2021 IEEE 29th International Requirements Engineering Conference Workshops (REW) | 978-1-6654-1898-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/REW53955.2021.00025

Is BERT the New Silver Bullet? - An Empirical Investigation of Requirements Dependency Classification

Gouri Deshpande*, Behnaz Sheikhi†, Saipreetham Chakka †,
Dylan Lachou Zotegouon†, Mohammad Navid Masahati*, and Guenther Ruhe*
*Dept. of Computer Science
University of Calgary, Calgary, Canada
†Dept. of Electrical and Software Engineering
University of Calgary, Calgary, Canada
Email: {gouri.deshpande, behnaz.sheikhi1, saipreetham.chakka1,
dylanvalentin.lachou, mohammadnavid.masaha, ruhe}@ucalgary.ca

***Abstract*—Bidirectional Encoder Representations from Transformers (BERT) is a successful transformer-based Machine Learning technique for Natural Language Processing (NLP) based tasks developed by Google. It has taken various domains by storm, and Software Engineering is one among them. But does this mean that BERT is the new Silver Bullet? It is certainly not. We demonstrate it through an empirical investigation of the Requirements Dependency Classification (RDC). In general, based on various criteria used for evaluation, decisions on classification method preference may vary. For RDC, we go beyond traditional metrics such as the F1 score and consider Return-on-Investment (ROI) to evaluate two techniques for such decision making. We study RDC-BERT (fine-tuned BERT using data specific to requirements dependency classification) and compare with Random Forest, our baseline. For RDC and data from FOSS system Redmine, we demonstrate how decisions on method preference vary based on (i) accuracy, (ii) ROI, and (iii) sensitivity analysis. Results show that for all the three scenarios, method preference decisions depend on learning and evaluation parameters. Although these results are with respected to the chosen data sets, we argue that the proposed methodology is a prospective approach to study similar questions for data analytics, in general.**

I. Introduction

Not every solution can be transferred from one context to another. This has been translated to a widely accepted quote, *One size does not fit all* in the domain of Software Engineering [1] and beyond. It emphasizes that models, techniques, tools, or processes need to be adapted to the context of their usage. This notion is further underpinned by Turing Award winner Fred Brooks [2], who stated, "there is no single development, in either technology or management technique, which by itself promises even one order of magnitude [tenfold] improvement within a decade in productivity, in reliability, in simplicity".

Recently Bidirectional Encoder Representations from Transformers (BERT) has received massive attention due to outstanding performance in various Natural Language Processing (NLP) tasks since its inception [3], [4], [5], [6]. The pre-trained BERT model is trained on massive unlabeled data (such as Wikipedia) over different pre-training tasks. For fine-tuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream task [7]. This task is Requirements Dependency Classification (RDC) for our study.

The fine-tuned BERT model using RDC-specific data from the Redmine FOSS project [8] is referred to as *RDC-BERT* in our study. We compare RDC-BERT with the Random Forest algorithm[1] with the perspective of not just accuracy but also Return-on-Investment (ROI).

Requirement dependencies affect software development activities such as the design, testing, and releasing of software products. Requirement changes are one of the most crucial aspects that occur during requirement specification as a change in a requirement can trigger changes in other related requirements. Relationships between requirements act as a basis for change propagation analysis and drive various software development decisions [9]. However, such change propagation poses challenges to the developers because it consumes substantial efforts as the requirements are mostly documented in natural language. Hence, it is crucial to know/extract all possible relationships that could occur among the requirements.

Data Analytics (DA) is time and effort-consuming,

[1]We evaluated Random Forest, Support Vector Machine and Naive Bayes ML algorithms and chose the model with the highest F1 score as our baseline to compare with RDC-BERT.

978-1-6654-1898-0/21/$31.00 ©2021 IEEE
DOI 10.1109/REW53955.2021.00025

and not automatically valuable. Moreover, organizations, who rely heavily on Machine Learning, seek transparency in the algorithms that guide decisions and explore additional criteria to evaluate algorithms [10] objectively. Since for decision-makers, it is vital to have an affinity to analytics rather than programming and modeling [11], we emphasize that it is crucial to consider ROI like criteria to analyze value and relate it to the effort invested for a chosen method.

We propose using ROI as evidence to support the need for additional data (How much?), subsequent effort instead of just pushing for advanced analytics (what?), and its implementation (how?) for a given problem [12]. We consider the complete life-cycle of DA, which includes all pre-processing and post-processing stages to enable practitioners to control the degree and scope of DA usage.

Overall, our paper makes the following contributions

- Compare and evaluate ML technique: Random Forest (baseline) and RDC-BERT in terms of accuracy.
- Formulate a mechanism to evaluate the two ML methods in terms of ROI and demonstrate that the F1 score should not be the sole criteria to weigh the efficacy of methods.
- Analyze how varying cost factor estimates impact ML algorithm preference decisions.

We provide access to our data[2] and source code to facilitate further research along the lines explored here.

Following this Introduction, we describe basic concepts and the research questions addressed in this paper in section II. This is followed by the approach to answer the proposed research questions in section III. Section IV provides more information on the dataset used, while Section V elaborates our ROI model. Results are then discussed in Section VI while related work is detailed in Section VIII. Section VII discusses the validity of results and we conclude the paper with an outlook to future research in Section IX.

II. Basic Concepts and Research Questions

In this section, we present basic concepts and formulate our research questions.

