2022-02-02

Requirements Dependency Extraction: Advanced Machine Learning Approaches and their ROI Analysis

Deshpande, Gouri

UNIVERSITY OF CALGARY

Requirements Dependency Extraction: Advanced Machine Learning Approaches and their ROI Analysis

by

Gouri Deshpande

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA
FEBRUARY, 2022

Abstract

Dependencies among requirements significantly impact the design, development, and testing of evolving software products. Requirements Dependencies Extraction (RDE) is a cognitively complex task due to rich semantics in natural language-based requirements, which impose challenges in automating the extraction and analysis of dependencies. The challenges intensify further when dependency types are considered. RDE is a part of the extensive decision support system to make effective software release planning, development, and testing decisions.

Recently, Machine Learning and Natural Language Processing techniques have successfully automated tasks in Requirements Engineering to a large extent. Despite this success, there are some challenges to the automation of RDE - 1) Due to the nature of the problem, it is cognitively difficult to identify all the dependencies among requirements; hence generating or procuring high-quality annotations for automation through Machine Learning is an arduous task. 2) In the real-world, unlabelled data is abundant and supervised ML techniques need a training set. Lack of data for training is one of the challenges when using ML for RDE. 3) Textual requirements lack structure due to natural language, and feature extraction (transformation of the raw text into suitable internal numerical representations i.e.feature vector) techniques of NLP lead to ML techniques' success. However, feature extraction method identification and application are cost and effort-intensive. 4) While there is a broad spectrum of Machine Learning techniques to choose from for RDE automation, not all techniques are economically viable in all the scenarios considering data size and effort investment. Hence, there is a need to evaluate the ML techniques beyond just performance measures for effective decision making.

This thesis addresses these challenges and provides solutions. The results described in this thesis are derived from a series of empirical studies on industry and open-source software (OSS) datasets. The main contributions are as follows:

- Performed a comprehensive assessment of Weakly Supervised Learning and Active Learning (AL) to address the data acquisition challenges using public and OSS datasets. Additionally, we compared Active Learning with Ontology-based retrieval (OBR) and further developed a hybrid solution that showed a 50% reduction in the labeling (human) effort for the two industry dataset evaluations from: Siemens Austria and Blackline safety.

- Evaluated and compared a conventional ML-based Transfer Learning and state-of-the-art Deep Learning (DL) method (Fine-tuned Bidirectional Encoder Representations from Transformers (BERT)) for 6 Mozilla products (OSS) to address lack of training data challenge. We showed that the DL method outperformed the within project's conventional ML models by 27% to 50% (on F1-score measure).

- Demonstrated that the state-of-the-art DL method (fine-tuned BERT) could successfully overcome the feature extraction challenge of RDE as fine-tuned BERT outperformed conventional ML methods by 13% to 27% on the F1-score for the Firefox, Redmine and Typo3 product's datasets. Also, we showed that fine-tuned BERT successfully predicted the direction of dependency.

- Utilized a nine-stage ML process model and proposed a novel ROI of ML classification modeling approach. ROI of ML classification showed scenarios when it is viable to utilize complex methods over conventional methods considering the cost and benefits of data accumulation. Utilizing OSS datasets for evaluations and practitioner inputs for cost factors, we showed accuracy and ROI trade-offs in ML approach selection for RDE. Thus, we have demonstrated empirical evidence of ROI as an additional criterion for ML performance evaluation.

Preface

Various key publication included in the thesis and their collaboration status is as follows:

Chapter 2: The survey is covered by Ethics Certificate number REB16-1491_REN2, issued by the University of Calgary for the project "Evaluation, Analysis and Optimization of Software Release Readiness" on 28 August 2019. The survey results are published in the International Software Product Management Association newsletter: https://ispma.org/elicitation-and-maintenance-of-requirements-dependencies-a-state-of-the-practice-survey/

Part of the content from chapter 2 is also published as "Sreyantra: Automated software requirement inter-dependencies elicitation, analysis and learning", 2019 IEEE/ACM International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), of which I am the author.

The preliminary study part of this chapter has been published as "Data-driven elicitation and optimization of dependencies between requirements", 2019 IEEE 27^{th} International Requirements Engineering Conference (RE) and used with permission from Chahal Arora and Guenther Ruhe of which I am the primary author and contributor.

Chapter 4: The content of this chapter has been published as "Requirements dependency extraction by integrating active learning with ontology-based retrieval", 2020 IEEE International Requirements Engineering Conference (RE) and used with permission from its authors, of which I am the primary author and contributor.

Chapter 7: The content of this chapter has been published as "Is BERT the New Silver Bullet? - An Empirical Investigation of Requirements Dependency Classification", International Workshop on Artificial Intelligence for Requirements Engineering, 2021, and used with permission from its authors, of which I am the primary author and contributor.

Chapter 8: The content of this chapter has been published as "Beyond accuracy: Roi-driven data analytics of empirical data", 14^{th} ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). 2020, and used with permission from its authors, and I am the primary author and contributor.

Chapter 9: The content of this chapter has been submitted to a journal, and it is under review. It has been made available on arxiv as "How Much Data Analytics is Enough? The ROI of Machine Learning Classification and its Application to Requirements Dependency Classification" with arXiv preprint, arXiv:2109.14097 (2021), and used with permission from Chad Saunders and Guenther Ruhe as I am the primary author and contributor.

Acknowledgements

The last four and a half years have been the most productive years of my life thus far. Very many people have imparted everlasting teachings on me in this journey. My supervisor Dr.Guenther Ruhe tops this list, who coached me, believed in me, and pushed hard to excel and work towards the goals. I feel blessed to have him as my supervisor, who is my inspiration. I could not have imagined reaching where I am today without my master's supervisor Dr. Saha. He has been pivotal in my career, transforming my thinking and kindling my research interest. I am eternally grateful to him. Dr. Rokne, also on my committee, supported and mentored me all this time. A heartfelt thanks to him for all the guidance he has bestowed upon me all these years. I also want to thank Dr. Nayebi for taking interest in my research and posing tough questions early on, which have helped shape my research. Constructive feedback from my examiners, Dr. Alessio Ferrari and Dr. Mariana Bento have immensely helped me make this thesis better. Thank you for your effort in reading my thesis thoroughly and providing your valuable insights.

My graduate student collaborators, Chahal Arora, Ikagarjot Kamra, Saipreetham Chakka, Behnaz Sheikhi and Dylan Zotegouon, have helped me realize my research ideas. My gratitude to you all for your help and support thus far. Kind regards to Dr. Xavier Franch, Dr. Cristina Palomares, Quim Motger and Katarzyna Biesialska from the University of Catalunya for collaborating with me and teaching me a great deal about research. Industry collaborator, Jason Ho from Blackline safety provided his valuable time and dataset for our research. Thank you for all your patience. Thanks to Dr. Chad Saunders, who taught me entrepreneurship and helped me shape my last paper. Priyaa Srinivasan, my office mate who always made tea for me and hopped in for some intense research-related and random discussions, are the times that I will cherish forever. Thank you. My colleagues Debjyoti Mukherjee and Alireza Ahmadi were a delight to have long chats and research discussion in the SEDS lab. Thanks for being such fantastic mates. Office admins Jackie and Maryam are the best. They were people I could relate to and have a random conversation with at the water cooler. I will cherish those times as well. Jackie, I will remain indebted to you for sharing all those parenting tips and personal experiences. Thanks for your inclusive nature.

Moving to another continent across the globe for PhD was not an easy choice for me. My friends who

always gathered over social media to celebrate my small and big victories helped me cope with stress, thanks you all. My mother, Pratibha Ginde and mother-in-law Vandana Deshpande who always got my back, are the pillars of my support system. Love you for always being there for me. My brothers Prasanna Ginde and Pranav Ginde have been very kind and supportive all my life. Thanks you. My two kids, Avani and Dhruv, are my oxygen. They help me pause and reflect. Thanks for being my source of affection and joy. Sameet, my husband, the epitome of patience, a university topper with multiple patents, has been taking care of everything he can to help me pursue a PhD. I feel blessed to have you on my side. I would not have asked for anything more. Thank you for everything; love you. My dad, Purushottam Ginde, who is blessing me from above is my source of inspiration and guiding light. Miss you baba.

Dedicated to my family, for their unconditional love and support

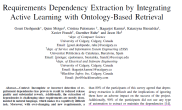
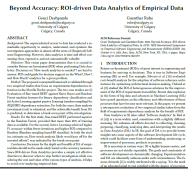
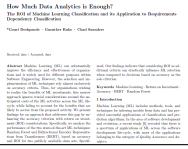
	Publications
PART 1: Exploration	
Preliminary experiment State of Practice Survey	 Deshpande Gouri and G. Ruhe. Elicitation and maintenance of requirements dependencies: A state of the practice survey. https://ispma.org/elicitation-and-maintenance-of-requirementsdependencies-a-state-of-the-practice-survey/
Requirements Dependency Extraction (RDE) agenda (Doctoral symposium)	 Deshpande, Gouri. "Sreyantra: Automated software requirement inter-dependencies elicitation, analysis and learning." 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)
Weakly Supervised Learning for RDE (Preliminary experiment)	 Deshpande, Gouri, Chahal Arora, and Guenther Ruhe. "Data-driven elicitation and optimization of dependencies between requirements." 2019 IEEE 27th International Requirements Engineering Conference (RE). IEEE, 2019.
PART 2: Data Challenge	
Active Learning for RDE	
Hybrid models for RDE	 Deshpande, Gouri, Quim Motger, Cristina Palomares, Ikagarjot Kamra, Katarzyna Biesialska, Xavier Franch, Guenther Ruhe, and Jason Ho. "Requirements dependency extraction by integrating active learning with ontology-based retrieval." In 2020 IEEE 28th International Requirements Engineering Conference (RE), pp. 78-89. IEEE, 2020.
Cross Project Dependency Extraction	
PART 3: Feature Extraction Challenge Fine-tuned BERT for RDE	
Is BERT the New Silver Bullet?	 Deshpande, Gouri, Behnaz Sheikhi, Saipreetham Chakka, Dylan Lachou Zotegouon, Mohammad Navid Masahati, and Guenther Ruhe. "Is BERT the New Silver Bullet?-An Empirical Investigation of Requirements Dependency Classification." In 2021 IEEE 29th Inter. Requirements Engineering Conf. Workshops (REW), pp. 136-145. IEEE, 2021
PART 4: Performance Evaluation Challenge	
Beyond Accuracy: ROI of Data Analytics	 Deshpande, Gouri, and Guenther Ruhe. "Beyond accuracy: ROI-driven data analytics of empirical data." Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). 2020
ROI of Machine Learning Classification	 Deshpande, Gouri, Guenther Ruhe, and Chad Saunders. "How Much Data Analytics is Enough? The ROI of Machine Learning Classification and its Application to Requirements Dependency Classification." arXiv preprint arXiv:2109.14097 (2021).

