ILLOD Replication Package: An Open-Source Framework for Abbreviation-Expansion Pair Detection and Term Consolidation in Requirements

Hussein Hasso
Fraunhofer FKIE
Wachtberg, Germany
hussein.hasso@fkie.fraunhofer.de

Katharina Großer
University of Koblenz
Koblenz, Germany
grosser@uni-koblenz.de
0000-0003-4532-0270

Iliass Aymaz
Fraunhofer FKIE
Wachtberg, Germany
iliass.aymaz@fkie.fraunhofer.de

Hanna Geppert
Fraunhofer FKIE
Wachtberg, Germany
hanna.geppert@fkie.fraunhofer.de

Jan Jürjens
University of Koblenz
Koblenz, Germany
juerjens@uni-koblenz.de
Fraunhofer ISST
Dortmund, Germany
0000-0002-8938-0470

***Abstract*—ILLOD is a tool for detecting abbreviation-expansion pairs (AEPs) in requirement sets. It utilizes syntactic features such as Initial Letters, term Lengths, Order, and Distribution of characters to determine if a term is a potential long form to a given abbreviation. The artifact bundles all source code and data resources to replicate evaluation results presented for ILLOD in two research papers published at the REFSQ2022 Conference and in the Information and Software Technology (IST) journal. In addition, ILLOD can be used to detect AEPs, perform abbreviation detection, and the input data-set can be used for further research in requirements engineering or other related fields. The repository is organized into different directories containing data, Python sources, and notebooks for experiments and evaluations. Detailed instructions are provided to load and use the tool on a local system, and the results generated by ILLOD are stored in output files. The tool demonstrates its effectiveness in detecting AEPs and consolidating glossary terms, and the evaluation results provide insights into the performance of different classifiers. The artifact repository is a valuable resource for researchers and practitioners in the field of requirements engineering and related areas.**

***Index Terms*—Requirements Engineering, Glossary Term Extraction, Abbreviation-Expansion Pair Detection, Synonym Detection, Requirement Data-Set, Abbreviation Data-Set.**

I. Introduction

Abbreviation-expansion pairs (AEPs) play a crucial role in requirement engineering, specifically in glossary term extraction and consolidation in requirement documents. Detecting and understanding these AEPs is essential for ensuring the quality and clarity of requirements. Hasso et al. [1] proposed an approach called ILLOD (Initial Letters, term Lengths, character Order, and Distribution) for identifying potential long forms to a given abbreviation by analyzing syntactic features of both terms, which extends the algorithm *findBestLongForm* presented by Schwartz and Hearst [2]. The approach has been further enhanced in subsequent work to improve the accuracy and effectiveness of AEP detection [3]. Here, the detection of abbreviations is refined, to better cope with heterogeneous abbreviation styles—particularly, lower-cased abbreviations and bi-grams of two consecutive abbreviations, while through recursive calls and a more sophisticated character distribution analysis, the extended ILLOD+ achieves higher recall and precision. These advancements provide a valuable tool for researchers and practitioners in requirements engineering.

The artifact presented in this paper is available on GitHub[1]and Zenodo[2] [4]. The artifact repository consists of Python sources, data files, and evaluation scripts that can be used to reproduce the results presented in both research papers by Hasso et al. [1], [3]. In particular, the extended experiment with a data-set of 1934 requirements from *PURE* [5] and more than 500 potential abbreviations presented in the journal paper published in *Information and Software Technology* (IST) [3] is made more accessible. In addition, it is facilitated to run the tool with custom data-sets to enable usage beyond the experiment reproduction. The repository provides a detailed README file with instructions on how to use the tool, reproduce the experiments, and load custom data-sets.

II. Overview and Structure of Artifact

The **ILLOD_REFSQ** and **ILLOD_IST** directories contain the original replication sources (Jupyter notebooks) and data for the respective research papers [1], [3].

The **MAIN** directory is the core of the presented artifact and contains the necessary files to run ILLOD, ILLOD+ and other

[1]https://github.com/AEPForGTE/ILLOD
[2]https://zenodo.org/record/8123990

AEP classifiers, as well as experiments for their evaluation. This directory depends on the ILLOD_IST directory. The sources in MAIN make three functions available:

1) **Create a test data-set:** Replace long-form terms in a requirement set with abbreviations from predefined lists to create a modified requirement set.
2) **Replicate Evaluation:** Execute and evaluate different classifiers for AEP detection, using the modified requirement set based on the analysis of created AEP groups (clusters that link a single abbreviation with all its identified potential expansions).
3) **Use ILLOD for custom data-set:** Optionally, extract AEP groups from a user-specified file.

The **input_data directory** contains the input files required to run the different AEP classifiers, including ILLOD and ILLOD+. It additionally contains the PURE [5] requirements data-set, a list of more than 500 abbreviations and their corresponding long forms, derived from noun-chunks of the PURE data-set, and optionally additional user-specified files for AEP detection on custom data-sets.

Two further lists of AEPs can be found in the ILLOD_REFSQ directory (based on [6] and noun-chunks extracted from the *PROMISE* [7] NFR data-set [8]).