A. Definitions

Requirements dependencies are (various types of) relationships among requirements. For a set of requirements R if any pair of requirements (r, s) belongs to R then,

Definition: A pair *(r, s)* of requirements *r* and *s* is *related* if implementation of one requirement during development impacts the other one [13] then requirements *r* and *s* are in a (symmetric) relationship called *RELATES_TO*[3], i.e. [*r RELATES_TO s*].

Definition: For a pair *(r, s)* of requirements, if *r* and *s* do not have dependency relationship then, *r* and *s* are in a relationship called *INDEPENDENT*.

B. Research Questions

We evaluate three research questions in the context of Requirements Dependency Classification (RDC) using the Redmine dataset:

RQ1: In terms of accuracy and varying training set size, how does RDC-BERT compare with Random Forest?

RQ2: How does the ROI of RDC-BERT compare with the ROI of Random Forest?

RQ3: How sensitive are the results of RQ2 for varying cost estimates?

C. Evaluation Metrics

Confusion matrix: A confusion matrix[4] is a matrix that contains information relating to actual and predicted classifications. For *n* classes, CM will be an $n \times n$ matrix associated with a classifier. Table I shows the principal entries of CM for a binary class classification.

TABLE I: A confusion matrix of binary (two) class classification problem

	Predicted Negative	**Predicted Positive**
Actual Negative	True Negative (TN)	False Positive (FP)
Actual Positive	False Negative (FN)	True Positive (TP)

F1 score: F1 score is a measure of the model's accuracy. Its computation based on actual and predicted class values is shown in (1).

$$F1 = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{1}$$

ROI: To determine the ROI, we follow its simplest form of calculation relating the difference between $Benefit$ and $Cost$ to the amount of $Cost$ as shown in (2). Both $Benefit$ and $Cost$ are measured as human effort in person-hours.

$$ROI = (Benefit - Cost)/Cost \tag{2}$$

In the crux, our study evaluates various conditions under which RDC-BERT could be preferred over the baseline

[2]http://doi.org/10.5281/zenodo.5044654

[3]'Related issues' allow developers to link issues to each other in order to simplify their workflow.

[4]We utlize values from FP and FN in our study

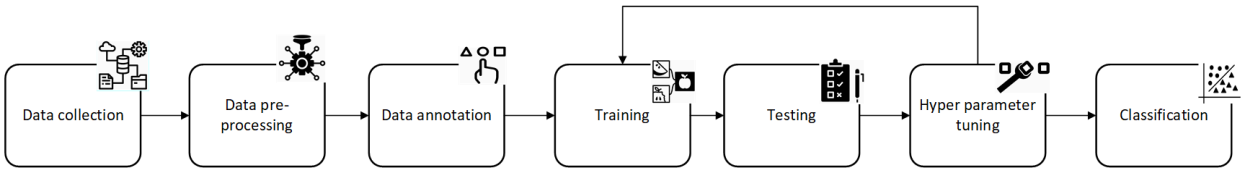

Fig. 1: Various steps that formulate the Machine Learning process (images curtsy NounProject.com)

ML (i.e. RF). Specifically, overriding the exclusive accuracy considerations, we compare F1 and ROI from RF and RDC-BERT and analyze different results depending on the preference criterion such as train set and varying cost factors.

III. Methodology

For this study, we utilize a labeled dataset (apriori information of requirement dependencies). Beginning with a small portion for training, we increase the training set size incrementally. In this manner, we compare RDC-BERT with the RF on varying values from the confusion matrix (II-C). In this section, the methodology of this study for CM values of varying sizes of the training set is presented.

A. ML Classification Process

To utilize ML for any problem, it is a must to follow a sequence of steps. All these steps and their possible replication could be effort-intensive, which have implications on the ROI projection. Different ML process steps are mentioned in the literature. The most accepted steps are as shown in Figure 1. After the problem formulation, the first step is to perform data collection from one or more sources. Since data can not be used directly in its raw form, it is essential to extract or construct usable data before cleaning and pre-processing as part of data preparation. Such pre-processed data is further used for training. Model evaluation is to analyze the accuracy of the trained model on the test set. Hyper-parameter tuning is often used to enhance the performance of the model on the test set before it is used for the actual (unlabeled) data classification [7].

B. Research Design

Figure 2 provides an overview of the different steps of the methodology and how it relates to the three RQs.

As shown in step ❶ of Figure 2, textual data was processed to extract requirement descriptions. Then, it was further pre-processed (❷) to eliminate noise such as spatial characters and numbers. Thus generated output (step ❸) is fed to RDC-BERT and RF for training. Since RF needs explicit requirement classification, we use TF-IDF to generate word vectors (step ❹) before training.

Care is taken to process the same data snapshot, which was also fed to the baseline, to fine-tune the pre-trained BERT model in step ❺. Further, the fine-tuned BERT model (RDC-BERT) is then used for classification.

To derive the ROI values, we utilize various cost estimates (obtained from practitioners) towards these metrics (❻, ❼). The definition of these parameters is as shown in Table II). Later in step ❽, we also compute the ROI for RDC-BERT and compare it with the ROI of RF. Finally, sensitivity analysis is performed to understand how factors impact the ROI computation (❾).

TABLE II: Confusion matrix terminology specified for RDC

Parameters	Meaning
True Positive (TP)	Predicted dependent requirement pair is truly dependent
True Negative (TN)	Predicted independent feature pair is truly independent
False Positive (FP)	Independent requirement pair is incorrectly predicted as a dependent (miss-identified)
False Negative (FN)	Dependent requirement pair is incorrectly predicted as independent (dependency is missed)

C. RDC-BERT

We use a pre-trained BERT model in combination with our RDC-specific dataset. The result is a fine-tuning BERT model called *RDC-BERT*. We use BertForSequenceClassification from the huggingface PyTorch library [14] for this implementation.

