International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Enhancing Software Requirements Classification: Integrating Recurrent Neural Networks and Natural Language Processing for Managing Structural Complexity

Justine Nakirijja[1], Abid Yahya[2,4], Ravi Samikannu[3], and Lory Liza D.Bulay-og[4]

Submitted:12/03/2024 **Revised**: 27/04/2024 **Accepted**: 04/05/2024

Abstract: The inherent complexity of software requirements poses significant challenges in project planning and quality assurance. This research addresses these challenges by enhancing the classification of software requirements. It explores the dynamic relationships between requirements and their associated acceptance criteria through advanced deep-learning methods. The primary objective is to improve the accuracy and efficiency of requirements classification, thereby contributing to more effective project management and development processes. We propose a novel approach using a Recurrent Neural Network for Requirement Engineering (RNNRE) model. This model integrates natural language processing to analyze and process complex, multilevel requirements' temporal and functional dynamics. Our methodology is rigorously tested on the Baseline EMR, a comprehensive real-world dataset, to assess the model's effectiveness and accuracy in classifying software requirements. Results: The study reveals that finer granularity in requirement conditions substantially influences classification outcomes, impacting the precision of acceptance statements. The RNNRE model demonstrates robust performance, achieving an accuracy of 82.6%, a recall rate of 80%, and a precision of 100%. These results notably surpass the performance of several benchmarked state-of-the-art models, showcasing the model's effectiveness in handling complex requirement scenarios. The RNNRE model marks a significant advancement in refining the requirements engineering process, particularly for intricate and multileveled requirements. This research demonstrates the practical application of deep learning in requirements classification. It contributes valuable insights to the field, enhancing the understanding and methodology of managing structural complexity in software requirements engineering.

***Keywords**: Requirements Engineering, Deep Learning, Recurrent Neural Network, Requirements Classification, Structural Complexity in Software, NLP.*

Introduction

The success of software projects is intricately linked to satisfying customer expectations, a determination often formalized through acceptance testing at project completion [29]. However, the seeds of success are sown much earlier in the project lifecycle, specifically during the pre-development acceptance planning conducted jointly by the project team and the customer. This process establishes the acceptance conditions (AC), which define the detailed quality benchmarks for each requirement and shape the eventual user perception and satisfaction [25].

This relationship between requirements and their acceptance conditions is complex, as ACs can apply variably to single or multiple needs. Such variability necessitates meticulous trackingfrom the initial planning stages to the final sign-off. The project's ultimate goals, aligning withuser satisfaction and product requirements, hinge on the quality expectations set out by theseACs [14]. Moreover, the granularity of ACs can vary significantly, leading to potential issues with class imbalance and the challenge of accurately classifying and mapping requirements. This point becomes increasingly complex with advancing technology and evolving user expectations. Current methodologies for handling complex systems in software engineering address various aspects of this complexity [8, 9, 10, 26]. Yet, there is a notable gap in addressing the intrinsic and extrinsic complexities that originate from the nuanced requirements structure in alignment withACs. This gap is particularly evident in agile project environments, where the rapid deliveryof high-quality software is paramount [23]. They identified critical reasons for project failures,as illustrated by poor user input at 13%, incomplete requirements at 12%, changing requirements at 12%, poor starting at 6%, inadequate technical skills at 7%, and others at 50%. As such,there is a vital need for innovative approaches to adeptly manage the structural complexities within requirements sets. The

[1]*Department of Data Science, Networks and Artificial Intelligence, Kyambogo University, Kampala, Uganda;* Corresponding Author
[2,3]*Department of Electrical and Communications Systems Engineering, Botswana University of Science and Technology, Palapye, Botswana;*
[3]*Department of Electronics and Communication Engineering, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai, India;*
[4]*University of Science and Technology of Southern Philippines (USTP), Alubijid (Main)Campus, Philippines.*

researcher [22] noted that "70% of software projects fail due to poor requirements with an associated rework spend just north of $45 billion annually".

In this paper, we present a novel approach that leverages the robust capabilities of deep learning to enhance the requirements of the engineering process. By focusing on the alignment of ACs with their corresponding requirements, our research offers a method to navigate the complexitypresented by varied AC granularity levels. We introduce a model that not only aids in classifying requirements but also in recognizing their acceptance for project planning purposes. This work, therefore, contributes to the ongoing pursuit of sophisticated models that can be seamlessly integrated within the requirements engineering framework to ensure the delivery of quality software products that meet or exceed stakeholder expectations.

In this study, we regard this as structural complexity within a requirement dataset that has to be controlled to mitigate the risk of dissatisfaction with the end product. We inferthat the continued lack of a mechanism to track and manage this structural complexity within requirements impacts the quality of a condition and the downstream processes immensely.

Literature Review

Natural Language Processing

Natural Language Processing (NLP) and Information Retrieval (IR) techniques have been applied to address some of these challenges, including improving information (in this case, requirements) tracing. NLP applies to "Shallow Knowledge" from requirement text, tracing therelationship among conditions [31].