Table of Contents

List of Figures

List of Tables

Abbreviations

Acronym	Definition
AI	Artificial Intelligence
AL	Active Learning
CH	Challenge
CPDE	Cross Project Dependency Extraction
DA	Data Analytics
DL	Deep Learning
ML	Machine Learning
NL	Natural Language
NLP	Natural Language Processing
OBR	Ontology-based Retrieval
OSS	Open Source Software
PL	Passive Learning
RD	Requirements Dependency
RDA	Requirements Dependencies Analysis
RDE	Requirements Dependencies Extraction
RDE-BERT	Fine-tuned BERT model using RD dataset
RE	Requirements Engineering
ROI	Return of Investment
TL	Transfer Learning
WPDE	Within Project Dependency Extraction

Epigraph

In my pursuit of joy:

विद्या ददाति विनयं विनयाद्याति पात्रताम् ।
पात्रत्वाद्धनमाप्नोति धनाद्धर्मं ततः सुखम् ॥ ५ ॥

vidyA dadAti vinayaM, vinayAdyAti pAtratAM |
pAtratvAddhanamApnoti, dhanAddharmaM tataH sukhaM || 5 ||

- Hitopadesha, *Narayana Pandit, 14^{th} century scholar*

Meaning: Knowledge makes one disciplined and humble, humility begets worthiness, from worthiness one creates wealth and enrichment, enrichment leads to right conduct, from that (comes) joy (and contentment). (translation curtsy: https://blog.practicalsanskrit.com/2009/07/knowledge-leads-to-happiness.html)

Chapter 1

Introduction

Requirements Engineering (RE), referred to as the secret weapon for better AI and better software [23], has received considerable attention in research due to its implication on software project success [69]. RE is an early life-cycle process that precedes analysis, design, coding and testing. It has been regarded as decision-centred, and recent analytics trends have enabled proactive decision making for developing software-intensive products further [104]. To aggregate the pain in RE two successive survey runs with industry (practitioners) [159] [69] [45] were carried out. They confirmed that the requirements are primarily documented in natural language (textual form). Also, managing changing (over time and frequently) requirements is a continuous process handled mainly through manual efforts.

Dependencies among requirements are concerned with the relationships between requirements and their impact on various activities in the software development process [43]. There are a variety of dependency types such as *Requires*, *And*, *Or*, *Precedence* etc. Few of the sample requirements pairs of the *Requires* dependency are as shown in Table 1.1. The term "dependency" can have a different connotation in implementation space, feature efforts space, feature value space and feature usage space [133]. Requirements Dependency Extraction (RDE) is process of identifying the relationship and its type between a pair of requirements.

Table 1.1: Sample dependency pair from Firefox dataset: *Requires* is a directed dependency and the dependency direction is from left to right in the following examples.

Dependency type	**ID**	**Description**	**ID**	**Description**
Requires	1432952	add ability to associate saved billing address with payment card in add/edit card form	1429180	option to use new billing address when adding new payment card
	1394451	update illustration for error connection failure	1358293	ux error connection failure copy design and illustration update
	1524948	introduce session group to allow to manage multiple session at same time	1298912	multiple snapshot perform periodic session backup and let user restore particular backup

Other industry surveys [157], [36] conducted to understand the pain of RDE confirmed that more than 80% of the requirements are in some form of dependency with others.

Practical Relevance: Figure 1.1 is an illustration of the practical relevance of considering requirements dependencies for incremental and iterative software development. Having multiple release cycles: R_{i-1}, R_i, R_{i+1} defines the order of implementing and testing new or updated features. However, if a requirement is implemented in a release R_i but requires a requirement implemented in a later release, then the requirement will not be usable. Similar arguments hold for two requirements that are related to each other but are implemented in different releases. Thus, identifying the dependencies early on is crucial as it drives the implementation as well as testing and rework efforts immensely.

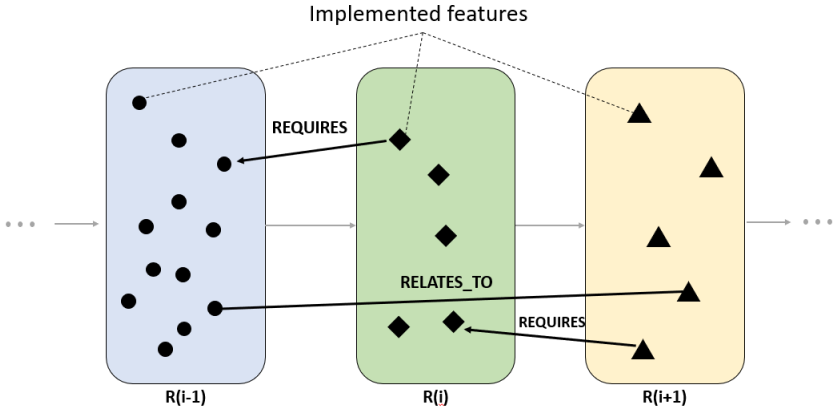


Figure 1.1: Requirements dependencies across various releases of a project

Changing requirements: Software development is an incremental, iterative and dynamic process. 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 [135]. Additionally, such change propagation challenges the developers because it consumes substantial efforts due to the natural language in requirements' definitions. Hence, it is crucial to know/extract all possible relationships that could occur among the requirements for a product's success [58].

Disregarded dependencies: Ignoring or not taking dependencies into account early on could potentially result in rework in the design, development and testing of the product's life cycle. This additional effort increases the investment and decreases the value of the product. Moreover, some of the dependent features might be deemed unusable and dysfunctional in the absence of others. For example, disintegrating a user story/scenario into a meaningful sequence and combining software features is a non-trivial effort that can not be achieved if dependencies are disregarded or missed.

Magnitude of the problem: For pair-wise dependencies, just with n = 50 requirements, assuming it takes one minute to verify one of the possible n*(n-1)/2 dependencies, it would take approximately 20 person-hours effort. This effort will grow drastically due to new or changed requirements, given the dynamic nature of software evolution. Thus, it is explicable that given the number of potential interdependencies for any small or large software application,

manual dependency identification is a time, effort and cost-intensive task. Even if requirements are well written, domain expertise is inevitable in identifying the dependency types since it demands specific software products related information and domain knowledge.

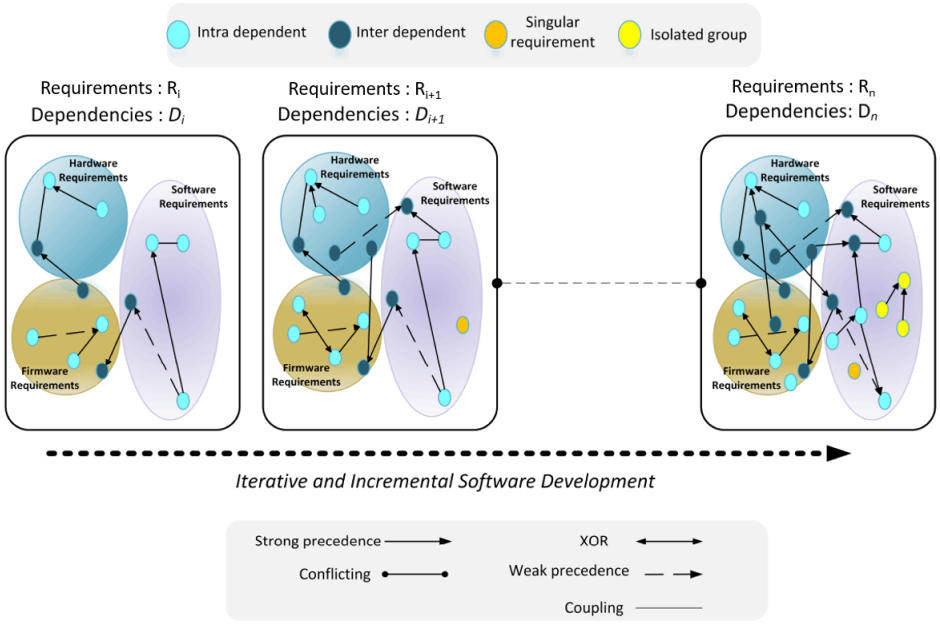


Figure 1.2: Representative requirements interdependency graph and their evolution over a period of time. This is an example product of our collaborator (Blackline Safety), which has three different components: firmware, software and hardware.

Iterative and incremental software development: Figure 1.2 is an illustration of a scenario in the incremental and iterative software development life cycle which has hardware as one of the components apart from software and firmware. In every product development iteration, new and existing features (also termed as requirements in this context) evolve, triggering modifications to dependencies. In Figure 1.2, a product with dark blue coloured nodes represents the requirements that have inter-project dependencies. i.e. the software requirements have some form of dependency with requirements from hardware and firmware parts of the product. These can be various structural dependency types such as strong precedence, weak precedence and coupling. The yellow coloured nodes represent the isolated groups of requirements, whereas the dark yellow solo node is the singular requirement. Any changes to these dependency types can substantially impact the neighbouring dependencies as depicted in this sub-network. This impact could relate to structural and changes in the nature and strength of dependencies. Additionally, new requirements and change requests can lead to change in dependencies, which could cause conflicts within requirements and increase or decrease the importance of other dependent requirements.

Research in Requirements Engineering focusing on dependencies and their types has a long history of studying dependencies related to syntactic and semantic criteria. While some authors have proposed taxonomies focusing on requirements engineering in general [128] [44], others focused on application areas such as release planning [37]. For example, Dahlstedt [44] provided the classification of most fundamental dependency types such as Requires, Refines,

Similar_to, Increases/Decreases_value_of, and classified them into Structural and Cost/Value interdependencies. Recently, Zhang et al. [174] consolidated these taxonomies in the context of change propagation analysis and proposed a model that included nine types of dependencies.

Although there have been studies for dependency type identification in the literature [124] that used similarity-based measures, in the recent past, various other studies have explored diverse computational methods that used Natural Language Processing (NLP) [124], fuzzy logic [123], predicate logic [176] and machine learning [139] and Deep Learning (DL) [78] techniques to address Requirements Dependency related research. Most of these studies thus far focus only on the supervised ML techniques to address RDE. However, these pave the path to advanced technology adaptation for RE-related problems.