In every instance, for a given training set size, RDC-BERT was trained through three epochs with a batch size of 32, and a learning rate of 2e-5. In each epoch, the training set was divided into 90% for training and 10% for validation. Finally, RDC-BERT was used to classify the test set and the resulting F1 score and confusion matrix were captured.

D. Baseline: Random Forest

While for RDC-BERT, the data were retained in their original form, it was further passed through the

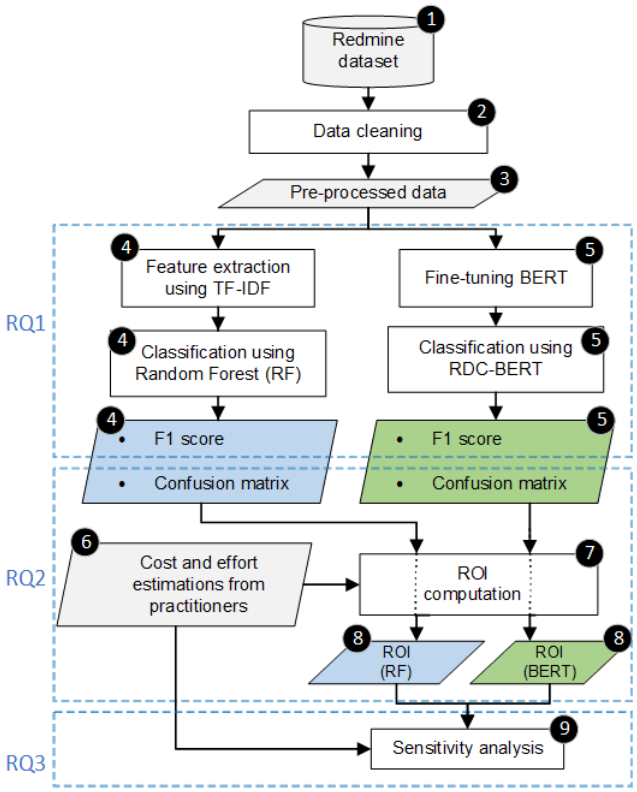

Fig. 2: Research design of the study

NLP pipeline (explained in detail in section IV. B) [15] for stopword removal, lemmatization, and TF-IDF vectorization[12], before feeding it to RF for classification. The same train and test split dataset were utilized in both RDC-BERT and RF to retain data authenticity.

For a given training set, hyper-parameter tuning was performed using random search [16]. We also performed 10 times 10-fold cross-validation. That way, the best F1 score and confusion matrix on the held-out test set was captured.

E. Evaluation Setup

Classification: For RDC, we denote requirement pairs having *RELATES_TO* dependencies as *positive classes* and *INDEPENDENT* pairs as *negative classes*. Dataset was balanced using under-sampling technique [17].

Iteration: First, for both RF and RDC-BERT, the original data was split into two parts with a 80:20 ratio between training and test sets. For training, in the first iteration, 5% of the dataset was randomly picked. Over the subsequent iterations, the training set was incrementally increased by adding randomly picked 5%. This process was repeated until the size of 80% for the training set was achieved.

Hence, steps ❹ through ❽ constitute one iteration. The steps were repeated in every subsequent iteration: Computation of the F1 score, of the confusion matrix for a held-out test set, and the subsequent ROI computation. Each iteration was repeated 10 times.

IV. Data

A. Data Collection

Redmine [8] is a free and open-source, web-based project management and issue tracking software tool. Various issues related to various projects are updated each day which helps software developers to track for effective implementation. Redmine also hosts the Redmine project's data in this issue tracking tool. In the Redmine project, requirements are a specific type of issue that is extracted.

Totally 6,949 issues of various types such as defect, patch and requirements, were extracted. Of these 3,994 were requirements. For each requirement, its id, description, subject, links, and date of creation fields were gathered through Redmine's API.

B. Data Preparation

During the coding and testing phases of software development, it is essential to be aware of various other requirements that would depend on each other to avoid re-work due to test case failures. From a version release perspective, knowing *related* requirements help to handle and release them in conjunction, as their implementation, testing, and customer value are facilitated from handling them in the same release.

The "subject" field contained meaningful information which described the requirement briefly. Thus we used content from "subject" as a textual requirement description. Analyzing sentence lengths of this field revealed that most of the lengths of the sentences were in the range of 4 to 25 words. So, as a first step, sentences that had fewer than three words were eliminated, which reduced the requirements to 3,259.

The "links" field of a requirement consisted of id and relationship tuple. We looked up dependent requirements based on this id and generated a dependency. Overall 2,469 requirements, which had one or more ids listed in "links" field), generated 3,664 *RELATES_TO* dependency pairs.

For this study, we exclusively analyzed the *RELATES_TO* dependency since it is the most frequently occurring dependency in this dataset. Table III shows sample pairs of this dependency type. We used the 790 requirements that had empty "links" field (meaning no

TABLE III: Sample *RELATES_TO* dependency pairs

ID	Description	ID	Description
8562	Watchers list too big in new issue form	34556	Setting to change the maximum number to display on the new issue form
34549	Add keyboard shortcuts for wiki toolbar buttons	30459	Switch edit/preview tabs with keyboard shortcuts

dependencies) to generate *INDEPENDENT* pairs. A pair of id was randomly picked from this pool to generate 10,000 *INDEPENDENT* pairs.

