IDNM 680: 2017 Data Challenge Event - Synopsis & Review

1st Della-Giustina, James
Department of Mathematics
Towson University
Towson, Maryland
jdella12@students.towson.edu

***Abstract*—The purpose of this paper is to examine and summarize results of seven different papers submitted at the 25th IEEE International Requirements Engineering Conference. Each of the papers focused on applying different natural language processing (NLP) and machine learning techniques on up to two provided datasets for the purpose of extracting software requirements.**

***Index Terms*—requirements engineering, natural language processing, machine learning, deep learning, software lifecycle**

I. Introduction

The 25th IEEE International Requirements Engineering Conference, was a conference held in Lisbon, Portugal in early September 2017. Requirement engineering is a discipline within software engineering that emphasizes the process of collecting, defining, analyzing, organizing, and validating the requirements for a software system or product, from the beginning of the development phase to the end of the software lifecycle [1]. This process enables both companies and end-users to actively participate in the maintenance, development, and evolution of a specific piece of software, by providing critical feedback, insights, and desired functionalities.

At this conference, seven papers were presented that were focused on the analysis and extraction of software requirements from user feedback/reviews using a variety of different techniques. There is quite often a disconnect between the desires of the end user, the companies who finance such endeavors, and the developers who write and maintain the final product. These submitted papers aimed to answer whether automation through the machine learning, sentiment analysis, or data mining could address this disconnect.

In this brief survey, we will explore the methodologies of each; highlighting novel approaches and discussing potential improvements. We will not divulge into discussing which model performed best, as all testing validation (if performed) was done by the authors themselves, and therefore may be impartial or biased.

II. The IlmSeven Dataset

In their paper "The IlmSeven Dataset" [2], the authors explore the use of open source software projects as a resource for research into requirement engineering and traceability. They argue that such projects offer a wealth of benefits, including being well-documented, having a large community of developers, and being generally sizable.

To justify this claim, they collected data from seven open source software projects: Derby, Drools, Groovy, Infinispan, Maven, Pig, and Seam2. Data was sourced from the projects' issue trackers, source code repositories, and mailing lists, and then stored in a relational database consisting of four tables: issues, reports, change sets, and links. These tables were used to track bugs and feature requests, document testing results, track changes to the source code, and connect artifacts to each other.

Using this database, the authors studied the traceability of requirements in the seven projects and found that while the traceability was generally good, there were some areas where improvements could be made. Specifically, some requirements were not linked to any issues or reports, and some issues and reports were not linked to any change sets. The authors suggested that future research could focus on enhancing traceability in open source software projects.

Overall, the authors concluded that the IlmSeven dataset is a valuable resource for research in requirements engineering and traceability. They also recommended that future research continue to explore ways to improve the traceability of requirements in open source software projects.

III. Toward Automating Crowd RE

In "Toward Automating Crowd RE" [3] the authors examine feedback from a large amount, or 'crowd', of people for requirements engineering. Crowd RE aims to enable large-scale user participation in RE tasks, such as identifying and prioritizing requirements, conflict analysis, and negotiation. However, relying solely on human effort can be expensive, and Crowd RE can benefit from automated techniques that process the large amounts of unstructured and noisy data produced by the crowd to discover valuable insights.

However, finding suitable datasets for training and testing automated techniques is challenging. The authors present the smart home requirements' dataset, which they constructed by involving 609 Amazon Mechanical Turk users and collected in two phases. In the first phase, they asked crowd workers to produce requirements for a smart home, while in the second phase, other workers rated the requirements gathered in the first phase for their clarity, usefulness, and novelty. They also collected information on worker demographics, personality traits, and creative potentials through surveys conducted before

and after the research phase. While they did not explicitly correlate any of the collected personal information with the quality or number of requested requirements, the data for this analysis is available.

They also identify opportunities for developing automated Crowd RE techniques based on their dataset, including clustering to process similar user stories and making data acquisition more efficient. They pose five key questions at the end of the paper for future research.

IV. What works better? A study of classifying requirements

In "What works better? A study of classifying requirements" [4], the authors look into the automation of classifying software requirements as functional (FR) or non-functional(NFR). Because engineers, stakeholders, and end users may all use different language to describe the same requirement, automation can prove to be difficult. The dataset chosen was from the OpenScienceter-PROMISE repository and consisted of 625 requirements, split into 255 FRs and 370 NFRs - subsequently split into eleven sub-categories.