Recent advancements in the Natural Language Processing (NLP) field have taken various computer science domains by storm. Requirements Engineering is not an exception. Various RE-relevant sources such as Software Requirements Specification documents, tweets, app reviews etc have been explored in the context of traceability [78], requirements retrieval [121], functional and no functional requirements classification [84] etc. NLP for RE research has attracted over 350 scholarly articles since 2004 across 170 different publication venues [177]. Findings from a mapping study [177] have shown that most recent NLP technologies such as Word embeddings, Pre-trained models have not been fully explored in the RE world, and these pose a direct impact on the progress of NLP for RE research.

Machine Learning (ML) enables the development of efficient and effective systems through automation. RE is not an exception as utilizing state-of-the-art NLP techniques for RE overlap with ML to a large extent [59], hence, ML for RE is an umbrella term that even covers a more significant part of current NLP for RE research [173]. However, there are several challenges such as data, tools and feature extraction mechanisms that hinder applied NLP and ML in the context of RDE.

1.1 Challenges and Approaches

Exploring recent studies helped us gauge the current standing in the field of RDE (Section 2.1). Findings showed that RDE automation through applied ML is in the nascent exploration stage. Further, our state of practice survey (Section 2.2) reinstated the need for RDE and sifted various challenges and opportunities specific to RDE. Finally, the results from the preliminary study (Section 2.3) were encouraging with regards to applied ML for RDE. Closer scrutiny of this empirical study revealed various challenges for enhanced automation performance addressed by this research.

List of challenges and corresponding research questions are as follows:

1. **Data:** Due to the nature of the problem, it is cognitively difficult to identify all the dependencies that could exist among requirements; hence generating or procuring annotations for automation through Machine Learning is an arduous task. Additionally, in the real-world, unlabelled data is abundant and a good (well-performing) supervised ML model needs a good training set (annotations that are representative of the dataset) [30] [62].

Lack of data for training set adds another dimension to data challenge when using ML for RDE.

(a) Knowledge acquisition bottleneck: RDE automation through ML methods or Ontology could provide decision support for decision-makers. However, knowledge acquisition (labeled data) representing the problem domain is the bottleneck for applied ML and Ontology-based solutions. While advanced ML approaches such as Active Learning (human in the loop approach to annotation gathering) can be one method based on utilizing minimal labeling effort by oracles (example: Human expert), such that the labeling cost of training good model is minimized [141] [17]. For the Ontology-based approach, investing efforts in designing ontology could be another method [80] [115].

Challenge 1: Due to the nature of the problem, it is cognitively difficult to identify all the dependencies among requirements; hence generating or procuring high-quality annotations for automation through Machine Learning is an arduous task.

Thus, we formulated and evaluated Research Questions (RQ) to tackle this challenge as

RQ 1: How effective are Weakly Supervised Learning (WSL) and Active Learning (AL) methods to overcome the challenge of data acquisition for RDE?

Approach: We evaluate advanced ML approaches: WSL and AL for RDE using public and OSS datasets. Also, compare AL with Ontology based-extraction and explore AL and Ontology-based hybrid models for RDE automation which are not explored in RE.

(b) Limited or scarce training data: Lack of train set is an inherent problem while utilizing supervised ML methods for RDE, especially for smaller projects in the early development phase. Transfer Learning [109] is a method to leverage data of a project to classify another similar project with limited data for training. This method will facilitate an enhanced classification mechanism and overcome data crunch-related issues. However, this has not been explored in RDE until now.

Challenge 2: Although supervised ML techniques could be the first choice for automation, in the real-world, unlabelled data is abundant and supervised ML techniques need a training set. Limited training data is one of the challenges when using ML for RDE.

Thus we formulate the following research question and propose an approach to overcome this challenge.

RQ 2: How effective is the Transfer Learning approach, which has been widely used in defect and effort prediction, for RDE?

Approach: Utilizing data from the larger project (source) to train ML technique and predict for smaller projects(target) could leverage annotated data availability of a larger project to train ML and then use for predicting dependencies of smaller target projects which lack training data for automation.

2. **Feature extraction:** Textual requirements lack structure due to natural language, and feature extraction techniques of NLP lead to ML techniques' success. Feature extraction is to transform the raw text into suitable

internal numerical representations i.e.feature vector from which a text classifier can learn and classify patterns in the input [177]. A recent mapping study for ML in RE [173] looked at over 65 scholarly articles published over the last decade. Their findings showed that selecting the ML technique for a given problem does not follow any specific criteria and that its success is driven by effective feature extraction. However, feature extraction method identification and application are cost and effort-intensive.

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 [11], [73], [46], [84]. The pre-trained BERT model is trained on massive unlabeled data (such as Wikipedia) over different pre-training tasks. This off-the-shelf BERT model (pre-trained on a large text corpus) can be fine-tuned further on specific tasks by providing only a small amount of data. Such Transfer Learning enabled through fine-tuning of BERT models has been successfully used in RE-related tasks recently [84] [11].

BERT eliminates the need for formal NLP-based pre-processing and feature extraction since the model represents the steps of the traditional NLP pipeline in an interpretable and localizable way internally and adjusts the NLP pipeline dynamically [59] [84] [154]. Essentially, BERT can automatically detect the features (textual representations) needed for classification or detection, which, when learned by the classifier, allows better generalization even over new or (and) unseen data [177] [84].

BERT has been successful at many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI), where it is crucial to understand the relationship between two sentences, which is not directly captured by language modeling [59]. RDE problem fits this scenario very well. Also, there has been evidence that the DL language models can encode a range of syntactic and semantic information abstractions traditionally believed necessary for language processing and that they can model complex interactions between different levels of hierarchical information [154]. Apart from utilizing BERT for Transfer Learning, in this thesis, we utilize BERT mainly to address the feature extraction (language modeling) related challenges concerning RDE.

The most frequently utilized methods in the existing RE literature are Bag of Words, Parts-of-speech tagging [177] with an inclination to using DL-based techniques. Also, based on the preliminary study, it was evident that selecting a good NLP feature extraction technique could substantially enhance the performance of the ML method.

Challenge 3: Textual requirements lack structure due to natural language, and feature extraction (transformation of the raw text into suitable internal numerical representations, i.e.feature vector) techniques of NLP lead to ML techniques' success. However, feature extraction method identification and application are cost and effort-intensive.

Thus we formulate the following research question and propose an approach to overcome this challenge.

RQ 3: Does the state-of-the-art Deep Learning pre-trained language models solve the feature extraction prob-

lem for RDE and improve the ML method's performance?

Approach: Using off-the-shelf pre-trained BERT models and further fine-tuning them using requirements dependency dataset, we explore the solution for the feature extraction problem for RDE.

3. **ROI analysis:** While there is a broad spectrum of Machine Learning techniques to choose from for task automation, not all techniques are economically viable in all the scenarios considering training data size and effort investment. When in doubt, simple ML techniques are recommended over complex ones [62], [30]. However, what if there was a method to estimate such investments and compute the benefits when other complex algorithms are chosen? Interestingly, scholarly research exploring ROI of ML is scarce [119] [71]. However, a few discussions and approaches have been proposed in the gray literature, especially on the business front, to weigh the benefits of AI/ML-based software solutions in terms of economic value (dollars). For instance, [167] emphasizes the need for ROI as a measure for evaluating the benefits of ML. On the other hand, in another blog [7], a method is proposed to arrive at ROI economic value, which emphasizes the cost incurred due to prediction outcomes of the ML technique. Although these approaches are intriguing and align well with our agenda, they lack empirical rigour, evaluation and do not consider the cost of data accumulation and pre-processing, which is considered to be a large part of applied ML effort [72].

   Hence, there is a need to evaluate the ML techniques beyond just performance measures for effective decision making. In other words, it is necessary to understand how far one should chase the accuracy parameter when the cost of achieving the differential is substantial.

   Challenge 4: While there is a broad spectrum of Machine Learning techniques to choose from for RDE automation, not all techniques are economically viable in all the scenarios considering data size and effort investment. Hence, there is a need to evaluate the ML techniques beyond just performance measures for effective decision making.

   Thus we formulate the following research question and propose an approach to overcome this challenge.

   RQ 4: What is the benefit of using ROI as an additional criterion for ML classification performance evaluation?

   Approach: We propose an ML process model and associate cost factors to various stages of this process and also consider the ML classification performance to model the ROI of an ML technique. Through various empirical evaluations for cost factors from practitioners, we tried to answer the question, " How much data analytics is enough?"

Figure 1.3 shows the summary of premises of each one of the challenges, their datasets, measures and methods used in this thesis. The results presented from various empirical evaluations performed on OSS and industry datasets have demonstrated that RDE automation is possible using various advanced ML approaches to a large extent. One significant evidence that emerged from this research is the trade-off between ROI and accuracy measure can impact

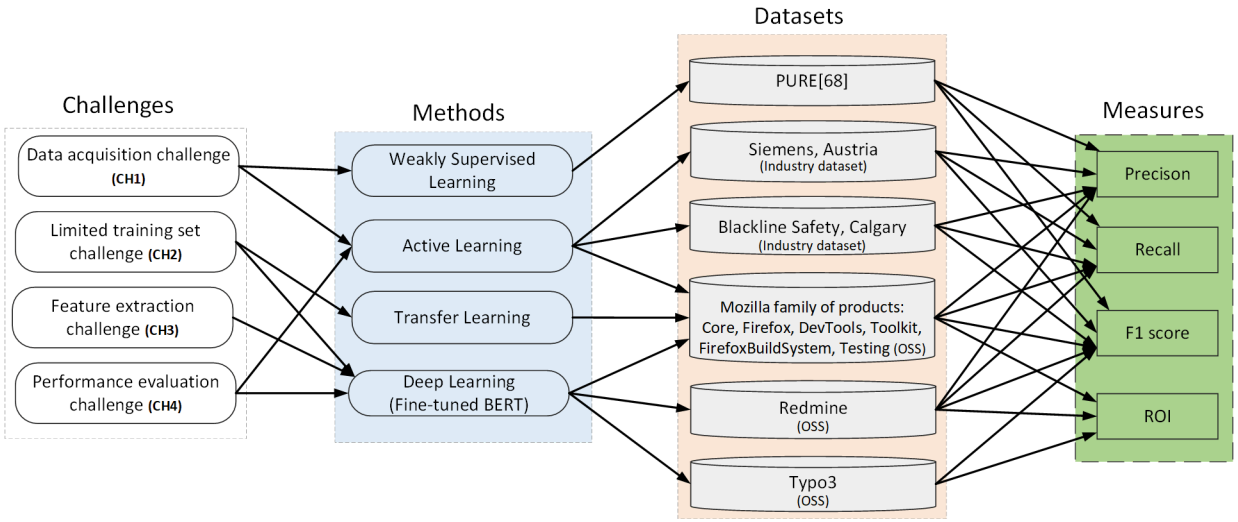


Figure 1.3: Various dimensions of my research work and their interconnections: All the methods listed were compared with the Supervised Learning (a.k.a conventional ML) methods for several datasets and varying performance measures

the ML method selection for RDE.