Natural language processing is a branch of artificial intelligence that enables computers to understand, manipulate, and interpret human language [28]. A common component of NLPis artificial intelligence text analysis, which defines extracting information from extensive text data, also known as text mining [21]. Text mining converts unstructured data into structured data for machine learning [28]. With machine learning, Natural language processing (NLP) involves manually identifying critical text sections or labeling readers. It can locate sentiments,speech parts, proper nouns, and text in images, PDFs, and other documents.

NLP Techniques

Natural Language Processing (NLP) uses two techniques, syntactic analytics and semantic analysis, to help computers understand text.

Syntactic Analysis or Parsing examines text using basic grammatical principles to detect sentence structure, word arrangement, and how they connect. The major subtasks for this technique include:

Tokenization: Involves dividing a text into smaller pieces called tokens (which may be phrases or words) to simplify material handling.

Part of Speech Tagging: Labels tokens like verbs, adverbs, adjectives, nouns, etc. This helps determine the meaning of words (for example, the term "book" refers to different objects, whether employed as a verb or a noun).

Lemmatization and Stemming reduce inflected phrases and derivationally related forms of words to their base form to facilitate analysis.

Stop-word Removal: Often eliminates words that do not contribute value, such as "I," "they," "have," and others.

The semantic analysis relies on capturing the meaning of the text. It utilizes the syntaxtree (i.e., structures such as phrases, clauses, sentences, and texts) generated in the parsing process to interpret language-independent meanings. Initially, it will examine the significanceof each word (lexical semantics) [28]. Then, the arrangement of words and what they signify is reviewed in context. The primary task of semantic analysis includes determining the meaningof the given sentence and representing that meaning in an appropriate form.

However, these generally require human effort to analyze and create requirements-based features. Subsequently, machine learning approaches, particularly classification, show promise. Several ML-related techniques have been developed to manage the identification and classification of non-functional requirements (NFRs) in requirements documents [6]. [2] the most used supervised learning algorithms in the literature are Support Vector Machine, naive Bayes, Decision Tree, K-nearest Neighbor, and Random Forest.

Related Work

Several studies have been proposed to handle challenges that lead to the misclassification of textual requirements, as critical studies presented in Table 1.

Table 1.

Research focus	Techniques used	Data used	Performance/ Results	Limitations
Ontologies to classify the requirements in the RE context (Alrmuaih, Mirza and Alsalamah, 2020)	Requirements Classification ontology (RCO)	Not Applicable	A hybrid approach with several artificial intelligence techniques that RCO can be used to automate the requirements classification process	Not stated
SLR on automatic classification of software requirements (Baqais and Alshayeb, 2020)	Mostly used NLP tools, WEKA in-built tools, SVM, MNB, LR, K-NN, J48, CNN	Not Mentioned	Not Applicable	More contribution is still needed in this research
Text feature extraction techniques, with Non-functional requirements (Canedo and Mendes, 2020)	Comparison of (BoW) vs. (TF-IDF) vs. (CHI2), Algorithms: (LR), (SVM), (MNB) and (KNN)	Labeled data	Binary classification: F-Measure 91, general classification 0.78 and 0.74 in NF classification	imbalance and smaller dataset influence the classification performance results in a machine learning environment

Requirements classification (RC) using analysis of textual natural language is the trend tosolve numerous software engineering challenges. As indicated in Table 1, the most recent study points to using artificial intelligence as a suitable alternative. Even the systematic literaturereview shows that several studies on classifying requirements using machine learning and deep learning, such as [3] applied convolutional neural network CNN, still recommend furtherstudies with other deep learning techniques. Some studies, as in [4], focused on RC modelingusing a small dataset with class imbalance but observed that this impacts the model performance [7, 17]—the researchers in [1] compared RNN and CNN performance on a small dataset. In [11],BERT's deep learning technique was enhanced into NoBERT, with transfer learning focusedon binary and multiclass classification. Other researchers [20] used LSTM and GRU with thesmall and class-imbalanced dataset. The work in [16] presented points to guide the selection oftechniques that can be applied in constructing classification models. While this all is promisingwork for RC, the interest of this study was to evaluate RNN with large, multileveled networks.RNN models are widely used in classification studies such as [1, 30]. RNNs are neural networks specializing in processing sequences and are often used in Natural Language Processing (NLP) tasks because of their effectiveness in handling text. Most vanilla neural nets andConvolutional Neural Networks (CNNs) usually work with predetermined sizes, and they take fixed-size inputs and produce fixed-size outputs. With this flexibility that allows for variable-length sequences (as both inputs and outputs), we applied an RNN's many-to-one (M1RNN), explicitly using RNN's variant, the LSTM, to examine the relationships between requirements and the associated ACs in a many-to-one connection.

Long Short-Term Memory

Long-short-term memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. Unlike traditional RNNs, which may struggle with long-range dependencies due to vanishing or exploding gradients, LSTMsare designed to retain information efficiently for extended periods. They achieve this througha complex architecture comprising memory cells and various gates: forget, input, and output gates, each playing a distinct

role in managing the flow of information.

Forget Gate

The forget gate, also known as the remember vector, determines how much previous informationis retained for future steps. It uses a sigmoid function to output a value between 0 and 1, indicating the proportion of each component of the cell state to be forgotten.

Input Gate

The input gate is responsible for updating the cell state. It decides the degree of importancefor new information to be added to the current state, using a sigmoid function to filter values and a tanh function to scale them within a range of -1 to 1.

