www.lemurproject.org. [Accessed: 30-Jan-2019].

[26] C. Fox, "A Stop List for General Text," *SIGIR Forum*, vol. 24, no. 1–2, pp. 19–21, Sep. 1989.

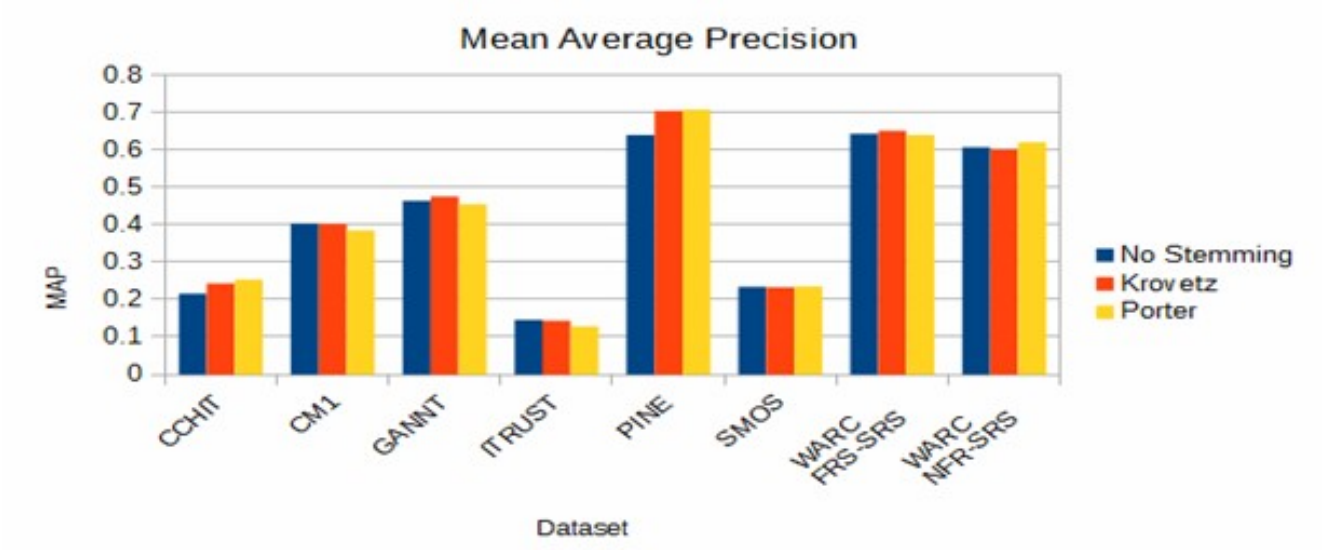

Fig. 10. MAP for all datasets.

APPENDIX

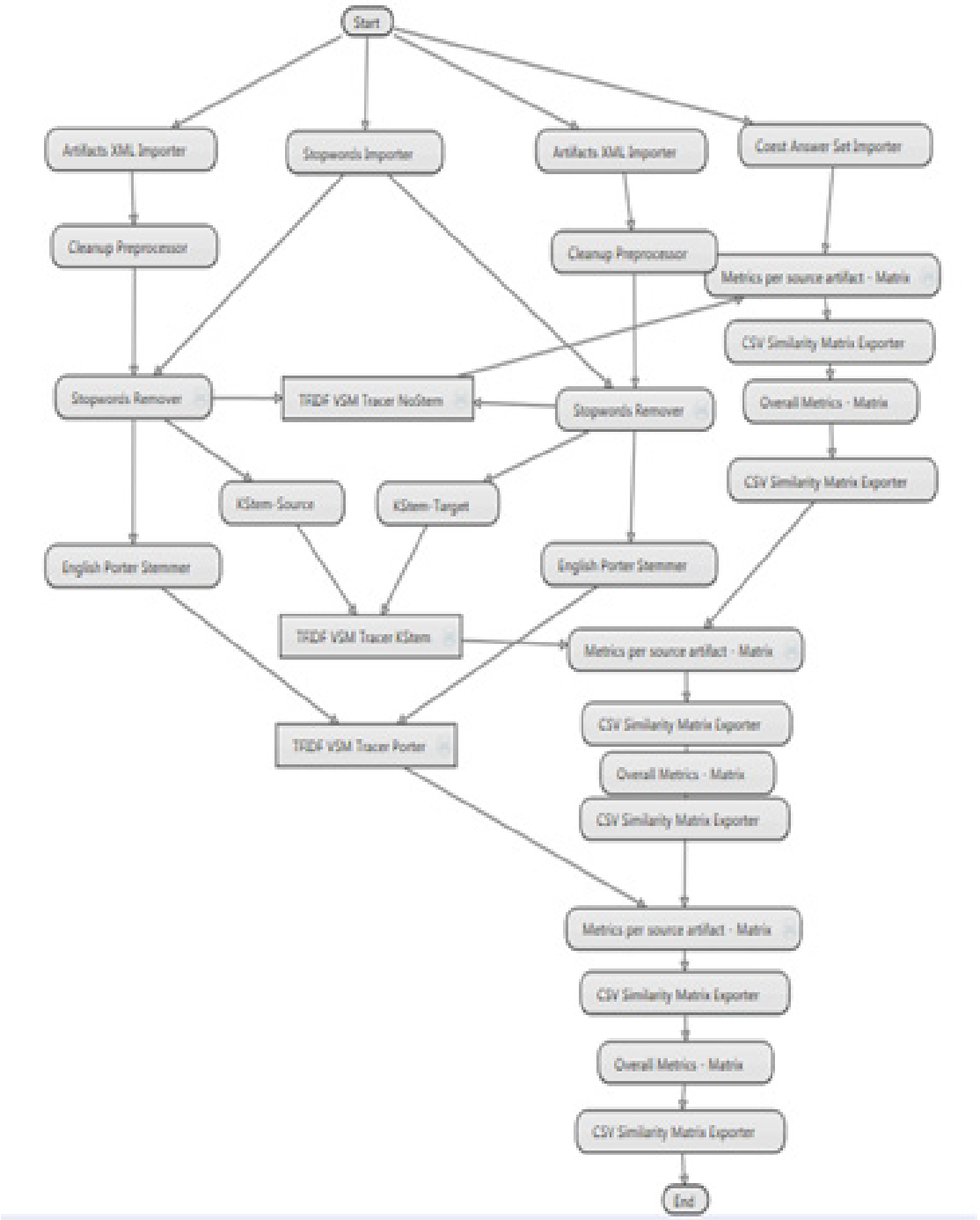

Fig. 1. TraceLab experiment.