1.2 Primary Terminologies and Definitions

Dependency: A pair of requirements are called *Dependent*, if there is at least one type of dependency such as *Requires*, *Similar*, *Or*, *And* and *XOR* between them. Otherwise, they are *Independent*

Requires: The fulfilment of one requirement (say, *A*) depends on the fulfilment of another requirement (say, *B*) Then *A Requires B* is the relationship between them. It is a special form of *Dependent* relationship. Alternatively, it is also referred to as (B *Precedes* A) [174] [42].

And: If a pair of requirement are required in conjunction then they are in a relationship called *And* [36].

Similar: If a pair of requirement are semantically similar, then they are in a relationship called *Similar* [37].

Other : *Other* type of dependency is when a requirement pair is *Dependent* and the dependency type is not *Requires* (could be any of the other dependency types listed in this section)

Relates_to : if implementation of one requirement during development impacts the other one then requirements are in a relationship called *Relates_to* [5].

Refines : If a target requirement is defined in more detail by another requirement then they are in a relationship called *Refines* [128].

***Requirements Dependency Extraction*:** *Requirements Dependency Extraction* (RDE) is the process of identifying the relationship (dependency) and its type between a pair of requirements. While binary RDE classifies

Dependent and *Independent* requirements pairs, multi-class RDE classifies three or more dependency types such as *Requires*, *Similar*, and *Independent*

***Advanced Machine Learning approaches*:** Weakly Supervised Learning, Active Learning, Transfer Learning and Deep Learning techniques are grouped under one umbrella term called *Advanced Machine Learning approaches* [64]

Conventional ML methods : Traditional or *Conventional ML methods* are Supervised Machine Learning methods such as Naive Bayes, Random Forest and Support Vector Machine, which are generally used in text classification [173] [121].

***Weakly Supervised Learning* (WSL):** *Weakly Supervised Learning* combines the benefits of supervised learning and unsupervised learning [178]. WSL is an umbrella term covering a variety of techniques, which attempt to construct predictive models by learning with weak supervision. One such approach is *conservative co-testing* strategy [118] where, for each iteration, an unlabeled example is labeled if the two or more classifiers agree on the labeling [152].

***Active Learning* (AL):** *Active Learning* is a method based on utilizing minimal labeling effort by oracles (example: Human expert), such that the labeling cost of training good model is minimized [141] [17]. In our research, when data is already labeled, for AL, we pretend they are unlabeled until queried and labeled by a simulated oracle.

***Passive Learning* (a.k.a random sampling):** *Passive Learning*, also known as random sampling, is when data for training is sampled randomly from a data pool [141]. This is a general approach taken while training in Supervised Machine Learning methods.

***Self-training*:** A learner is trained with a small labelled data set first, and then it is used to classify the unlabelled data. Typically, the most confident unlabelled instances and their predicted labels are added to the training set, and the process repeats.

***Transfer Learning (TL)*:** *Transfer Learning* is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem [93]. In the context of RDE, we adapt this definition as to when there is insufficient data to automate RDE; Transfer Learning can be used to transfer dependency extraction knowledge learned from other source projects S to the target project T.

***Cross project dependency extraction (CPDE)*:** *Cross project dependency extraction* can be viewed as a specific case of Transfer Learning, which extracts dependency knowledge from a source project and transfers it to a target project.

***Within project dependency extraction (WPDE)*:** *Within project dependency extraction* is using data from within the same project for RDE.

***Version-wise CPDE*:** Cross-version defect prediction [105] has been explored in the literature wherein the training data from different versions of a product are used to train a classifier and predict the defects for a newer or updated version of the product. Motivated by this, we simulated a scenario of CPDE, referred to as *Version-wise CPDE* for a target project under development when the data for training is available only over the time of the software development life cycle.

***RDE-BERT*:** Bidirectional Encoder Representations from Transformers (BERT), a Deep Learning-based state-of-the-art language model, pre-trained on a large textual corpus such as English Wikipedia has been successful at various NLP tasks such as Question Answering and Natural Language Inference [59]. These pre-trained, Off-the-shelf BERT models (pre-trained on a large text corpus) can be fine-tuned further on specific tasks by providing only a small amount of data [11], [73], [46], [84]. Such fine-tuned BERT using RDE specific data is referred to as *RDE-BERT*.

1.3 Research Methodology

Based on research methodologies described by Roel Wieringa [165], and Easterbrook et al. [147], I used various empirical methods in this thesis to evaluate research questions.

1.3.1 Empirical Methods

Qualitative Survey

Findings from interviews or document analysis can serve as a starting point for a case study by both setting the context for the researchers as well as identifying important issues and variables for the study [69] [68] [157]. To identify the essential open issues, set the context of this research and starting point of various case studies and experiments for RDE, I conducted a survey [147]. This Google survey consisting of qualitative and quantitative questions was circulated through newsletters and personal contacts. Over 75 practitioners in various roles took this survey. This was used as a pre-study to find out opportunities and risks for further research.

Apart from surveys, case studies, an empirical method for investigating phenomena in the real-world environment, and experiments, an empirical method that involves artificial control variables, are the most commonly used research methods for empirical research [175], [147]. These two methods were utilized in this research.

Case Studies

I conducted several lightweight case studies to evaluate our research questions utilizing various publicly available and Open Source Software (OSS) datasets.

- Evaluated Weakly Supervised Learning for RDE, a pilot study (preliminary case study) to verify the feasibility of possible automation using ML, using PURE [70] public data set.

- Evaluated fine-tuned BERT for the RDE automation using Redmine [5] OSS as a case study and evaluated research questions.
- Used the Firefox [2] dataset as a source for exploring ROI as an additional criterion to evaluate ML methods for RDE.

Experiments

We propose ROI modeling in this thesis. In order to evaluate its applicability, we utilized various values for cost and benefit factors for experiment purposes based on our own experiences. After a thorough evaluation of the model, we proceed towards industry evaluations. The preliminary study needed data to evaluate the ML method; hence, two students from the Software Engineering course were employed for manual annotation. We utilized Cohen's kappa to measure agreement [121] between the annotators to generate the final train set.

1.3.2 Conceptual Frameworks

ML Process Model

While evaluating the ROI of ML in the context of RDE, we proposed a detailed ML process model. Although ML has been explored widely in Software Engineering, there is no formally defined process model. We think this is the first step towards that since ML in SE and SE for ML have been an area of interest in recent times.

ROI Modeling for ML

Exploration of various ML methods to achieve better accuracy lead us to contemplate cost versus benefit analysis of various ML methods for RDE. As such, we associated cost to various stages of ML Process model and ML prediction results (TP, FP & PN) and overall value a product would generate as a beneficial factor to model ROI of ML

1.3.3 Data Analysis

The following subsections provide details regarding the quantitative data analysis methods used from various forms of Machine Learning in this thesis.

Supervised Learning: For various empirical analyses performed in this study, we use three ML algorithms, namely: Naive Bayes (NB), Random Forest (RF) and Support Vector Machine (SVM) since they have performed well in the RE related tasks [121] [173].

Weakly Supervised Learning: Weakly Supervised Learning (WSL) [178] combines the benefits of Supervised Learning and Unsupervised Learning. It is motivated by the high cost of data labeling. There are various strategies for using unlabeled data to improve the performance of conventional (supervised learning) ML methods, especially when a small amount of labeled data is available, which is insufficient to train a good learner, while abundant unlabeled data are available. WSL is an umbrella term covering a variety of techniques, which attempt to construct predictive

models by learning with weak supervision. This approach is a form of *conservative co-testing* strategy [118] where, for each iteration, an unlabeled example is labelled if the two classifiers agree on the labeling [152].

Active Learning: In the real world, labeled data is scarce, and labeling is a time-consuming effort. At the same time, unlabeled data is easy to accumulate and available in abundance. Active Learning (AL) systems attempt to overcome the labeling bottleneck by asking queries in the form of unlabeled instances to be labeled by an oracle (e.g., a human annotator). Hence it is also referred to as human-in-the-loop ML. Unlike random sampling (Passive Learning) technique in which data is selected randomly for training ML, AL interactively queries the user to obtain the essential unlabeled data point [141]. In this way, the AL aims to achieve high accuracy using as few labeled instances as possible, thereby minimizing the cost of obtaining labeled data. The fundamental hypothesis is that if the learning algorithm chooses the data from itself then it will perform better with less training.

There are three different types of AL scenarios in the literature, and we have used pool-based sampling in our research because it has been used in the real-world problem domains of ML with textual content-based datasets [142]. Pool-based sampling assumes that there is a small set of labeled data L and a large pool of unlabeled data U available. Queries selectively drawn from the pool are usually assumed to be static or non-changing. Typically, instances are queried greedily, according to an informativeness measure used to evaluate all instances in the pool. We have used the three most commonly used informativeness evaluation mechanisms called Uncertainty sampling techniques, namely: Least confidence, MinMargin and Entropy [60]. Uncertainty sampling techniques target the data points near the decision boundary in the current ML model. For example, in the binary classification scenario, the most uncertain data points with a probability close to 50% will lie near the decision boundary.

1. Least Confidence: selects those instances with the lowest confidence level to be labeled next.
2. Smallest Margin: selects the instances where the margin between the two most likely labels is narrow, meaning that the classifier struggles to differentiate between those two most likely classes.
3. Label Entropy: the learner queries the unlabeled instance for which the model has the highest output variance in its prediction

Transfer Learning: Transfer learning is an ML method to leverage a project's data to classify another similar project with limited data for training. This method is an approach to overcome constraints on the training set size, which is widely investigated in the software defect prediction [109] and effort estimation [110] literature. We evaluate the influence of cross-project (transfer) learning in the context of RDE in the advent of smaller training sets.

Figure 1.4 shows the high-level architecture of TL workflow. Data for source projects are first accumulated from repositories in the raw form - both metadata and data for the respective fields of interest. Further, through the python program, relationship information is used to generate dependent and independent requirements pair tuples: $< R1, R2, Dependencytype >$.

ML model is built for a source project, which is then used to predict the classes for other project instances (target projects), similar to that of the source project.