After execution of main.py, placed in the MAIN directory, the **output_data directory** will contain new output files with the detected AEP groups, as well as the modified requirements data-set with injected abbreviations. The number of randomly chosen noun-chunks to exchange through an abbreviation can be specified between 0 and 400. A similar smaller and manually created data-set based on requirements from the *PROMISE* NFR data-set [8] can be found in the ILLOD_REFSQ directory. The **src directory** contains the Python source code with helper functions for abbreviation and term extraction and three further components needed for reproduction purposes or further customization:

The **generate_uncontrolled_abbreviations** component replaces terms with abbreviations in the input requirements. It reads input requirements from “pure_requirements.csv” and abbreviation-replacement data from “LF-SF_Pairs.csv”.

The **build_AEP_groups** component detects AEP groups by extracting abbreviations and terms from the requirements. It builds AEP groups by associating abbreviations with candidate long forms identified within the full term list by the different AEP classifiers (LD, JWS, DC, ILLOD, and the two ILLOD+ variants ILLOD_A & ILLOD_B) [3].

The **evaluate_ILLOD** component performs a ten-fold validation with respective 100 randomly injected abbreviations. It evaluates all mentioned AEP classifiers, counting true and false suggestions for each found abbreviation and their respective long forms on average over all ten passes.

In summary, the repository enables easy replication of experiments and utilization of ILLOD & ILLOD+. It provides data, source code, and evaluation results, along with instructions for reproducing the results, also with custom data-sets.

III. Future Use of the Artifact

The artifact provides technical means to automatically identify abbreviation-expansion pairs (AEPs) to facilitate the consolidation of terms. The artifact can be used for:

a) Reproducing experiments from [1] and [3].
b) AEP detection using ILLOD(+).
c) Abbreviation detection algorithm implementation.
d) Accessing and utilizing the provided input datasets for further AEP research in requirements engineering or other application domains.

ILLOD+ serves as a benchmark for enhanced AEP detection approaches, and there is potential for improvement by adapting it to domain-specific patterns, supporting multiple languages, and enhancing initial abbreviation detection.

Integration of ILLOD+ into existing requirement engineering tools and workflows can enhance abbreviation identification and term consolidation, making the process more efficient and reliable for software engineers and requirement analysts.

In conclusion, ILLOD+ contributes to requirement engineering by facilitating term consolidation and abbreviation identification. With further enhancements and integration, ILLOD+ has the potential to become a widely adopted approach for improving the quality and unequivocal understanding of requirement documents.

Acknowledgment

We thank the volunteers who provided us with abbreviation examples for our experiments. Shout out to Jupyter :)

This work is supported by project EnTrust in the research initiative of the Ministry of Science and Health, Rhineland-Palatinate, Germany and European Space Agency’s (ESA) NPI program (No. 4000118174/16/NL/MH/GM).

References

[1] H. Hasso, K. Großer, I. Aymaz, H. Geppert, and J. Jürjens, “Abbreviation-expansion pair detection for glossary term extraction,” in *Requirements Engineering: Foundation for Software Quality*, V. Gervasi and A. Vogelsang, Eds. Springer International Publishing, 2022, pp. 63–78. [Online]. Available: https://doi.org/10.1007/978-3-030-98464-9_6

[2] A. S. Schwartz and M. A. Hearst, “A simple algorithm for identifying abbreviation definitions in biomedical text,” in *Biocomputing 2003*. World Scientific, 2002, pp. 451–462. [Online]. Available: https://doi.org/10.1142/9789812776303_0042

[3] H. Hasso, K. Großer, I. Aymaz, H. Geppert, and J. Jürjens, “Enhanced abbreviation–expansion pair detection for glossary term extraction,” *Information and Software Technology*, vol. 159, p. 107203, July 2023. [Online]. Available: https://doi.org/10.1016/j.infsof.2023.107203

[4] H. Hasso, K. Großer, I. Aymaz, H. Geppert, and J. Jürjens. ILLOD: A Tool for Abbreviation-Expansion Pair Detection in Natural Language Requirements. [Online]. Available: https://doi.org/10.5281/zenodo.8020980

[5] A. Ferrari, G. O. Spagnolo, and S. Gnesi, “Pure: A dataset of public requirements documents,” in *25th IEEE International Requirements Engineering Conference (RE’17)*, 2017, pp. 502–505. [Online]. Available: https://doi.org/10.1109/RE.2017.29

[6] Computer Hope, “Computer acronyms and abbreviations,” 2021, visited on 10/16/2021. [Online]. Available: https://www.computerhope.com/jargon/acronyms.htm

[7] J. Sayyad Shirabad and T. Menzies, “PROMISE software engineering repository,” School of Information Technology and Engineering, University of Ottawa, Canada, 2005. [Online]. Available: http://promise.site.uottawa.ca/SERepository/

[8] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, “Automated classification of non-functional requirements,” *Requirements Engineering*, vol. 12, no. 2, pp. 103–120, 2007. [Online]. Available: http://ctp.di.fct.unl.pt/RE2017/downloads/datasets/nfr.arff