The paper first examines the effectiveness of decision tree learning models in classifying FR and NFRs, and assess the impact of data normalization for classification performance. To achieve data normalization, the authors implemented various procedures, including assigning part-of-speech tags to individual words in a requirement, replacing specific software and user mentions with generic terms such as PRODUCT and USER, standardizing time formats across all data points, and manually identifying significant words or phrases in a limited subset of NFRs to develop rules for the model in feature identification. After performing these steps, their approach involved extracting features such as the number of adjectives, adverbs, cardinals, and number of degree of adjective/adverbs, and then training a decision tree model using 10-fold cross validation. They achieved a classification accuracy of 89.92% for the unprocessed dataset and 94.40% for the normalized version. Although 4.5% accuracy increase is non-trivial, the amount of preprocessing needed to be performed indicates that perhaps less structured datasets would have benefitted more than the one used.

The study evaluates the performance of various machine learning models in classifying NFRs into sub-categories, including topic modeling with Latent Dirichlet Allocation and Biterm Topic Model, clustering with hierarchical, k-means, and a hybrid approach, and naïve Bayes classification. The authors found that normalizing the dataset significantly improved the efficiency of topic modeling algorithms and naïve Bayes, which yielded the best performance among the models tested. The paper concludes by discussing the limitations of the dataset.

V. PURE: a dataset of public requirements documents

In [5], the authors present a dataset called PURE (Public Requirements dataset) consisting of 79 requirement documents collected from the web (from public companies and universities) and manually labeled based on number of pages, document name, etc., for use in natural language processing exploration. Additionally, they establish a standardized XML schema file for uniform document representation, of which 12 of the documents were converted into. After noting problems with the data set such as an imbalance of technical vocabulary, writing styles, and ratio of documents from public companies and universities, the authors pulled natural language statistics from the dataset compared to a well known 'Brown' corpus, consisting of approximately 500 examples of English-language texts. After comparing the results to the Brown corpus, the authors noted that 62% of 'lexical' words used in the PURE dataset are absent from the Brown corpus, indicating a high level of technical and specific vocabulary.

The authors conclude the paper by noting that significant work still needs to be performed (i.e. annotations) in order to generalize this dataset for use in many NLP tasks as well as the conversion of the remaining documents to the unified XML file schema.

VI. RE Data challenge: Requirements Identification with Word2Vec and TensorFlow

In "RE Data challenge: Requirements Identification with Word2Vec and TensorFlow" [6], the authors apply two Google deep learning tools on requirements engineering documents; Word2Vec and TensorFlow. In a very basic explanation, Word2Vec is a natural language processing tool that converts a string of words to a vector representation that has been learned from an enormous corpus of text. Word2Vec has proven to be a valuable tool in the context of text classification and language translation to name just a few. On the other hand, TensorFlow is an open source python library that focuses on easily implementing machine learning models. More specifically, the authors used this library to build convolutional neural networks which, although originally designed for image classification, have proved to be useful in natural language processing.

The first dataset that they explored focused on extracting requirements that were labeled as security related, as well as another data set that labeled requirements as functional requirements (FR) or non-functional requirements (NFR). Following previous studies on the first data set, the authors recreated known results using a naïve Bayes method as well as adopting the approach to the second dataset, both of which served as a baseline against their experimental models. Their model using Word2Vec performed significantly better than their baseline, with an almost 15% increase in the precision score. However, for their TensorFlow model without Word2Vec, the results fell short of the baseline, performing worse in every metric except for a slight gain in precision. For the second dataset, there was again an improvement in every metric when using Word2Vec. However, the model that did not use Word2Vec performed almost identically to their baseline naïve Bayes method.

The authors conclude the paper by noting the model improvement that Word2Vec offers, especially when considering

requirement documents that are not rich in text. They note that future work includes testing these conclusions against other RE datasets, as well as revisiting the second dataset and further classifying the NFRs into their distinct classes.

VII. Automatically Classifying Functional and Non-Functional Requirements Using Supervised Machine Learning

In [7], the authors applied support vector machines (SVM) to a RE dataset with labels for functional requirements (FR) and non-functional requirements (NFR). They sought to answer three questions, how well could their model distinguish FR and NFRs, if a requirement is NFR, how well could they classify it into the top four NFR categories, and could their model be improved by training on an additional dataset.