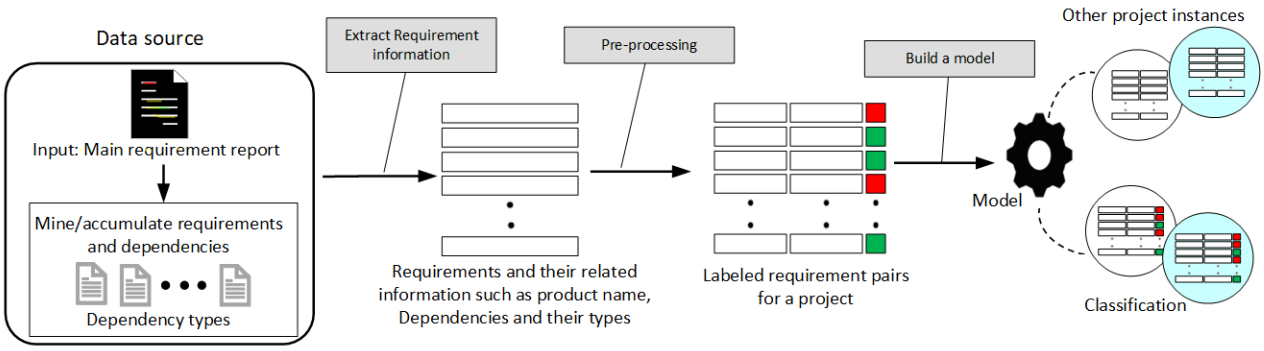


Figure 1.4: Transfer Learning concept in the context of RDE, model trained on a (source) project is used to predict for other (target) project instances

BERT: Recently, Bidirectional Encoder Representations from Transformers (BERT) has shown tremendous success at solving NLP tasks such as Question answering and Natural Language Inference where there is a need to understand the underlying context and the relationship between the sentences [59] [84] [154]. In the recent RE studies, BERT has been explored widely to classify functional and non-functional requirements [84], classifying trace links [81] and determining similarity between requirements [11].

Apart from utilizing BERT for Transfer Learning, in this thesis, we utilize BERT mainly to address the feature extraction (language modeling) related challenges concerning RDE. 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 [59]. This task is Requirements Dependency Extraction (RDE) for our study.

We used pre-trained base model BERT for our research which was fine-tuned further using our RDE-specific dataset. The result is a fine-tuning BERT model called *RDE-BERT*. We use BertForSequenceClassification from the huggingface PyTorch library [87] for this implementation.

In every instance, for a given training set size, RDE-BERT was trained through three epochs with a batch size of 32 and a learning rate of 2e-5. The training set was divided into 90% for training and 10% for validation in each epoch.

Ontology: Ontologies are widely used in RE research (see [48] for a survey). They are a well-suited conceptual artefact to manage knowledge. Particularly, domain ontologies [107] provide a formal representation of a specific domain serving as a means for communication and agreement. Hence, their use in activities such as RDE is a reasonable choice to get domain-specific solutions. In the latest study, Guan et al. [77] showed how ontology and semantic web technology could be used to automate RDE.

The ontology defines dependency relationships between specific terms related to the domain of the requirements. Using this information, it is possible to apply NLP techniques to extract meaning from these requirements and relations. Further, ML techniques could also be applied for conceptual clustering to classify the requirements into the defined ontology.

In this research, we compared Ontology-based RDE with AL based RDE and generated a hybrid model to

overcome the disadvantages of the respective approaches for two industry data sets. Hence the ontology was built using the W3C Web Ontology Language (OWL[1]) for these experiments. The Ontology-based RDE tool is called OpenReq-DD, which is exposed as a RESTful service with an API[2] to provide the required data and perform the dependency extraction [114].

Evaluation measures:

Following are various evaluation metric computations and measures used in this research work. These measures are chosen due to the nature of the RDE problem, which is NLP-centered. Also, a recent survey of ML for RE [173] found that precision, recall, and F1-score (harmonic mean of precision and recall) are widely used for information retrieval tasks [29] such as RE.

- **Confusion matrix:** A confusion matrix[3] 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 1.2 shows the principal entries of CM for a binary class classification.

Table 1.2: 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 (1.1, 1.2, 1.3)

$$Precision = \frac{TP}{TP + FP} \tag{1.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{1.2}$$

$$F1 = \frac{2 \times PrecisionRecall}{Precision + Recall} \tag{1.3}$$

  Missing dependencies could be detrimental for planning in iterative and incremental software development; thus, tracking recall is of interest to us. However, a large amount of a specific dependency type being classified as another might generate overheard work for the decision-makers. Thus both these aspects are important. F-score corresponds to the trade-off value of precision and recall, and thus we chose to utilize it as an indicator for ML performance evaluation.

  Daniel Berry [25] highlights that the need to focus on recall is far more crucial than the precision, because to find the missing information, a human has to do the entire task manually anyway. However, we want to highlights the importance of procuring high quality training data for achieving near 100% recall which is an arduous task in itself. Also, as emphasized by Daniel Berry, the recall of any automation tool if higher than the recall of a human doing the task manually could be accepted as a part solution, realistically. This is certainly

[1] https://www.w3.org/OWL
[2] https://api.openreq.eu/dependency-detection/swagger-ui.html
[3] We utilize values from FP and FN in our study

our goal in this research. We do not envision to replace the human experts, rather, provide them with a decision support system for reducing efforts and cost involved in RDE to a large extent.

- **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 (1.4). Both $Benefit$ and $Cost$ are measured as human effort in person-hours.

$$ROI = (Benefit - Cost)/Cost \tag{1.4}$$

1.3.4 Data Accumulation

For various empirical studies, I used datasets from different sources listed as follows:

1. European Rail Track Management System software specifications from public dataset: PURE [70]. We used this dataset for RQ1 evaluation.
2. Data mined for 64 Open Source Software Projects from Bugzilla [2] bug tracking repositories using respective REST API and used for empirical evaluations of all the four RQs.
3. Data mined from Redmine [5] bug tracking tool for two projects: Typo3 [10], Redmine [5] were utilized to evaluate RQ3 and RQ4.
4. Additionally, data from our industry collaborators: Simens Austria and Blackline Safety, were also procured for specific evaluations for RQ1.

1.4 Thesis Contributions

The core research question that guides this thesis is stated as follows: How well can the requirements dependencies and their types from textual content (i.e. structured or unstructured) extraction be automated using advanced ML approaches, and what is their ROI? **To address this question, in this research**, we propose a novel comprehensive tool called *SReYantra*. "One size does not fil all" also applies to RDE. This prototype tool is designed to facilitate different types of learning, depending on the problem's configuration. ROI analysis provides additional guidance on which technique fits best in what context. *SReYantra* tool (under development) explores the textual content to infer underlying dependency information based on NLP. It utilizes advanced ML techniques such as Weakly Supervised Learning, Active Learning, Transfer Learning and DL language model: BERT (Bidirectional Encoder Representations from Transformers) as its various components for automating dependency extraction. It also provides a mechanism to compute the ROI of ML algorithms to present a clear picture of trade-offs between cost and benefits.

The term *Yantra* in `SReYantra`[4], means a machine or a systematic broadly applicable "system, method, instrument, technique or practice". Since we propose one such method for research related to **S**oftware **Re**quirements, hence the name `SReYantra`. Figure 1.5 depicts the premises of my research work. As shown, this thesis consists of

[4]https://en.wikipedia.org/wiki/Yantra

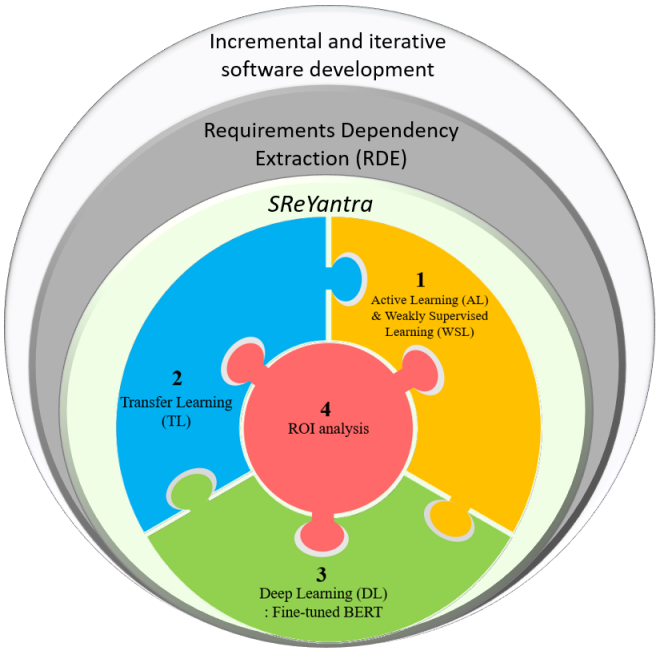


Figure 1.5: Birds eye view of premises of my research work and high level interconnection of various components

four main components of incremental and iterative software development focusing on RDE, which address the four challenges.

Research contributions are as follows:

1. **Addressed knowledge acquisition challenge through WSL, AL, and hybrid solutions (CH1):** Explored WSL and AL methods to accumulate the annotated data. The explorations on public dataset [70], and OSS datasets: Mozilla family of projects [2] showed improved results compared to conventional random sampling-based methods. Further, we compared the AL approach with Ontology-based retrieval and eventually proposed a hybrid of these two methods as a practical approach to address the RDE problem. The outcome of this study demonstrated that the human efforts in RDE could be reduced by 50% for the two industry datasets: Blackline Safety, Canada and Siemens, Austria, evaluations.

2. **Addressed lack of training data using Transfer Learning (CH2):** We analyzed the six Mozilla family products (OSS) for binary and multi-class requirements dependency classification and showed that conventional ML methods were pessimistic (could not comprehend the relationship between the underlying data effectively) for Transfer Learning. We also evaluated the DL-based method, demonstrating that Transfer Learning could yield 27% to 50% better performance for the F1-score measure.

3. **Addressed feature extraction challenge using DL methods (CH3):** Fine-tuned BERT (DL based method) for RDE specific data and showed that it could overcome the feature extraction challenge. Results from analysis of Firefox, Redmine and Typo3 datasets outperformed conventional ML methods by 13% to 27% for the F1-score measure. Additionally, fine-tuned BERT successfully identified the direction (for example, A

Requires B then the direction of the dependency would be A$\longrightarrow$ B) of the dependencies with F1-score over 70% and outperformed conventional ML methods by 90% for Firefox dataset.

4. **Proposed novel ROI modeling and ML process model for performance evaluation (CH4):** Effectiveness of the ML methods are generally evaluated based on their performance measures such as accuracy and F1-score. In this thesis, I propose ROI as an additional criterion for evaluation by weighing cost as an investment in ML implementation and benefit as the reward of using ML.