Output Gate

The output gate defines the next hidden state, which carries information about previous inputs. The current state and the last hidden state are fed into a sigmoid function, and the new cell state is processed through a tanh function. Their product determines the information to be included in the hidden form.

LSTMs are particularly well-suited for handling multivariate time-series data, as they canmap multiple inputs to a single output, manage varying numbers of time steps, and handle variable-length inputs. This flexibility allows them to capture temporal dependencies effectively.The mathematical model of an LSTM can be expressed through a series of equations representing the operations within the memory cells and gates:

$$\nabla t_n = t_{n+1} - t_n \quad (1)$$

Given that x_{tn} is a row vector with P dimensions, the prediction model can be described asfollows:

$$\hat{y}_{tN+1} = f(xt_1, xt_2 \ldots, at_N, \Delta t_1 \Delta t_2 \ldots, \Delta t_N) \quad (2)$$

The behavior of the LSTM at any time step t is governed by:

$$h_t = \sigma(W_h x_t + U_h h_{t-1} + b_h) \quad (3)$$

$$y_t = \sigma(W_y h_t + b_y) \quad (4)$$

Where σ represents the activation function, and W_h, U_h, b_h, W_y, b_y are the weights and biases of the network. These equations are iteratively applied to process sequences, and predictions are made based on learned temporal dependencies.

Methodology: Recurrent Neural Networks for Requirement Engineering (RNNRE)

Research Context of RNN

Predictive classification modeling is essential in predicting a future value based on historicaldata at the preceding time step. This process necessitates the preparation of input and output pairs given the time series data. Formally, time series classification is defined as follows:

Definition 1: A univariate time series $X = [X_1, X_2, \ldots, X_n]$ is an ordered set of acceptance conditions (ACs). The length of X, denoted as |X|, equals the number of fundamental values N.

Definition 2: An M-dimensional multivariate time series (MTS), $[X_1, X_2, \ldots, X_m]$,, consists of M different univariate time series requirements (R), each with $X^i \in R^N$.

Definition 3: A dataset $D = \{(X_1, Y_1 1), (X_2, Y_2), \ldots, (X_N, Y_N)\}$ is a collection of pairs (X_i, Y_i), where X_i could either be a univariate or a multivariate time series AC, and Y_i is its corresponding one-hot label vector. For a dataset with K classes, the one-hot label vector Y_i is a K-dimensional vector where the j-th element is 1 if the class of X_i is j, and 0 otherwise.

Classification involving time series data is performed by training a classifier to learn from a dataset using a probability distribution and then using it to relate ACs with requirements [13].

Phase of RNNRE

The methodology for RNNRE is characterized by a process divided into the following fourphases:

1. Data pre-processing.
2. Word Vectorization and Labelling.
3. RNN model construction.
4. Model Training and Testing to align Requirements to ACs.

The Fig.1 shows the Representation of an alignment of requirement(S) to acceptance conditions (AC)

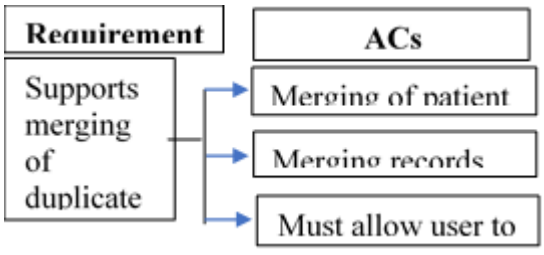

Fig.1. Representation of an alignment of requirement(S) to acceptance conditions (AC)

Procedural Flow

Building this classification model involves a four-phased process, as illustrated in Fig.2.

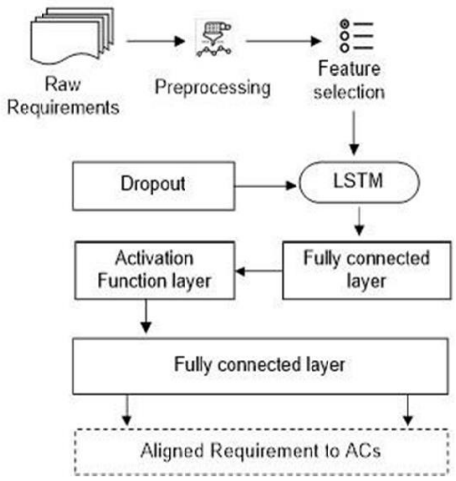

Fig.2. Procedural phases for Multileveled Requirements Classification

Phase 1: Data Pre-Processing

The initial step in this framework is cleaning and preprocessing the software requirement dataset. This stage is preceded by data acquisition and includes removing special characters, stop words, case-folding, lemmatization, and tokenization. The outcome is that the data is well-cleaned and ready for the learning environment.

Phase 2: Word Vectorization/Labeling

Word vectorization converts sentences into a computer-understandable format for deep-learning models that cannot process natural language. This process is crucial for enabling pattern recognition within the data. The text of the requirements is transformed into word vectors using the word2vec model, explicitly utilizing the skip-gram approach for feature extraction. In this context, word2vec maps a word to a vector v ∈ R, where R represents the set of real numbers. Thus, the transformation of a sentence can be represented by a matrix m ∈ $R^{n\times l}$, where n is the embedded vector size and l is the length of the sentence.