C. Data Pre-processing

We do not need to perform pre-processing for RDC-BERT due to the robust nature of pre-trained BERT. For RF, we removed URLs, punctuation and non-English characters first. Then the sentences were tokenized (converted sentences into smaller units called tokens). Then, these word vectors were processed to remove stop words such as *and, the, in, at*, etc. Finally, word vectors were lemmatized further (which is useful in removing the inflectional ending from words in a sentence and returns the base or dictionary form of a word, which is known as the lemma) before translating into numerical vectors using TF-IDF vectorization [18].

V. ROI MODELING

The Return-of-investment (ROI) computation has two important independent variables: *Return* and *Investment*. In our modeling, we associate *investment* primarily with the effort towards the process of ML for RDC and *return* to the benefit projected from the insights and results received once applied in the problem context. In this section, we will elaborate more on these two basic dimensions of ROI computation.

A. Investment: The Cost Factor

Data processing is an umbrella term used to combine data collection (C_{dg}), pre-processing (C_{pp}), and labeling (C_l)[5] under one hood, each one of which is a cost component. However, not all costs are fixed and some vary based on the solution approach used to tackle any decision problem. Additionally, there is a cost associated with hyperparameter tuning (C_{ft}) modeling and evaluation (C_e).

[5]We are using a completely labelled data for this study to evaluate the three RQs. Thus, the cost of labeling has been used as a proxy to imitate the real-world scenario where un-labelled data is abundant and labelled data is scarce to use ML

TABLE IV: Parameters used for ROI computation

	Symbol	Meaning	Unit
Cost (per sample)	C_{dg}	Data collection time	Minutes
	C_{pp}	Pre-processing time	Minutes
	C_e	Training and testing time	Minutes
	C_l	Labeling time	Minutes
	C_{ft}	Hyper-parameter tuning time	Minutes
	C_{res}	Human resource labour cost	$/hour
Classification Penalty	$Cost_{FP}$	Penalty per FP	$
	$Cost_{FN}$	Penalty per FN	$
Others	N_H	#Human resources	Number
	N_{train}	Training set size	Number
	N_{test}	Test set size	Number
	$Value_{product}$	Estimated value of the product per release	$

B. Return: The Value Factor

In the context RDC problem, the benefit could be modeled in terms of the ability of the ML model to produce the least amount of overhead by 1) Incorrectly classifying independent as a dependent (False Positive) 2) Incorrectly classifying dependent as independent (False Negative). So, using $Cost_{FP}$ and $Cost_{FN}$ as estimated re-work costs due to *classification penalty* then $Sum(Cost_{FN}+Cost_{FP})$ would be the cumulative expense that a company has to bear.

In a release cycle, if *estimated value* that a product could generate is :$Value_{product}$ then the $Benefit$ would be the difference of the *estimated value* and the *classification overhead*. Table IV lists the relevant cost components and their corresponding units.

C. Cost and Benefit Estimation

As explained in Section III-E, in this empirical analysis, we conducted classification by utilizing a varying and increasing size of the training set. Further, for the ROI analysis, in every iteration, $Cost$ and $Benefit$ were computed using the parameters explained in Table IV. $Cost$ is the sum of the data processing costs ($C_{dg}+C_{pp}+C_{ft}+C_e+C_l$)/60 (in hours) for all the train (N_{train}) and test (N_{test}) samples ($=n$) in every iteration. This is further translated into dollar cost based on hourly charges (C_{res}) of N_H human resources as shown in 3.

$$Cost = n * \frac{(C_{dg}+C_{pp}+C_{ft}+C_e+C_l)}{60} * N_H * C_{res} \quad (3)$$

$Return$ computations for RDC, assumes reward ($Cost_{FP}$) for misidentifying independent requirements (FP) and heavily penalizing ($Cost_{FN}$) instances that

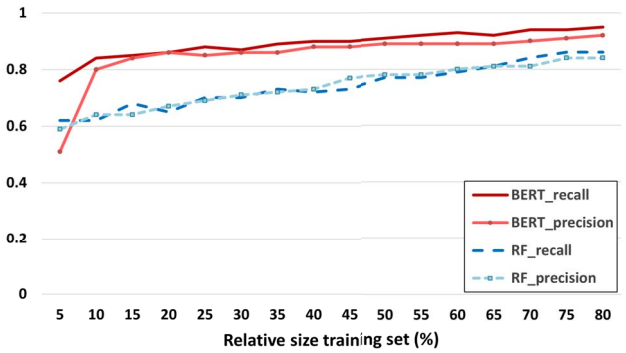

Fig. 3: Precision and recall of RDC-BERT and RF

were falsely classified as independent (FN). Equations (4) and (5) show these computations.

$$TotalPenalty = FP * Cost_{FP} + FN * Cost_{FN} \quad (4)$$

$$Return = Value_{product} - TotalPenalty \quad (5)$$

D. ROI Sensitivity Analysis

We perform sensitivity analysis to investigate the effects of various principal parameters on the ROI computation. Harman et al. [19] emphasized that Software engineering is plagued by problems associated with unreliable cost estimates and sensitivity analysis could be a method to assess the impact of inaccuracies of the cost estimation.

For sensitivity analysis, we only modify the parameter values repeatedly and capture the results. Table V shows parameters and corresponding values considered while computing the cost and benefit for ROI analysis. Parameters such as $Cost_{FP}$ (cost of misidentifying the positive prediction), $Cost_{FN}$ (cost of missing a dependency) and $Value_{product}$ are defined to calculate the total benefit in each iteration. A sensitivity analysis was performed to identify how these values impacted the ROI preference between techniques once cost factors were changed in a predefined range.