   Through the three empirical evaluations on three open-source software: Firefox [2], Redmine [5], and Typo3 [10] using practitioners inputs for cost factors, I demonstrated that focusing on improving accuracy using advanced ML methods might not generate value for the additional effort spent in data accumulation and pre-processing. Specific findings are:

   - **Firefox:** AL showed the best ROI performance in the early iterations (with smaller datasets) compared to random sampling. For training set sizes of at least 40% of the whole set, the DL method performed better than the conventional ML method in terms of accuracy and ROI. In both the scenarios chasing a higher F1-score reduced the ROI. Also, after three iterations for Active Learning vs Passive Learning, Active Learning showed the best ROI= 4.5 and F1 = 0.6. However, ROI degraded over the iterations, although accuracy improved.
   - **Redmine:** Fine-tuned BERT (DL method) needed at least 45% or more data to generate positive ROI. With just about 25% to 40% data available for training, conventional ML and DL methods performed about the same and generated negative ROI.
   - **Typo3:** Fine-tuned BERT approached the 80% benchmark accuracy with approximately 50% of the training data while conventional ML required 70% training data to attain the same level of accuracy. However, conventional ML outperformed the DL method for smaller train sets (incurring lower negative returns). At the same time, positive ROIs were observed only for the more extensive train set at which the DL method was consistently better than the conventional ML method.

   These outcomes strongly affirmed that, beyond accuracy, there is a need to use additional criteria such as ROI for ML performance evaluation.

5. **ML based tools for RDE:** All the modules (independently implemented sub parts, thus far) of the *SReYantra* tool developed as part of this research have been made available in public repositories[5]. We have made Mozilla[6] and Redmine[7] datasets publicly available.

Table 1.3 lists relevant publications of my thesis. **My other publications not included in this thesis are as follows**

[5]https://git.io/JDvFq
[6]https://doi.org/10.5281/zenodo.5914956
[7]https://doi.org/10.5281/zenodo.5044654

Table 1.3: List of relevant publications and corresponding challenges mapping of my research: one survey (S), five conference (C), and one journal paper (J)

		Challenge	**Status**
Survey [55]	Deshpande, Gouri and Ruhe, Guenther. "Elicitation and maintenance of requirements dependencies: A state-of-the practice survey". https://ispma.org/elicitation-and-maintenance-of-requirementsdependencies-a-state-of-the-practice-survey/	Motivation	Published
C1 [49]	Deshpande, Gouri. "Sreyantra: Automated software requirement inter-dependencies elicitation, analysis and learning". 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion). IEEE, 2019.	Motivation	Published
C2 [51]	Deshpande, Gouri, Chahal Arora, and Guenther Ruhe. "Data-driven elicitation and optimization of dependencies between requirements." 2019 IEEE 27th International Requirements Engineering Conference (RE). IEEE, 2019.	Challenge 1	Published
C3 [53]	Deshpande, Gouri, et al. "Requirements dependency extraction by integrating active learning with ontology-based retrieval." 2020 IEEE 28th International Requirements Engineering Conference (RE). IEEE, 2020.	Challenge 1	Published
C4 [58]	Deshpande, Gouri, Behnaz Sheikhi, Saipreetha Chakka, Dylan Valentin Lachou Zotegouon, Navid Masahati,Guenther Ruhe, "Is BERT the New Silver Bullet? - An Empirical Investigation of Requirements Dependency Classification", 8th International Workshop on Artificial Intelligence for Requirements Engineering, Canada, 2021 **Best emerging results and vision paper award**	Challenges 2 & 3	Published
C5 [56]	Deshpande, Gouri, and Guenther Ruhe. "Beyond accuracy: Roi-driven data analytics of empirical data." Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). 2020.	Challenges 3 & 4	Published
J1 [57]	Deshpande, Gouri, Guenther Ruhe, and Chad Saunders. "How Much Data Analytics is Enough? The ROI of Machine Learning Classification and its Application to Requirements Dependency Classification." arXiv preprint arXiv:2109.14097 (2021).	Challenges 3 & 4	Submitted

C6 **[139]** Samer, Ralph, Martin Stettinger, Müslüm Atas, Felfernig Alexander, Ruhe Guenther, and **Deshpande Gouri**. "New approaches to the identification of dependencies between requirements." In 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), pp. 1265-1270. IEEE, 2019.

C7 **[54] Deshpande Gouri** and Rokne Jon. "User feedback from tweets vs app store reviews: an exploratory study of frequency, timing and content." In 2018 5th International Workshop on Artificial Intelligence for Requirements Engineering (AIRE), pp. 15-21. IEEE, 2018.

1.5 Organization Overview

Chapter 2: Our research is exploratory in nature. Thus to understand the open challenges, we analyzed existing literature, conducted a survey with practitioners and evaluated a publicly available data set as an observational lightweight case study using WSL method. Chapter 1 elaborates on these three artifacts of our research work, which act as the basis for challenges and gaps identification.

Chapter 3: In this chapter, Active Learning tool development to address data acquisition challenges is elaborated.

Chapter 4: Based on the various experiments conducted in Chapter 3, we explored and compared AL with an Ontology-based approach and proposed hybrid models utilizing these two methods to address the RDE problem. This work was done in collaboration with the University of Catalunya, Spain. This chapter utilizes content from already published work.

Chapter 5: Exploring further, in the chapter, we evaluated various ways to utilize Transfer Learning through cross-project dependency extraction based on conventional ML methods such as Naive Bayes, Random Forest and Support Vector Machine. A DL-based BERT is also evaluated for Transfer Learning in this chapter.

Chapter 6 and 7: In these chapters, we utilize fine-tuned BERT (a DL-based method) to address the feature extraction challenge for RDE. Based on the results of various experiments, we also explored if BERT is a new silver bullet that could solve the RDE automation challenge to a great extent.

Chapter 8 and 9: In these chapters, we proposed the ML process and a novel ROI model to explore ROI as an additional measure to weigh the ML methods' performance.

Chapter 10 and 11: This chapter provides a vision for a holistic tool for RDE, which has been partially implemented thus far. Finally, a summary of the thesis and future work is consolidated in the last chapter (11)

Please note that few of the references and outcomes from out state-of-the-art survey (Section 2.2) could be repeated in the following chapters in order to keep the context and flow comprehensible.

Part I

Exploration

Chapter 2

Background, Survey and Preliminary Study

2.1 Related Work

Researchers have explored requirement dependencies from traceability perspective and utilized NLP and machine learning to a great extent on requirements artefacts [78] [44]. However, to our knowledge, these studies are limited to extracting the trace and not the structural dependency among the requirements, which is the core of our research. We base our research on the fact that requirement dependency [20] specific research focuses on relationships and their types between a particular type of trace object – namely, explicitly stated requirements. Additionally, our survey with practitioners shows that the dependency information has implications on maintenance and ignoring dependencies has a significant impact on project success [55].

J.N.och Dag et al. [124] analyzed requirements textual content and focused on the dependency identification based on similarity measures, such as Dice, Jaccard, and Cosine coefficients. However, he focused only on the *Similar* dependency type and mentioned linguistic methodology and domain-specific vocabulary use as future work. Chichyan et al. [39] discussed how the semantics of unstructured requirements could be deduced to find the dependency types using Natural Language Processing (NLP) techniques. Ngo-The et al. [123] used fuzzy logic to model the uncertainty concerning the identification of structural dependency constraints between requirements. Weston et al. [164] applied predicate logic to detect conflicting requirements.

Goknil et al. [76], [75] explored requirement dependencies in greater depth from a traceability perspective for change impact analysis. This author used the formal semantics of relation types to infer new relations and determine contradictory relations in the requirements documents. This research utilized semantic web technologies to identify conflicts-with *Refines* and *Contains* relations. Although a tool was developed as part of this work, which provided various modules to identify, maintain and visualize the dependency network as a graph was intriguing; the approach

was theoretical and only applied on a toy data set lacking industrial and empirical study.

In the advent of recent advancements in NLP and ML, RDE automation has garnered serious interest. Recently, Atas et al. [19] used POS-tag (Parts-Of-Speach Tagging) and n-grams for feature extraction in the textual requirements of an industry dataset (annotated by students) before using supervised ML methods for classification. However, Atas et al. highlight the small size of the dataset and the need for domain experts in annotations as serious threats to the validity of the results.

In the recent past, Samer et al. [139] analyzed small industry data sets and utilized Latent Semantic Analysis to extract the dependency types for RDE automation. Although these studies are the first steps towards RDE automation, they lack emphasis on domain-specific knowledge. They need larger training sets and evaluation on various other dependency types for generalization.

2.2 A State of Practice Survey

To understand the challenges faced by the practitioners in dependencies identification and extraction, we conducted a survey. We aimed to understand the perceived importance of requirements dependency identification and state-of-practice extraction. This exploratory survey questionnaire[1] consisted of various close and open-ended questions which were circulated by the ISPMA[2] newsletter as a Google form [55]. To our knowledge, this is one of the first surveys of this nature.

2.2.1 Survey Research Questions (SRQs)

SRQ1: To what extent do practitioners speculate the presence of dependencies in their software systems?

Justification: In 2001, in an industry survey by Carlshamre [36] it was confirmed that up to 80% of the requirements are in some form of dependency with others. We seek to understand what current generation practitioners observe in their software systems through this question.

SRQ2: What do practitioners think about dependencies types and their presence in their software systems?

Justification: In the literature, there have been over 20 different types of dependency types that have been identified [128], [44]. To identify the most predominant ones, we posed this question to the practitioners.

SRQ3: How do practitioners perceive the impact of disregarded (missing) dependencies?

Justification: Although literature explores the consequences of disregarding the dependencies to a large extent, it was essential to gauge the perceived notion of it by the practitioners.

SRQ4: What is practitioners' perceived importance of dependency analysis between multiple requirements (such as transition relationship)?

Justification: Given the variety of dependencies that could exist in any software, from the literature, there is

[1] https://goo.gl/forms/TBie370fZ2kzoD443

[2] International Software Project Management Association (ISPMA). This open, non-profit organization establishes Software Project Management as a discipline in both academia and industry: https://community.ispma.org/

no understanding about the transitive relationship that could exist among a group of requirements that could have a cumulative impact on development, testing or release decisions.

SRQ5: What tools do practitioners use for dependency identification? And what functionality would help it make it better?

Justification: The dependencies change and modify due to various factors such as change requests or new requirements in the course of the software development life cycle. There are several mechanisms to track these; however, it is a laborious and manual task. We were curious to understand practitioners' vision for RDE automation.

2.2.2 Methodology

Scope: In this survey, our focus was to gather information regarding current practices of dependency extraction, maintenance and challenges. Essentially, we hoped to understand practitioners such as managers and developers and testers who are generally involved in release, development and testing activities to provide their insights to set our research agenda.