This study employed embeddings from language models, particularly ELMo, for word embedding [15]. ELMo considers the entire sentence context to assign each word a unique embedding and functions as a bidirectional RNN trained on a specific task to generate these embedding [15]. The resulting individual word vectors populate an embedding matrix, the dimensions of which are determined by the vocabulary size and the embedding extent.

Phase 3: RNN Model Construction

We utilize an LSTM network for the RNN model, composed of:

Input Layer: Prepares the model input with sequence length and embedding dimension.
Hidden Layer: Employs LSTM units with dropout to reduce overfitting.
Output Layer: Uses activation functions like softmax and sigmoid to output the finalclassification.

Phase 4: Model Training and Testing

The model is trained and tested using various training-testing split ratios. The performance is analyzed as the proportion of testing data varies.

Extension to GCN-RIA for Structural Analysis

Following the RNN model construction, we extend our approach with the Graph Convolutional Networks for Requirements Interdependency Analysis (GCN-RIA) to capture the structural dependencies among requirements. Given a graph G = (V, E) representing the interdependencies among software requirements:

Let V be the set of nodes (requirements) and E the set of edges (dependencies).

Each node $v_i \in V$ has a feature vector x_i.

The graph convolution operation at layer l is defined as:

$$H^{(l+1)} = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \quad (5)$$

$\hat{A} = A + I$ is the adjacency matrix with self-connections, D is the degree matrix, $H^{(l)}$ is the activations, and W (l) is the weights.

This approach captures the complex interdependencies in software requirements, enhancing the predictive accuracy of the RNNRE model.

Model Training and Testing

Following the structural analysis with GCN-RIA, we integrate the Transformer-based Multi- Aspect Requirements Analysis (TMARA) for a comprehensive linguistic analysis:

Consider a sequence of tokenized requirements R $R = \{r_1, r_2, \ldots, r_n\}$.

Embed each token into a high-dimensional space, obtaining embedding $E = \{e_1, e_2, \ldots, e_n\}$.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (6)$$

Employ the Transformer model's self-attention mechanism:

Q, K, V are query, key, and value matrices, and d_k is key dimensionality.

TMARA's integration allows the model to consider various linguistic aspects of software requirements, providing a nuanced understanding essential for

accurate classification.

Model Evaluation/ Performance Metrics

In evaluating predictive models, accuracy and loss are recommended as key performance metrics in the literature [12]. Platforms like Python provide built-in modules to support metrics suchas loss, score, accuracy, and utility functions for measuring classification performance. We considered recall and accuracy scores for binary classification, while for multi-class and multi-label sort, the F1 score is crucial. The metrics are defined as follows:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (7)$$

$$Recall = \frac{TP}{TP+FN} \quad (8)$$

$$Precision = \frac{TP}{TP+FP} \quad (9)$$

$$F1\ Score = \frac{2\times Precision\times Recall}{Precision+Recall} \quad (10)$$

Where T P is the actual positive rate, T N is the actual negative rate, FP is the false-positive rate, and FN is the false-negative rate for a classifier across all classes.

For deep learning models, we define a loss function, such as binary cross-entropy, to measure the accuracy metric for binary classification, multi-label classification, and many-to-one classification scenarios. This also applies to multi-class methods with the output layer configured with more than one unit and the activation function set to 'Softmax'.

Experimental Setup and Evaluation

We conducted two experiments for our analysis. The first experiment evaluated the proposed RNN model's performance against two datasets (Labeled and Unlabeled, EMR dataset), combined with an embedding layer for flexible word embedding in neural networks and text data. The experiments on the EMR dataset assessed complexity from the perspectives of Precision, Recall, and F1-score, varying the "Loss Function" and "Activation Function." The performance results are presented in Table 3.

Algorithmic Representation of Deep Learning Model Training and Evaluation

The subsequent algorithm delineates the process of training and evaluating a deep learning model for classification tasks. It outlines the initialization of pre-trained embedding, tokenization, sequence mapping, dataset partitioning, and the construction and training of a recurrent neural network (RNN) with a long short-term memory (LSTM) architecture.

Algorithm 1 Complex Deep Learning Classification Algorithm

1: **START**

2: $\Theta \leftarrow$ LoadPretrainedEmbeddings(V, L, d) ▷ V : vocabulary size, L: sequence length, d:embedding dimension

3: T ← Tokenize(T) ▷ T : Corpus of text

4: S ← MapToSequences(T)

5: D ← ∅ ▷ Data set

6: L ← ∅ ▷ Label set

7: *where*

8: D ← ExtractAcceptanceConditions(AC) ▷ AC: Acceptance conditions

9: L ← DefineRequirements(R) ▷ R: Requirements

10: $(D_{train}, D_{test}, L_{train}, L_{test}) \leftarrow$ PartitionDataSet(D, L)

11: Model ← InitializeRNNModel()

12: Model.Add(EmbeddingLayer(E, trainable=False)) ▷ Load embedding matrix E

13: Model.Add(LSTM(h)) ▷ Add LSTM layer with h hidden units

14: Model.Add(Dense(K, activation = σ)) ▷ K: Number of classes, σ: activation function

15: Model.Compile(loss = L, optimizer = O) ▷ L: Loss function, O: Optimizer

16: Model.Fit(D_{train}, L_{train})

17: Model.Evaluate(D_{test}, L_{test})

18: **END**

Datasets and Hyper-Parameter Tuning

Datasets: In our experimental framework, we concentrated on datasets with a lower-dimensionalfeature space, where the dimensionality of the feature set p is substantially lesser than the number of observations N, typically denoted by $p \ll N$. The dataset in question, the EMR requirements specification dataset, is enumerated in Table A within the appendix. This dataset encapsulates 108 individual requirements alongside their respective acceptance conditions (ACs). Each dataset instance represents a distinct input level or a time step within our model's context.