VI. Results

A. RQ1 - Comparison Based on Accuracy

Figure 3 shows the precision and recall curves for the two methods. Although precision and recall of the RDC-BERT are low at the beginning, they pick up dramatically after initial slow growth and outperform RF. As shown in Figure 4, over incremental training set size, F1 of RDC-BERT outperformed the RF by a margin of 10%. The highest F1 score that RDC-BERT could achieve was 0.93 whereas the RF peaked at 0.83. Right from the beginning, the F1 score of RDC-BERT continued to perform better comparatively and could reach its peak with 70% or more training set size, however, the increase hit a plateau beyond that point.

Finding 1
RDC-BERT outperforms RF in terms of F1 right from the beginning. RDC-BERT and Random Forest achieved a maximum of F1 = 0.93 and 0.83 respectively with 70% of training set. However F1 hits a plateau beyond this point.

B. RQ2 - Comparison Based on ROI

To compare RDC-BERT with RF in terms of ROI, we utilized domain expertise to estimate the various cost components of investment and benefit. Thus, we utilized estimations from the CEO of Typo3 [20], an OSS application, through an interview. Table V lists the parameters and corresponding values from that interview(❻). Typo3 and Redmine are projects of a similar domain and similar size, so we transferred estimates from Typo 3 to Redmine.

TABLE V: Parameter settings for the two analysis scenarios

Parameters	Values
$C_{fixed} = C_{dg} + C_{pp} + C_e$	1 min/sample
C_l	0.75 min/sample
C_{res}	$65/hr
C_{ft}	0.1 min/sample
N_H	9
N	7,328 (balanced dataset)
$Cost_{FN}$	$24,960
$Cost_{FP}$	$10,400
$Value_{product}$[1]	$4,000,000

[1]This value was computed using various cost estimates for a period of one release cycle (= 18 months)

Figure 5 shows a comparison of the ROI between RF and RDC-BERT. RF performs poorly and achieves positive ROI only with 75% or more training set size. However, RDC-BERT needs close to 45% of the training set size to achieve positive ROI, which is substantial and tends to be an expensive investment in terms of effort and cost. Additionally, the ROI begins to plateau beyond 70% training set size.

Note that RDC-BERT starts to yield positive ROI only after it is provided with more than 45% of the training set. Interestingly, if only 25% to 40% of the dataset is available for training, then choosing either of the RFs

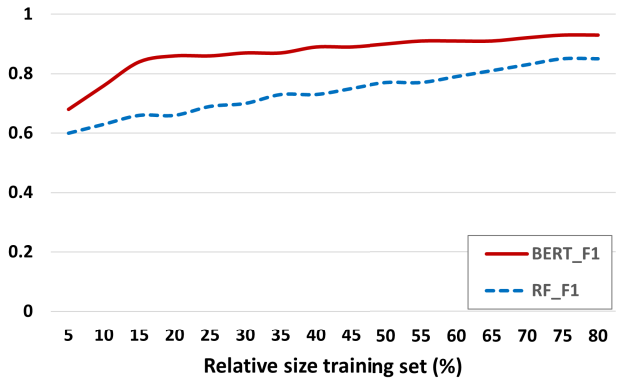

Fig. 4: F1 of RDC-BERT vs RF for varying training set sizes

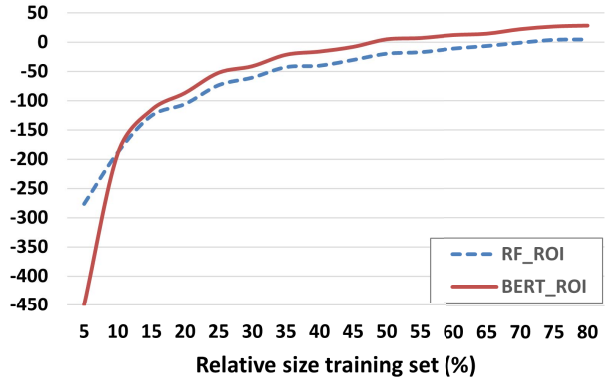

Fig. 5: ROI of RDC-BERT vs RF for varying training set sizes

and RDC-BERT does not make a huge difference. In hindsight, choosing RF could be more economical as it is computationally inexpensive comparatively.

Finding 2
RDC-BERT needs at least 45% or more data to generate positive ROI. With just about 25% to 40% data available for training, both RF and RDC-BERT perform about the same and generate negative ROI.

C. RQ3: Sensitive Analysis of Fine-tuned BERT

In this question, we were interested how much the results will change for changing (cost) values. We computed the ROI for both RDC-BERT and RF for varying $Cost_{FN}$ and $Cost_{FP}$ around the values defined in Table V. In particular, we studied two scenarios. We visualized the results as a heat-map. If the difference ROI(RDC-BERT) - ROI(RF) is positive then this means that RDC-BERT performs better. The higher the difference, the darker the color. Conversely, if the difference is negative then it shows that RF fares well comparatively.

In the first scenario, we varied $Cost_{FN}$ and $Cost_{FP}$ by 1000 units incrementally and computed results when RF and RDC-BERT utilized 5% of the dataset for training. We kept the $Value_{product}$ static at 4M. As shown in Figure 6, the RF was resilient to higher $Cost_{FN}$ values comparatively (yellow hue).