Instrument: We created a short survey (5 minutes) that had three questions related to role and organization, 13 questions focusing on dependency identification, types and frequency of occurrence, and management. of these 16 questions, 6 were close, and 2 were open-ended, and rest were five-point Likert scale questions. The survey was approved by the University of Calgary Conjoint Faculties Research Ethics Board. A google survey[3] was created and circulated in various areas such as the International Software Project Management newsletter and previous colleagues at work.

Participants: The target population was someone who is involved in release planning, development, and testing, or who is part of the dependency identification or (and) management. In order to gather the representative sample, we excluded the participants who did not serve in any of the said roles. The survey was made available for three weeks. We also excluded incomplete surveys. Since it is difficult to get practitioners to provide their insights (limited availability) to surveys, we considered all the 67 responses for analysis.

2.2.3 Findings

Participants' Information

In this survey, seventy participants with varying roles and designations provided their input. Among participants, 24% were managers, 52% were developers, analysts and testers. Rest were students and others as shown in Figure 2.1. About 43% participants worked on large scale systems, 39% on small scale projects, 13% on pilot projects and rest were grouped under "others" as shown in Figure 2.2.

[3]https://goo.gl/forms/TBie370fZ2kzoD443

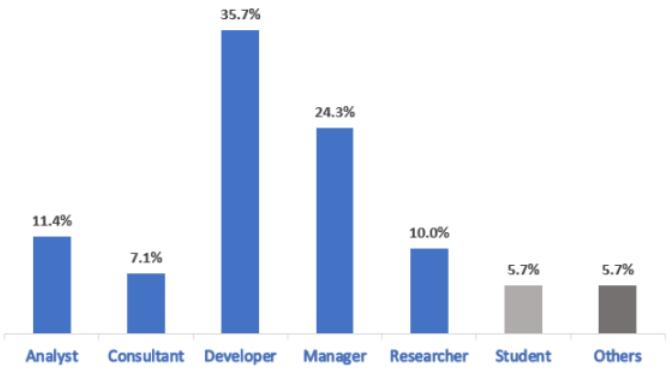


Figure 2.1: Participation summary of this survey: Most of the participants were involved in some part of software development lifecycle.

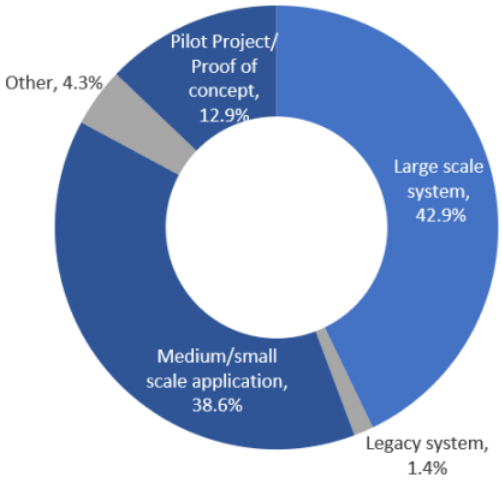


Figure 2.2: Various different kinds of projects our participants work for are as shown here: Most of the participants were working with large and medium scale software applications.

SRQ1: Extant of Requirements Dependencies

Most of the participants affirmed that dependencies are predominant in their software applications. As shown in Figure 2.3 approximately 63% of the participants believed that at least 20% of all requirements would be involved in some form of dependency. More importantly, 22.2% of all participants who accepted 50% or more requirements will have some form of interdependency. This finding strongly approves previous results from an industry survey [36].

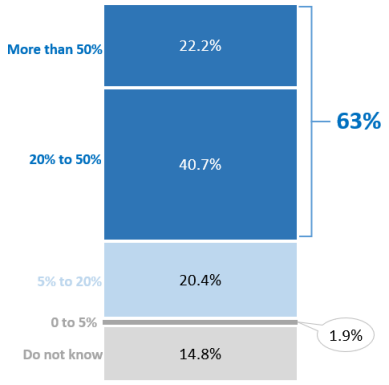


Figure 2.3: Findings of the survey question: How many (percentage of all) requirements do you think are involved in some form of dependency?

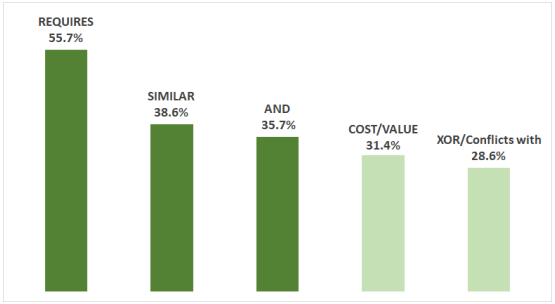


Figure 2.4: Selecting dependency type in the order of their occurrence (most frequently occurring dependency type first)

SRQ2: Requirements Dependencies Types and their Extant

Although there are several types to dependencies that could exist in software product, we chose most frequently appearing dependencies [128] [43] and asked our participants to rank them in the order of their occurrence as perceived by them. Results are as shown in Figure 2.4. The participants were divided in their responses as shown in Figure 2.4.*Requires* was the most sought-after dependency type followed by *Similar* and *And* and others.

SRQ3: Perceived Consequences of Missing Dependencies

From SRQ1 and SRQ2, it is evident that the participants affirm that requirements dependencies exist to a large extent; we asked them various 5 points Likert-based questions to understand their perceived consequences of not considering/missing the dependencies. Results are as shown in Figure 2.5. We presented four hypotheses as part of this. H1) Identifying and measuring the strength of requirement dependencies (strong versus weak dependency) is essential for product planning, H2) Elicitation of dependencies is cognitively challenging and consumes substantial effort. H3) Elicitation, consideration and maintaining requirements interdependencies will increase the chances of software project success, H4) The impact of missing essential dependencies between requirements for upcoming product releases is high. What can be seen in Figure 2.5 is the higher percentage of the participants (over 80%) who agreed or strongly agreed with the four given requirement interdependency hypothesis. In other words, they confirmed the high importance of identifying the strength of dependencies, the difficulty in dependency type extraction, the

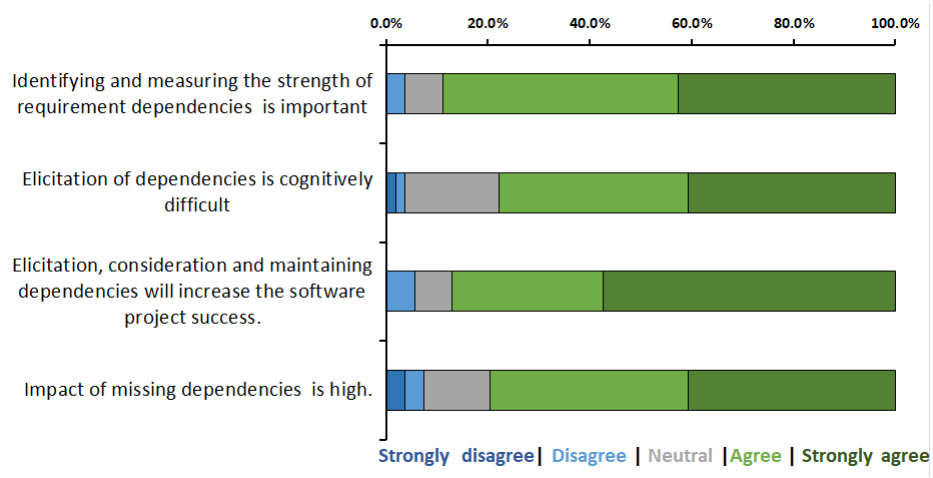


Figure 2.5: Survey responses for the Impact of missing dependencies & extraction of dependencies on a 5-point Likert scale

implications of dependency information maintenance on the project success and the high impact of not considering dependencies.

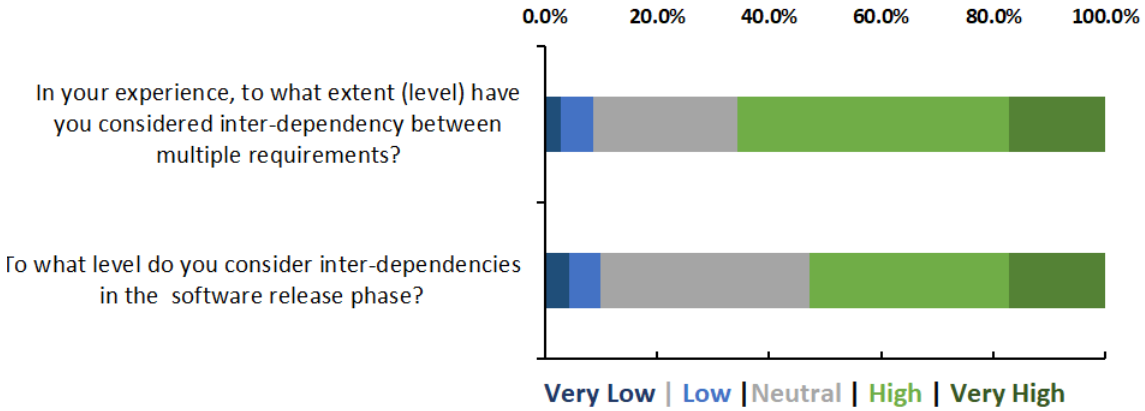


Figure 2.6: Responses for multiple feature analysis to extract dependencies and their importance in release planning

SRQ4: Dependencies Among Multiple Features

Further analysis of the survey results revealed some interesting findings. Figure 2.6 shows that close to 50% of the participants consider dependencies between multiple features (requirements) during release planning. Hence emphasizing the need for the exploration in this direction.

SRQ5: Dependency Extraction Automation

More than 90% of the participants confirmed that they are not using any tool to automate dependency identification or extraction. Also, an open-ended subjective question about the possible functionality of such a tool generated

fascinating responses. At least 50% of the participants demanded automation and interactive interface as an essential feature in the hypothetical automation tool. Few of the comments such as "Automatically detect related requirements and create parent/child relationship between such requirements", and "Keeps an inventory of implemented (and removed) requirements; 2. Integrates with JIRA; 3. Integrates with test management tool; 4. Identifies conflicting and similar requirements" resonates with overall problem definition. These findings provide strong evidence for the high demand for the automation of dependencies extraction in the industry. This is a driving criterion for this research.

2.2.4 Conclusion

In conclusion, this survey helped us reinstate the perceived importance of RDE. Also, findings from various SRQ analyses showed stressing the need for RDE automation and pinpointed various challenges and opportunities in this regard. We use these results to explore further the application of Machine Learning methods to automate RDE. Machine Learning essentially discovers patterns in the underlying dataset using a train set. Such a trained model could then be used to predict for unseen data. Thus, we conducted a preliminary experiment to understand the feasibility of ML in RDE, which is explained in detail in the next section.