Hyper-Parameter and Training: The hyper-parameters selected for the experimental process include a minibatch size 10, an epoch count of 50, a learning rate fixed at 0.001, and a dropout probability set to 0.2. The embedding layer is instantiated with a max length of 16 and an embedding dimension of 32 units. Data preparation involved extracting text from CSV files, tokenizing this text, and creating sequences from the processed words.

The global parameters utilized for the Recurrent Neural Network Requirement engineering (RNNRE) model are Embedding Dimension,32; MaxLength,16; Neurons in OutputLayer,1; Optimizer, Adam; Activation Functions, Sigmoid and Softmax; Performance Functions, Binary cross-entropy and Categorical cross-entropy.

Tensor Flow, a Python-based API engineered by Google, was the foundation for constructing deep neural networks. This platform's intrinsic tools were instrumental in evaluating and refining model performance. Consistent with prevailing studies, we utilized a sigmoid activation function at the output layer for binary and multi-label classification tasks and a soft-max activation function for multi-class classification objectives [24]. Our experimental trials spanned both labeled and unlabeled datasets, with labeled data demanding the use of softmax due to multiple class outputs, in contrast to the unlabeled dataset, which necessitated the use of sigmoid for its singular output class.

Sequential Approach Model

The architecture of the proposed model is compiled with the loss function binary cross-entropy and employs the optimizer Adam. Our model architecture consists of three distinct layers:

An embedding layer transforms words into vector space using pre-trained word embeddings, enabling the model to understand the context of the words in the input data.

A dense, fully connected layer in which each neuron receives input from all neurons in the previous layer, thereby integrating signals across the network.

The output layer generates the final prediction for the given inputs. This layer employs the sigmoid activation function for binary and multi-label classification tasks to output independent probabilities for each label.

The optimizer Adam is used for hyper-parameter tuning because it efficiently handles sparse gradients on noisy problems. A dropout rate 0.2 is integrated into the network as a regularization technique to prevent overfitting. The activation function relu is utilized in all layers except the output layer due to its effectiveness in addressing the vanishing gradient problem. In contrast, the sigmoid activation is specifically chosen for the output layer in binary and multi-label classifications, facilitating the model to output probabilities independently for each label without constraint on the sum of these probabilities.

Result Analysis

Experiment Two: RNNRE Performance on EMR Datasets

The classification tasks were executed on both labeled and unlabeled datasets. Labeled datasetsencompass low-level requirements categorized into distinct classes based on their attributes, while unlabeled datasets lack predefined categorization.

A segment of the labeled dataset employed for classification using LSTM is exemplified in Table 2.

Table 2: Sample of Labeled Dataset for Prediction using LSTM

High level Requires (HI.R)	Low-level Requirements (I.I.R)	Category	Label
Search Patient	The user can search for a Patient by name or ID.	Search_Patient Details _	1

	The system shall display the patient's identity details.	Search_Patient Details	1
	The user can scan a barcoded ID number to find the Patient.	Input	2
Add Patient Details	The user can register a new patient with a unique ID.	Generation Report	3

Requirements Classification with Labeled Data

Classification of requirements was performed using a labeled dataset, which was structured to facilitate the assignment of samples to generated labels. The efficacy of this classification is evidenced by the outcomes depicted in Fig.3, which elucidates the model's accuracy and losswhen utilizing labeled data.

Fig.3. Training and validation accuracy and loss for labelled data classification

Our model achieved commendable training accuracy, surpassing 98%, suggesting a solidlearning capacity from the labeled dataset. However, it is imperative to note that the validation/test accuracy averaged around 50%. This disparity may indicate an overfitting scenario,wherein the model's predictions are highly accurate for the training data yet fail to generalize to unseen data. Further examination into recall and precision metrics revealed values approaching 100%, indicative of the model's proficiency in correctly classifying the positive cases.To gain deeper insights into the model's ability to align specific requirements with their corresponding predicted classes, we subjected the last ten entries of the unlabeled dataset—denoted as X_{new}—to the trained model. This subset, extracted from the dataset presented in the appendix (see Table 4), is a test case to evaluate the model's predictive capabilities in a real-world scenario. The predicted outcomes are given in Table 3.

Sample Predictions and Their Implications for Requirements Assessment

The LSTM model was tasked with classifying a set of previously unseen requirements (denoted as X_{new}) from the EMR system. These requirements range from diagnostics to laboratory orders and patient follow-up protocols. The model's predictions for these requirements aredelineated in Table 3.

Table 3.

