Automating Software Traceability Link Recovery and Maintenance using Word Embeddings within a Shallow Neural Network

Marlan McInnes-Taylor, Faculty Mentor: Dr. Chris Mills
Florida State University Department of Computer Science

Abstract

In addition to code, software systems contain a multitude of files documenting features, known issues, legal requirements, etc. Software traceability is the ability to link related documents from these various sets through a process called *traceability link recovery*. While previous research has shown that established software traceability improves the quality of projects and makes them easier to maintain, establishing software traceability is often a manual, arduous, and error prone task. This research explores automating the process of software traceability link recovery using established techniques in machine learning. The results demonstrate that even with minimally tuned hyperparameters a shallow neural network can effectively predict which text-based artifacts within a software system are related to one another.

Introduction

Previous studies have shown that access to information explaining relationships between artifacts in a system leads to higher quality software and lower bug counts. This is largely credited to such information improving common software engineering tasks such as:

- program comprehension
- concept and bug localization
- defect prediction

The acquisition of traceability information is difficult and typically an afterthought during system construction. Consequently, hundreds or thousands of man hours can be spent after initial development manually inferring relationships between artifacts that are constantly in flux as the system changes during development and maintenance.

To address this situation, previous studies have attempted to either completely or partially automate the process of establishing traceability links between system components. In this work, we continue that research agenda by using neural networks to model *potential* links between text-based software artifacts and predict which are valid (i.e. two related documents) and which are invalid (i.e. two unrelated documents).

Materials and Methods

The training and test data were comprised of six datasets commonly used to validate approaches for automating traceability link recovery: Albergate, eAnci, eTour, iTrust, MODIS, and SMOS. The table below shows a breakdown of the data by project. Each of the projects involved in our evaluation is written in either Java or C++. The code used for parsing and cleaning the software artifacts is written in Python. TensorFlow 2.0 was used to train and evaluate the neural networks.

Summary of Datasets	
Total Artifacts	755
Potential Links	22346
Valid Links	9.46%
Invalid Links	90.54%

DATASETS USED IN THE EVALUATION

System	Total Artifacts	Invalid Links	Valid Links	Artifact Types *
Albergate	72	882	53 (5.67%)	UC, CC
eAnci	194	7091	554 (7.24%)	UC, CC
eTour	174	6363	365 (5.43%)	UC, CC
iTrust	80	1493	58 (3.74%)	UC, CC
MODIS	68	890	41 (4.40%)	HighR, LowR
SMOS	167	3512	1044 (22.91%)	UC, CC

* HighR = High-level Requirements, LowR = Low-level Requirements, UC = Use Cases, CC = Code Classes

Preprocessing

Document text was first tokenized then cleaned by removing stopwords, whitespace, punctuation, and purely numeric tokens. A Porter Stemmer was applied to all remaining terms. Once cleaned, we created *metadocuments* by concatenating all possible (order invariant) pairs of documents such that the documents in a pair belong to different sets of artifacts. Each metadocument contains data from a unique pair of potentially related documents in the system, and was labeled as either a valid link between two related documents or an invalid link between two unrelated ones as specified in each dataset's oracle file. As shown in the Datasets Table, class imbalance was present in all datasets. To mitigate negative effects of class imbalance on model performance, each dataset was balanced using the Synthetic Minority Over-sampling Technique (SMOTE), resulting in an equal number of valid and invalid links within each dataset.

Training and Validation

A shallow Tensorflow model was implemented to classify unlabeled metadocuments as valid or invalid. The model's first two layers perform text vectorization and transform the vectors using word embeddings. These vectors are then pooled before being passed into a 16 node dense layer followed by the output layer. We performed 50 trials of 10 fold cross validation with 15 epochs per fold on each dataset. Shuffled stratification was used to sample the dataset in each fold to minimize selection bias.

Results

System	Loss	Accuracy	Recall	Precision	F1Score
Albergate	0.28	0.95	0.91	0.98	0.94
eAnci	0.11	0.96	0.94	0.98	0.96
eTour	0.13	0.96	0.93	0.99	0.96
iTrust	0.17	0.97	0.95	0.99	0.97
MODIS	0.29	0.93	0.94	0.92	0.93
SMOS	0.33	0.87	0.75	0.98	0.85

Conclusions

Based on the results, it is clear that word embeddings used within a shallow network can successfully perform metadocument classification for the systems under study. Note that here we report recall, precision, and F1 score in addition to accuracy. This is because accuracy can be artificially inflated in imbalanced data. For example, in a dataset of 100 samples with 1 “false” label, a machine that always predicts “true” has high accuracy, but it's low recall illustrates the machines impracticality. These results act as a proof on concept, and first step towards creating a classifier which is useful outside of a purely research context.

Future Work

Our future work will focus on improving this model by further optimizing its hyperparameters and evaluating the model's generalizability. Initially, we plan to perform a series of experiments to identify the minimum data requirements using supportive techniques such as Active Learning and cross training. Furthermore, we will investigate how deepening the model’s architecture impacts performance in terms of our results metrics.

References

Please refer to submitted paper.