After balancing the dataset so that there was a roughly even proportion of FR and NFR samples to address the class imbalance, they scored the importance of each feature using decision tree models such as extra tree and random forest and two additional models. Then, focusing solely on NFRs, they found the top four most frequent classes to be usability, security, operational, and performance. Finally, they set up various different training sets by taking the dataset and under/over sampling for NFR classification.

To answer the first question, how well could their model distinguish FR and NFRs, the researchers found that automatic feature detection of k features drastically lowered the accuracy of the models for $k < 400$. Overall, the best scores were obtained without any automatic feature detection and were: FRs - .92 precision .93 recall, .93 F1 and NFRs - .93 precision, .92 recall, .92 F1.

For the second question of a classifying NFRs based on the top 4 most frequent classes, the authors ranked the binary classifier of the performance NFR to have the highest precision, recall and F1 scores, which was surprisingly achieved without any automatic feature detection. The next highest ranked binary classifier was security, then usability, and finally operational, which performed far worse than the others. This illustrates the class imbalance of the NFR subset of the dataset, since the fourth most frequent NFR class was only able to obtain .72 precision, .75 recall, and .73 F1 score.

For the last question, the authors found that randomly under/oversampling the dataset and introducing additional data points from the UC dataset did not improve the accuracy metrics. Additionally, oversampling the NFR dataset along with the UC dataset did not improve the classification accuracy.

REFERENCES

[1] O. Elgabry, "Requirements engineering - introduction (part 1)," Jan 2022. [Online]. Available: https://medium.com/omarelgabrys-blog/requirements-engineering-introduction-part-1-6d49001526d3

[2] M. Rath, P. Rempel, and P. Mäder, "The ilmseven dataset," in *25th IEEE International Requirements Engineering Conference, RE 2017, Lisbon, Portugal, September 4-8, 2017*, A. Moreira, J. Araújo, J. Hayes, and B. Paech, Eds. IEEE Computer Society, 2017, pp. 516–519. [Online]. Available: https://doi.org/10.1109/RE.2017.18

[3] P. K. Murukannaiah, N. Ajmeri, and M. P. Singh, "Toward automating crowd RE," in *25th IEEE International Requirements Engineering Conference, RE 2017, Lisbon, Portugal, September 4-8, 2017*, A. Moreira, J. Araújo, J. Hayes, and B. Paech, Eds. IEEE Computer Society, 2017, pp. 512–515. [Online]. Available: https://doi.org/10.1109/RE.2017.74

[4] Z. S. H. Abad, O. Karras, P. Ghazi, M. Glinz, G. Ruhe, and K. Schneider, "What works better? A study of classifying requirements," in *25th IEEE International Requirements Engineering Conference, RE 2017, Lisbon, Portugal, September 4-8, 2017*, A. Moreira, J. Araújo, J. Hayes, and B. Paech, Eds. IEEE Computer Society, 2017, pp. 496–501. [Online]. Available: https://doi.org/10.1109/RE.2017.36

[5] A. Ferrari, G. O. Spagnolo, and S. Gnesi, "PURE: A dataset of public requirements documents," in *25th IEEE International Requirements Engineering Conference, RE 2017, Lisbon, Portugal, September 4-8, 2017*, A. Moreira, J. Araújo, J. Hayes, and B. Paech, Eds. IEEE Computer Society, 2017, pp. 502–505. [Online]. Available: https://doi.org/10.1109/RE.2017.29

[6] A. Dekhtyar and V. Fong, "RE data challenge: Requirements identification with word2vec and tensorflow," in *25th IEEE International Requirements Engineering Conference, RE 2017, Lisbon, Portugal, September 4-8, 2017*, A. Moreira, J. Araújo, J. Hayes, and B. Paech, Eds. IEEE Computer Society, 2017, pp. 484–489. [Online]. Available: https://doi.org/10.1109/RE.2017.26

[7] Z. Kurtanovic and W. Maalej, "Automatically classifying functional and non-functional requirements using supervised machine learning," in *25th IEEE International Requirements Engineering Conference, RE 2017, Lisbon, Portugal, September 4-8, 2017*, A. Moreira, J. Araújo, J. Hayes, and B. Paech, Eds. IEEE Computer Society, 2017, pp. 490–495. [Online]. Available: https://doi.org/10.1109/RE.2017.82