Requirement No.	EMR Requirements	Predicted Category
95	The user can generate a list of patients...	Search Patient Details
93	Free text noncoded diagnoses should be avoided...	Generation Report
103	The system shall automatically generate Order IDs	Generation Report
97	The user can print the lost to follow-up report	Generation Report
100	The user can select Laboratory tests...	Generation Report

Practitioners can leverage these results to discern the overt and subtle requirements pivotal for meticulous project estimation [[19]]. Specifically, the model's ability to align needs with predicted categories facilitates planning and prioritization based on domain expertise. For instance,the clustering of requirements into Class 3 predominantly pertains to Clinical Documentation and Reporting, indicating a nuanced level of complexity associated with features related to clinical documentation.

This adept classification underscores the potential of the LSTM model in streamlining fea- ture implementation planning. Nonetheless, the results also highlight the necessity for a reason-able selection of a subset of unseen data, which is crucial for rigorously evaluating the predictive prowess of the model. The ensuing section elaborates on a scenario where the model undergoes evaluation using the entirety of the dataset.

Requirements Classification with Unlabeled Data

Methodology and Dataset

In this exploratory phase, we employed an unlabeled dataset comprising high-level requirements (HLR) and their corresponding acceptance conditions, denoted as low-level requirements (LLR). Our objective was to discern how an RNN model would categorize requirements in the absence of explicit labels. The dataset, composed of 108 samples, was the basis for gauging quality metrics such as accuracy, precision, and recall.

Challenges of Unlabeled Data

While accurate, manual labeling can be excessively labor-intensive and necessitate substantial domain expertise. Conversely, classifying unlabeled datasets streamlines the process but requires in-depth knowledge for effective hyperparameter tuning. The potential for overfitting presents asignificant challenge, mainly when dealing with limited datasets. Notwithstanding these issues, our model achieved promising results, with training accuracy surpassing 80% and validation metrics approaching perfection.

Implications

The implications of these findings are profound, especially when considering the scalability of the approach to more extensive datasets and varied requirement types. We anticipate extending our methodology to include clustering techniques via network science, such as network models and community detection, to enhance our understanding of requirement dependencies within software projects.

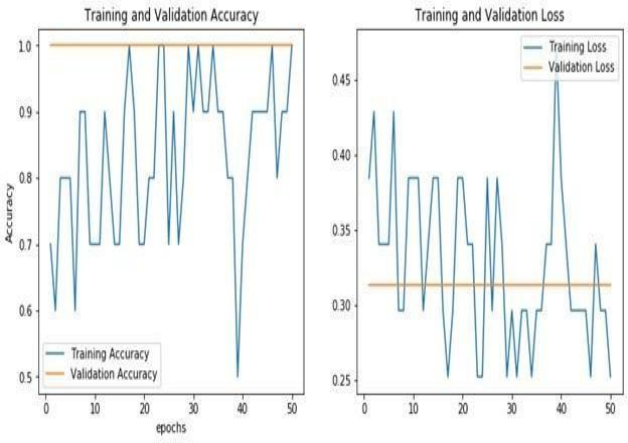

Fig.4 Model accuracy and loss with unlabeled data.

In-depth Discussion and Comparative Synthesis

The results derived from our empirical evaluations elucidate the complex dynamics that deep learning (DL) and recurrent neural network-based requirements engineering (RNNRE) could potentially introduce into the requirements engineering (RE) process. The initial experiment, which employed accuracy, precision, and recall as surrogates for reliability, consistency, and completeness, provided a sophisticated methodology for evaluating software requirements' intricatepatterns and integrity. Such a methodology is not merely procedural but critical for the nuancedalignment of requirements spanning explicit directives and implicit expectations integral to effective software project planning.

The insights documented in Table 2 and Table 3 underscore the transformative influence of DL and RNNRE as facilitators within the RE sphere. However, the variability in performance across different datasets accentuates the intricacy inherent in applying these advanced computational models. This underscores the necessity for a bespoke approach, where DL models are finely tuned to the particularities and semantic nuances of the dataset in question.

Our research approach did not fully explore the breadth of hyper-parameter optimization, which suggests that further enhancements could be realized. For instance, expanding the deep learning architecture with additional hidden layers or experimenting with alternative activation functions could unveil more profound layers of abstraction within the requirements, thereby enhancing the model's predictive accuracy and interpretability.

This investigation provokes critical considerations for industry adoption: Is manual labeling, which requires meticulous expert intervention, sustainable in burgeoning datasets? Alternatively, does the future beckon toward paradigms such as semi-supervised or unsupervised learning, which can extract structure from unstructured data with minimal human oversight?In scenarios involving unlabeled data, the integration of visualization tools becomes essential. Such devices can reveal the latent interdependencies within requirements, thus enabling more informed and strategic decision-making.

Our study ventures into unexplored territory by employing self-labeled and unlabeled data in RNNRE frameworks. Using the EMR certification dataset for classification purposes represents an innovative endeavor. Given the unique methodological path we have embarked upon, direct comparisons with extant research are not feasible. Nonetheless, our exploratory efforts have uncovered valuable insights and established a foundation for future research. Theimplications of our findings are manifold, prompting further inquiry into how these emerging technologies might be leveraged to engender a more agile, effective, and insightful RE

process.