Secondly, we fixed the $Value_{product}$ to 3M and re-evaluated the ROI at 10% of the dataset like before. As shown in the 7, results show that RF was also very effective for higher values of $Cost_{FN}$ cost factors compared to RDC-BERT which held strong for $Cost_{FP}$ mostly.

Finding 3
Sensitivity analysis emphasized that for smaller training set size with varying $Cost_{FN}$, $Cost_{FN}$ for two different $Value_{product}$ values, RF showed better ROI even for higher $Cost_{FN}$ values which is the most expensive cost factor.

Summary: As shown in Table VI, RDC-BERT excels when there is 45% or more data to train and provides the F1>0.70 even with 5% of the dataset to train with. On the other hand, RF performs well comparatively with the small dataset to train with and generates up to 0.83 F1. However, due to the high false-negative rate, the ROI model becomes sensitive to cost factors.

TABLE VI: Summary of the results from the three research questions

		RF	BERT
Training with 5% - 15% data	F1	🏆	
	ROI	🏆	
Training with over 20% data	F1		🏆
	ROI		🏆

VII. Threats to Validity

Internal validity - The experiments conducted in this study are in the context of binary classification. Although we speculate that it would not alter the conclusions drawn, its impact remains to be explored in future work.

Our ROI computation is based on the cost and value factors defined by another project from the same repository. Thus, its implications on the result remain open to speculation. However, these values are obtained from practitioners (CEO of a Typo3 FOSS project), hence, we believe this threat is mitigated to a great extent.

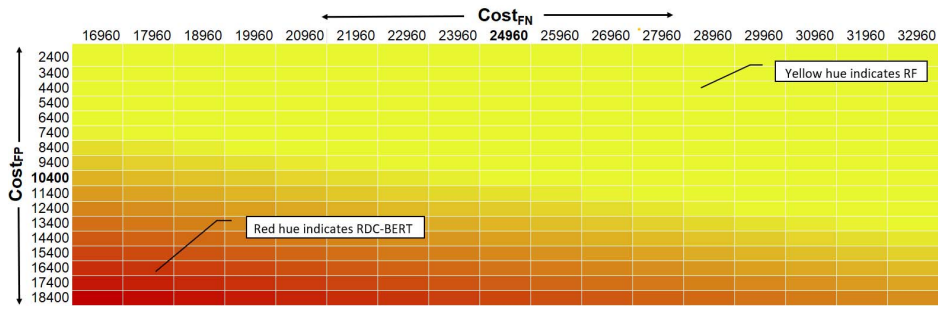

Fig. 6: Difference in ROI values of RDC-BERT and RF (RDC-BERT minus RF) for varying $Cost_{FN}$ and $Cost_{FP}$. The comparison is for 5% training set size and $Value_{product}$ = 4M. The yellow hue in heat-map (upper right triangle) shows that RF performs better than RDC-BERT for higher values of $Cost_{FN}$ in conjunction with lower values of $Cost_{FP}$

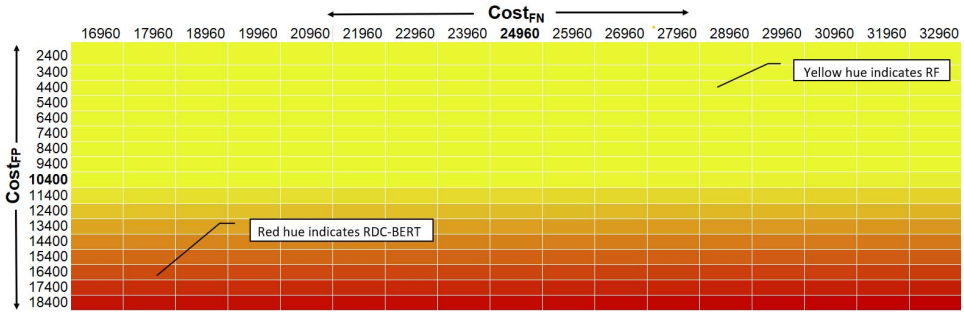

Fig. 7: Difference in ROI between RDC-BERT and RF for varying $Cost_{FN}$ and $Cost_{FP}$. The comparison is for 10% training set size and $Value_{product}$ = 3M. The yellow hue in heatmap (upper part of rectangle) shows that RF performed better for growing $Cost_{FN}$ and not so much for $Cost_{FP}$ compared to RDC-BERT when the dataset is smaller.

Obtaining project-related cost estimates from practitioners is an arduous task in itself. We do not exclude the implications of using cost estimates from just one practitioner, however, our preliminary results from [12] showed similar results for altered yet relative cost estimates.

External validity - This work focused on using one of the fine-tuned BERT models for a specific context and a specific data set. No claims for external validity are made, other than rejecting both initial null hypotheses.

Construct validity - The scope of our ML process is the ML classification model and development cost only. We do not focus on deployment or fit into the OSS developers' and maintainers' workflow. However, we envision incorporating it in our future work.

VIII. Related Work

A. Requirements Dependency Classification

The practical importance of the topic was confirmed by our survey [21] where more than 80% of the participants agreed or strongly agreed that (i) dependency type classification is difficult in practice, (ii) dependency information has implications on maintenance, and (iii) ignoring dependencies has a significant ill impact on project success.

In the recent past, many empirical studies have explored diverse computational methods that used Natural Language Processing (NLP) [22] [23], semi-supervised technique [24], hybrid techniques [25] and deep learning [26] to analyze requirement dependencies.

Deshpande [25] proposed a new method for extracting requirement dependencies. This method integrated active learning with ontology-based retrieval to extract requires, refines and others dependency types. Also, the results were analyzed on two industrial case studies, where two hybrid approaches were used to analyze the results.