Conclusion and Future Work

This study has explored the intricacies of Requirements Engineering (RE), mainly focusingon integrating diverse stakeholder interests and the challenges in accurately identifying and validating requirements to meet business objectives. The deployment of our Recurrent Neural Network for Requirement Engineering (RNNRE) model represents a significant methodological leap in effectively managing the structural complexity of requirements. This advancement enriches the tools available for RE, contributing to a more nuanced mapping and interpretation of needs.

Empirical evidence from our research underscores the influence of granularity in mapping conditions to requirements on classification accuracy. The RNNRE model has demonstrated impressive performance, aligning multi-level requirements with their corresponding acceptancecriteria and achieving quality metrics—accuracy, precision, and recall—exceeding 81%. However, the constraints of the limited dataset size and lack of diversity necessitate further empirical validation to affirm the model's robustness and broader applicability. Future research directions include.

Dataset Expansion: Exploring more extensive and diverse datasets to uncover more subtle aspects of requirements complexity.

Class Balancing Techniques: Investigating different techniques to enhance model performance, especially in imbalanced classes.

Alternative Learning Paradigms: Experimenting with unsupervised and semi-supervisedlearning methods to deepen the understanding of data relationships.

Comparative Analysis: Conduct benchmarking studies with established models anddatasets to validate and position the RNNRE model within the broader RE context.

Our research will expand to include clustering techniques on similar datasets to unearth the interdependencies among various requirements. This will help integrate classification and clustering methodologies, improving decision-making in RE by uncovering patterns not immediately visible through classification. Such advancements are expected to push the boundaries of machine learning applications in RE, equipping practitioners with sophisticated analytical tools for navigating the complexities of software development.

Funding Information

This research received no specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Author's Contributions

Justine Nakirijja, Conceptualization, Methodology, Software, Data curation, coding, first draft Writing, validation. Yahya Abid, methodology, Code review, formal analysis, Review of all drafts/Writing, Validation. Ravi Samikannu, formal analysis, review of all drafts/writing, and validation. Lory Liza D. Bulay-og, formal analysis, Review of all drafts/Writing, Validation.

References

[1] Abualhaija, S., Arora, C., Sabetzadeh, M., Briand, L. C. & Traynor, M. Automated demarcation of requirements in textual specifications: a machine learning-based approach. Empirical Software Engineering, 25 (2020), 5454–5497. https://link.springer.com/article/10.1007/s10664-020-09864-1

[2] Sleem, S., Capretz, L. F. & Ahmed, F. Benchmarking machine learning tech- nologies for software defect detection., 2015, https:arXiv preprint arXiv:1506.07563. https://doi.org/10.48550/arXiv.1506.07563

[3] Alrumaih, H., Mirza, A. & Alsalamah, H. Domain ontology for requirements classi-fication in a requirements engineering context. IEEE Access, 8 (2020), 89899–89908. https://ieeexplore.ieee.org/document/9091131

[4] AlOmar, E., A., Mkaouer, W., M., Newman, C. & Ouni, A. On preserving the behavior in software refactoring: A systematic mapping study. Information and Software Technology, 140 (2021), no. 2, 459–502. https://doi.org/10.1016/j.infsof.2021.106675

[5] Barry-Straume, J., Tschannen, A., Engels, D. W. & Fine, E. An evaluation of training size impacts validation accuracy for optimized convolutional neural networks. SMU Data Sci-ence Review, 1 (2018), no. 1, 12. https://scholar.smu.edu/datasciencereview/vol1/iss4/12.

[6] Maciejauskaitė, M. & Miliauskaitė, J. (2024). The efficiency of machine learning algorithms in classifying non-functional requirements. New

Trends in Computer Sciences, 2(1), 46–56. https://doi.org/10.3846/ntcs.2024.21574

[7] Patel, V., Mehta, P. & Lavingia, K. Software Requirement Classification Using Ma-chine Learning Algorithms. Proceedings of the 2023, nternational Conference on Artifi- cial Intelligence and Applications (ICAIA) Alliance Technology Conference (ATCON-1). https://ieeexplore.ieee.org/document/10169588

[8] Damasiotis, V., Fitsilis, P., Considine, P. & Kane, O. J. Analysis of software project complexity factors. Proceedings of the 2017 International Conference on Management Engineering, Software Engineering and Service Sciences, 2017, 54–58. https://dl.acm.org/doi/pdf/10.1145/3034950.3034989

[9] Fitsilis, P. Measuring the complexity of software projects. 2009 WRI World Congress on Computer Science and Information Engineering, 7 (2009), 644–648. http://dx.doi.org/10.1109/CSIE.2009.936

[10] Gupta, Varun and Fernandez-Crehuet, Jose Maria and Hanne, Thomas and Telesko, Rainer. Requirements engineering in software startups: A systematic mapping study. Ap- plied Sciences, 10 (2020), no. 17, 6125. http://dx.doi.org/10.3390/app10176125

[11] Hey, T., Keim, J., Koziolek, A. & Tichy, W. F. Norbert: Transfer learning for requirements classification—2020 IEEE 28th International Requirements Engineering Conference (RE) 2020, 169–179. https://doi.org/10.1109/RE48521.2020.00028

[12] Hossin, M. & Sulaiman, M. N. A review on evaluation metrics for data classification eval-uations.International journal of data mining & knowledge management process., 2015, https://DOI : 10.5121/ijdkp.2015.5201.