Recently, in an industry case study, Biesialska et al. [27] analyzed the large-scale agile development project to identify dependencies between user stories. They showed that automatic detection of dependencies could help teams organize their work effectively.

However, none of these studies considered ROI to evaluate ML techniques and mostly draw conclusions based on accuracy.

B. Applications of BERT in Requirements Engineering

Araujo et al. [28] explored the application of BERT to automate the identification of software requirements from app reviews. In a similar vein, [29] also compared BOW (Bag of Words) and pre-trained BERT model for app review classification into a bug, feature and user experience. This study showed that pre-processing of data significantly improved the BOW performance. However, BERT showed a greater advantage overall.

Sainiani et al. [30] Utilized the BERT model to extract and classify requirements from large software engineering contracts into predefined classes. Their results showed that with BERT, a higher F-score could be achieved for the classification of requirements.

Das et al. analyzed [5] state of the art models such as RoBERTa, DistillBERT against fine-tuned BERT models and pre-trained BERT models on requirements specific data. Their results showed that models trained on specific data excelled comparatively. In another study, Abbas et al. [3] explored the relationship between requirements similarity and software similarity. In that, they compared BERT models with TF-IDF and Doc2Vec models. However, in their analysis, TF-IDF performed well due to the structure of the dataset.

Fishbach et al. [4] studied automatically extracting causal relationships to derive automatic test case and dependency detection between requirements using BERT. Results revealed that their BERT-based solution performed best with a fixed length of tokens in text. Lin et al. [31] proposed a Trace-BERT framework to trace the link between NLP artifacts such as requirements with source code. Results showed that this method was more effective compared to other deep learning methods used in the past.

The list of studies shows that BERT is widely used in RE in general in recent times. However, is it viable to relate the results to the fact that a large amount of training data is needed? needs to be studied further.

C. ROI-based Decision-making in Software Engineering

Boehm et al. [32] [33] presented quantitative results on the ROI of Systems Engineering based on the analysis of the 161 software projects in the COCOMO II database.

Khoshgoftaar et al. [34] demonstrated an interesting case study of a large telecommunication software system and presents a method for cost-benefit analysis of a software quality classification model. The cost and benefit computations were based on the type-I (FP) and type-II (FN) predictions of the classification models. Although these cost-benefit models were ahead of their time, the time and effort investment done on data and metrics gathering was not considered eventually for cost computation.

Ling et al. [35] proposed a system to predict the escalation risk of current defect reports for maximum return on investment (ROI), based on mining historic defect report data from an online repository. ROI was computed by estimating the cost of not correcting an escalated defect (false negative) to be seven times the cost of correcting a non-escalated defect (false positive).

Ferrari et al. [36] studied the ROI for text mining and showed that it has not only a tangible impact in terms of ROI but also intangible benefits - which occur from the investment in the knowledge management solution that is not directly translated into returns. However, the caveat was that, that it must be considered in the process of judgment to integrate the financial perspective of analysis with the non-financial ones. A lot of benefits occurring from the investment in this knowledge management solution are not directly translated into returns, but they must be considered in the process of judgment to integrate the financial perspective of analysis with the non-financial ones.

Nagrecha et al. [37] proposed a Net Present Value model from which and strategies to determine the cost and impact of analytics programs for an organization.

In our research, we not only consider data pre-processing cost as an additional cost aspect but also transform machine learning metrics to dollar amounts to arrive at cost and benefits.

IX. Conclusion and Future Work

In this paper, we used a fine-tuned variant of the breakthrough BERT technology, called RDC-BERT and applied it to requirements dependencies classification from textual requirements. The focus was not to find "just another application" of BERT or "just another algorithm" for requirements dependency classification. Instead, we moved a step ahead and compared BERT with Random Forrest (baseline technique of our study) on multiple evaluation criteria (F1 and ROI). For the Redmine data set, we demonstrated that the preference between BERT and RF depends both on the project parameters and the chosen criteria for the comparison.

While BERT is a powerful and widely applicable technique, it does not mean that it automatically is the preference all the time over RF. We also demonstrated that it is crucial to look beyond just accuracy, as this

simplifies the situation by enabling us to consider much more than the number of occurrences of FN, FN, TP, and TN. Determining the ROI of investing into any of the two techniques helps to paint a more comprehensive picture for decision making.

Decision-making for the preference of technologies based on usage is a data-sensitive problem. In our future work, we will collect more data from OSS or proprietary projects. For concrete decisions, in our research context, cost and value predictions need to be qualified. Value-based software engineering was introduced mainly by Biffl et al. [38]. However, it is still not widely accepted in the community. Predicting the value of preventing FN or FP in a classification setting is a challenging task and needs further modeling and investigations. We hope that our paper is the first step in this direction.

REFERENCES

[1] T. Zimmermann, "One size does not fit all," in *Perspectives on Data Science for Software Engineering*. Elsevier, 2016, pp. 347–348.

[2] F. P. Brooks and N. S. Bullet-Essence, "Accidents of software engineering," *Computer*, vol. 20, no. 4, pp. 10–19, 1987.

[3] M. Abbas *et al.*, "Is requirements similarity a good proxy for software similarity? an empirical investigation in industry." in *REFSQ*, 2021, pp. 3–18.

[4] J. Fischbach, J. Frattini, and A. Vogelsang, "Cira: A tool for the automatic detection of causal relationships in requirements artifacts," *arXiv preprint arXiv:2103.06768*, 2021.