[13] Ismail, F. H., Forestier, G., Weber, J., Idoumghar, L. & Muller, P. A. Deep learning for time series classification: a review. Data mining and knowledge discovery, 33 (2019), 917–963. https://doi.org/10.1007/s10618-019-00619-

[14] Jiao, J. & Chen, C. H. Customer requirement management in product develop- ment: a review of research issues. Concurrent Engineering, 14 (2006), no. 3, 173–185. http://dx.doi.org/10.1177/1063293X06068357

[15] Khattak, F. K., Jeblee, S., Pou-Prom, C., Abdalla, M., Meaney, C. & Rudzicz, F. A survey of word embeddings for clinical text. Journal of Biomedical Informatics, 100 (2019), 100057. https://doi.org/10.1016/j.yjbinx.2019.100057

[16] Kowsari, K., Jafari, M. K., Heidarysafa, M., Mendu, S., Barnes, L. & Brown, D. Text classification algorithms: A survey. Information, 10 (2019), no. 4, 150. https://doi.org/10.3390/info10040150

[17] Li, C., Huang, L., Ge, J., Luo, B. & Ng, V. Automatically classifying user requests in crowdsourcing requirements engineering. Journal of Systems and Software, 138 (2018), 108–123. https://doi.org/10.1016/j.jss.2017.12.028

[18] Maxwell, A., Li, R., Yang, B., Weng, H., Ou, A., Hong, H., Zhou, Z., Gong, P. & Zhang, C. Deep learning architectures for multi-label classification of intelligent health risk prediction.

BMC bioinformatics, 18 (2017), 121–131. https://doi.org/10.1186/s12859-017-1898-z

[19] Onyeka, E. A process framework for managing implicit requirements using analogy-based reasoning: Doctoral consortium paper. IEEE 7th International Conference on Research Challenges in Information Science (RCIS), 2013, 1–5. https://ieeexplore.ieee.org/document/6577726

[20] Rahman, M. A., Haque, M. A., Tawhid, M. N. A. & Siddik, M. S. Classifying non-functional requirements using RNN variants for quality software development. Proceedings of the 3rdACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation, 2019, 25–30. https://doi.org/10.1145/3340482.3342745

[21] Khurana, D., Koli, A., Khatter, K. et al. Natural language processing: state of the art, current trends and challenges. Multimed Tools Appl 82, 3713–3744 (2023). https://doi.org/10.1007/s11042-022-13428-4

[22] Randell, A., Spellman, E., Ulrich, W. & Wallk, J. Leveraging business architec- ture to improve business requirements analysis. Business Architecture Guild, 2014. https://cdn.ymaws.com/www.businessarchitectureguild.org

[23] Rasheed, A., Zafar, B., Shehryar, T., Aslam, N. A., Sajid, M., Ali, N., Dar, S. H & Khalid, S.

Requirement engineering challenges in agile software development. Mathematical Problems in Engineering, 2021 (2021), 1–18. https://doi.org/10.1155/2021/6696695

[24] Ren, Y., Zhao, P., Sheng, Y., Yao, D. & Xu, Z. Robust softmax regression for multi-class classification with self-paced learning. Proceedings of the 26th International Joint Confer- ence on Artificial Intelligence, 2017, 2641–2647. https://doi.org/10.24963/ijcai.2017/368

[25] Rodriguez, S., Thangarajah, J. & Winikoff, M. User and System Stories: an agile approach for managing requirements in AOSE. Open Access Te Herenga Waka-Victoria University of Wellington, 2021. https://doi.org/10.26686/wgtn.14527758.v1

[26] San Cristóbal, J. R., Carral, L., Diaz, E., Fraguela, J. A., Iglesias, G. & oth- ers. Complexity and project management: A general overview. Complexity, 2018. https://api.semanticscholar.org/CorpusID:53098515

[27] Shabi, J., Reich, Y., Robinzon, R. & Mirer, T. A decision support model to manage overspecification in system development projects. Journal of Engineering Design, 32 (2021), no. 7, 323–345. https://doi.org/10.1080/09544828.2021.190897

[28] Torfi, A., Shirvani, R. A., Keneshloo, Y., Tavaf, N. & Fox, E. A. Natural language processing advancements by deep learning: A survey., 2020, https://arXiv preprint arXiv:2003.01200.

[29] Wallace, D. R. & Cherniavsky, J. C. Guide to software acceptance. DIANE Publishing, 1990. https://api.semanticscholar.org/CorpusID:56489605

[30] Winkler, J. P., Grönberg, J. & Vogelsang, A. Predicting How to Test Requirements: An Automated Approach. 2019 IEEE 27th International Requirements Engineering Conference (RE), 2019, 120–130. https://ieeexplore.ieee.org/document/8920404

[31] Zhao, L., Alhoshan, W., Ferrari, A. & Letsholo, K. J. Classification of natural lan- guage processing techniques for requirements engineering., 2022, https://arXiv preprint arXiv:2204.04282.