[5] S. Das *et al.*, "Sentence embedding models for similarity detection of software requirements," *SN Computer Science*, vol. 2, no. 2, pp. 1–11, 2021.

[6] T. Hey *et al.*, "Norbert: Transfer learning for requirements classification," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE, 2020, pp. 169–179.

[7] J. Devlin *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[8] Redmine.org. (2020, Apr) Redmine: bug-tracking system. [Online]. Available: https://www.redmine.org/projects/redmine/issues/

[9] G. Ruhe and M. Nayebi, "What counts is decisions, not numbers — toward an analytics design sheet," in *Perspectives on Data Science for SE*. Elsevier, 2016, pp. 111–114.

[10] P. Mikalef *et al.*, *Big data and business analytics: A research agenda for realizing business value*. Elsevier, 2020.

[11] B. Windt, H. Borgman, and C. Amrit, "Understanding leadership challenges and responses in data-driven transformations," 2019.

[12] G. Deshpande and G. Ruhe, "Beyond accuracy: Roi-driven data analytics of empirical data," in *Proc. ESEM*, ser. ESEM '20. ACM, 2020.

[13] "Related issues," https://www.redmine.org/projects/redmine/wiki/RedmineIssues, accessed: Aug 2021.

[14] "Bert with the huggingface pytorch library." https://huggingface.co/transformers/training.html, accessed: May 2021.

[15] W. Wang *et al.*, "Detecting software security vulnerabilities via requirements dependency analysis," *IEEE Transactions on Software Engineering*, 2020.

[16] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *J. of ML research*, vol. 13, no. 2, 2012.

[17] K. Srisopha *et al.*, "How should developers respond to app reviews? features predicting the success of developer responses," in *Proc. EASE*, 2021, pp. 119–128.

[18] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proc. of the first instructional conference on ML*, vol. 242, no. 1. Citeseer, 2003, pp. 29–48.

[19] M. Harman *et al.*, "Search based data sensitivity analysis applied to requirement engineering," in *Proc. of the 11th Annual conf. on Genetic and evolutionary computation*, 2009, pp. 1681–1688.

[20] Typo3.org. (2021, May) Typo3 — the professional, flexible content management system. [Online]. Available: https://typo3.org/

[21] G. Deshpande and G. Ruhe. (2020, Dec) Survey: Elicitation and maintenance of requirements dependencies. [Online]. Available: https://ispma.org/elicitation-and-maintenance-of-requirements-dependencies-a-state-of-the-practice-survey/

[22] J. Dag *et al.*, "A feasibility study of automated natural language requirements analysis in market-driven development," *RE*, vol. 7, no. 1, pp. 20–33, 2002.

[23] R. Samer *et al.*, "New approaches to the identification of dependencies between requirements," in *31st Conference on Tools with Artificial Intelligence*, ser. ICTAI '19. ACM, 2019.

[24] G. Deshpande *et al.*, "Data-driven elicitation and optimization of dependencies between requirements," *Proc. RE*, 2019.

[25] G. Deshpande, Q. Motger *et al.*, "Requirements dependency extraction by integrating active learning with ontology-based retrieval," *Proc. RE*, 2020.

[26] J. Guo *et al.*, "Semantically enhanced software traceability using deep learning techniques," in *2017 IEEE/ACM 39th ICSE*. IEEE, 2017, pp. 3–14.

[27] K. Biesialska, X. Franch, and V. Muntés-Mulero, "Mining dependencies in large-scale agile software development projects: A quantitative industry study," in *Evaluation and Assessment in Software Engineering*, 2021, pp. 20–29.

[28] de Araújo *et al.*, "Re-bert: automatic extraction of software requirements from app reviews using bert language model," in *Proc. 36th Annual ACM Symposium on Applied Computing*, 2021, pp. 1321–1327.

[29] A. Araujo *et al.*, "From bag-of-words to pre-trained neural language models: Improving automatic classification of app reviews for requirements engineering," in *Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional*. SBC, 2020, pp. 378–389.

[30] A. Sainani *et al.*, "Extracting and classifying requirements from software engineering contracts," in *2020 Proc. RE Conference*. IEEE, 2020, pp. 147–157.

[31] J. Lin *et al.*, "Traceability transformed: Generating more accurate links with pre-trained bert models," in *2021 Proc. ICSE Conference*. IEEE, 2021, pp. 324–335.

[32] B. Boehm *et al.*, "The roi of systems engineering: Some quantitative results for software-intensive systems," *Systems Engineering*, vol. 11, no. 3, pp. 221–234, 2008.

[33] B. Boehm, L. Huang *et al.*, "The roi of software dependability: The idave model," *IEEE software*, vol. 21, no. 3, pp. 54–61, 2004.

[34] T. M. Khoshgoftaar *et al.*, "Cost-Benefit Analysis of Software Quality Models," *Software Quality Journal*, vol. 9, no. 1, pp. 9–30, Jan. 2001. [Online]. Available: https://doi.org/10.1023/A:1016621219262

[35] C. X. Ling *et al.*, "Predicting software escalations with maximum roi," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*. IEEE, 2005, pp. 4–pp.

[36] M. Ferrari *et al.*, "Roi in text mining projects," *WIT Transactions on State-of-the-art in Science and Engineering*, vol. 17, 2005.

[37] S. Nagrecha and N. V. Chawla, "Quantifying decision making for data science: from data acquisition to modeling," *EPJ Data Science*, vol. 5, no. 1, p. 27, 2016.

[38] S. Biffl *et al.*, *Value-based software engineering*. Springer Science & Business Media, 2006.