2.3 Preliminary study: Weakly Supervised Learning for RDE

Requirements dependencies affect many software development life cycle activities, such as design, implementation, testing, release planning, and change management. They are the basis for various software development decisions. However, RDE is an error-prone, cognitively and computationally complex problem that requires substantial effort since most requirements are documented in natural language. We propose a novel approach to extract requirements dependencies utilizing Natural Language Processing (NLP) and Weakly Supervised Learning (WSL) in two stages. In the first stage, binary dependencies (dependent/independent) are identified, which are further analyzed to detect the type of dependency in the second stage. The evaluation of this approach on the PURE data set - European Rail Traffic Management System - using three machine learners (Random Forest, Support Vector Machine and Naïve Bayes) were compared and tested. Results showed that all the three learners exhibited similar accuracy measures, while SVM needed additional parameter tuning. The machine learners' accuracy was further improved by applying Weakly Supervised Learning to generate pseudo annotations for unlabelled data.

2.3.1 Introduction

In an industry case study, in the telco domain, Carlshamre [35] showed that a large percentage of requirements can have one or more dependent relationships. Also, recently, in automotive systems, Vogelsang et al. [158] found that at least 85% of the analyzed vehicle features depend on each other. These findings emphasize how requirements have to be designed or implemented, how they influence the cost and value of other requirements and how they increase

and decrease the overall implementation efforts when considered in conjunction.

Many studies in the past have recognized the difficulty associated with RDE in software engineering [68], [35], [39], [124], [123], [176]. Besides their extraction, one fundamental problem related to requirement dependencies is that, similar to the requirements themselves, dependencies evolve and change over time. Maintaining and tracking these changes is equally important. If critical dependencies are missed, this would likely result in the rework of the software's design, development, and testing. Furthermore, ignoring dependencies would reduce product releases' value (for the user).

In the past, various techniques including Natural Language Processing (NLP) [38], [124], Fuzzy logic [123], Predicate logic [164] was utilized to extract dependencies. A recent study [78], has applied supervised machine learning methods on requirements specific artefacts. The focus of this study has been mainly on *pair-wise* dependencies from a traceability perspective with no emphasis on the types of dependency and various degrees of dependencies. Since some dependencies might be important while others optional or good to have in the product, dependencies need not be limited to include just two requirements. Additionally, when considered in conjunction, dependencies can also result from value and effort synergies. Hence, it is necessary to focus on these aspects of research and dependencies across multiple requirements.

While dependency extraction is essential, release planning refers to the problem of selecting requirements based on the dependencies. A company has to consider and balance the trade-off between all these factors during release planning; The emphasis is to maximize the customer satisfaction and value synergies and minimize the effort synergies while choosing the most dependent requirements for any given release. Although there have been studies to extract value and cost-based dependencies [76], [75], there are no studies that consider them as synergies while selecting the requirements. If requirement dependencies are not tracked and maintained in the life cycle of the software development then, it would not be possible to facilitate dependency-centred release plans, which, as a result, would eventually maximize the value of the release and reduce the penalty when the dependencies are missed.

In this paper, as a first step towards broadening the scope of dependency analysis and maintenance, we propose (i) a systematic approach for extraction of mutual requirements dependencies and its initial evaluation utilizing Weakly Supervised Learning and (ii) extend this approach to cover more general types of dependencies. (iii) model dependency management as a multi-objective optimization problem, balancing value and effort synergies, and penalties (if any) from violating dependency constraints for release decisions.

The remainder of the content is structured as follows: Concepts and research questions are explained in Sections 2.3.2 and 2.3.3 respectively. Section 2.3.4 explains the results. Section 2.3.7 provides details on the threats to validity. Finally, Section 2.3.8 presents discussion.

2.3.2 Concepts and Research Questions

Maalej et al. [104] projected a paradigm shift in requirements engineering and software evolution towards data-driven processes. Our research follows this shift to study data-driven techniques to extract high-level (binary) and fine

granular dependencies (types of dependencies) between requirements. We also outline the use of the dependencies to optimize the release decisions. This section provides details on terminologies and concepts used in this paper.

High Level Dependencies (Binary: dependent or independent)

For a set of requirements R, and any pair of requirements $(r, s) \epsilon R$, the symmetric relationship is called a *dependent*, if there is at least one type of dependency ($Requires$, $Similar$, Or, And, XOR, value synergy, effort synergy) between r and s (independent of type, direction, and strength).

Fine Granular Dependencies

Assuming, we can extract high level dependencies, the second research question is to extract the type of the dependency. Details of these types are as following.

Definition: For a set of requirements R and any pair of requirements $(r, s) \epsilon R$, if r requires s, or s requires r, then, r and s are in a relationship called $Requires$.

Definition: For a set of requirements R and any pair of requirements $(r, s) \epsilon R$, if r and s are required in conjunction, then, r and s are in a relationship called And.

Definition: For a set of requirements R and any pair of requirements $(r, s) \epsilon R$, if r and s are semantically similar, then, r and s are in a relationship called $Similar$.

Definition: Violation of structural dependencies (like $Requires$, And etc.) is supposed to create a *penalty*. The degree of penalty depends on the impact of the violation to the user. We define the *penalty()* function on a nine-points scale (0-9 : low to high).

Weakly Supervised Learning

Weakly Supervised Learning (WSL) [178] combines the benefits of Supervised Learning and Unsupervised Learning. It is motivated by the high cost of data labeling. There are various strategies for using unlabeled data to improve the performance of standard Supervised Learning algorithms, especially in the situation where a small amount of labelled data is available, which is insufficient to train a good learner, while abundant unlabeled data are available. WSL is an umbrella term covering a variety of techniques, which attempt to construct predictive models by learning with weak supervision. This approach is a form of *conservative co-testing* strategy [118] where, for each iteration, an unlabeled example is labeled if the majority classifiers agree on the labeling [152].

As shown in Figure 2.7, firstly, machine learning models (NB: Naive Bayes and RF: Random Forest) are trained on the *original data set*. Further, these classifiers are used to classify the unlabeled data samples. The strategy is to extract just the data samples on which all the classifiers achieved consistent results. If all classifiers have predicted the same class label for a given data point (sample), this label is assigned as a pseudo annotation.

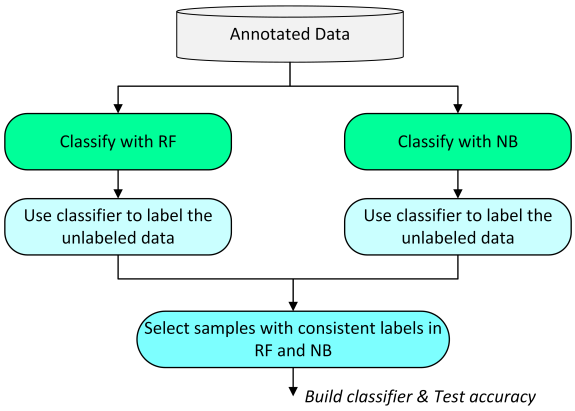


Figure 2.7: Main steps of our WSL based approach utilizing NB: Naive Bayes, RF: Random Forest for pseudo label selection

Research Questions

The approach is to utilize NLP, supervised and WSL algorithms to automate the extraction of fine granular requirements dependencies. The proposed research is organized around the three research questions (RQs).

RQ1.1 How accurate are supervised machine learning methods RF, SVM, and NB and WSL at extracting pairs of high-level dependencies?

RQ1.2 How accurate are supervised and WSL extracting fine granular requirements dependencies?

RQ1.3 How effective and how efficient is the usage of interactive swarm intelligence for determining the functionality of the upcoming software release?

Each RQ's output is input to the next RQ. Hence, RQ1.3 utilizes fine-grained dependencies information from RQ1.2 to perform release optimization, which has objectives specific to dependencies between requirements. We plan to utilize *swarm intelligence*, a collection of *bio-inspired* optimization algorithms, for this RQ. These algorithms have been proven successful in a large number of application engineering, image processing, and data mining [169] providing an option for performing optimization in a *interactive mode* with the release decision-maker. The logical structure of our approach is shown in Figure 2.8.

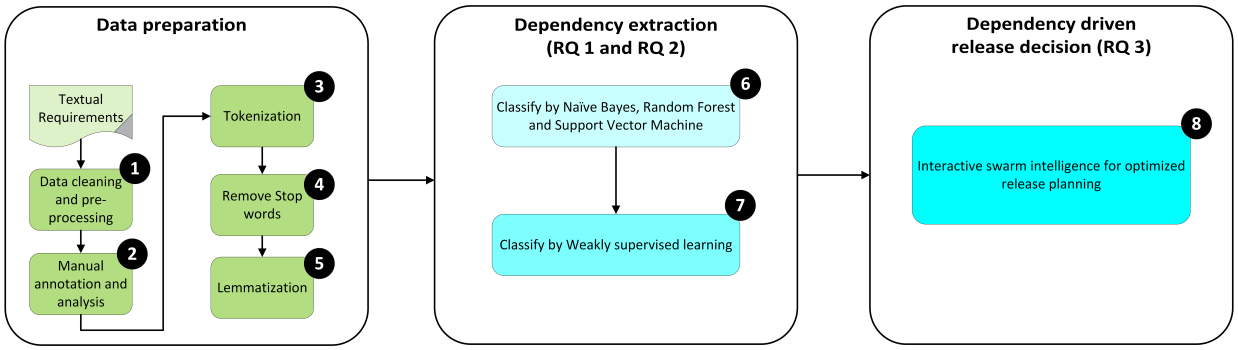


Figure 2.8: Structure of our research approach spread over three stages and 8 steps

2.3.3 Methodology

This section describes the research method, data preparation, classifiers and evaluation phases. The following describes the essential steps of the method from Figure 2.8 and how it was applied to our data.

Data Preparation

Firstly (Step ❶), raw data from the textual requirements information (or document) is processed to extract the requirement statements. Following this extraction, the manual annotation process (step ❷) must be carried out. To proceed with text classification (generating classifiers), the data set is passed through a NLP pipeline. Thereby, the data is first tokenized, eliminating possible stop words (English dictionary based), and then lemmatized using standard snowball Stemmer and WordNet Lemmatizer [140] (step ❸, ❹, ❺).

Classifiers

To solve RQ1.1 and RQ1.2, we utilized three classifiers: Random Forest (RF), Naive Bayes (NB) and Support Vector Machine (SVM) (step ❻). Multi-class annotations were also utilized to further develop multi-class classifiers (step ❼). While the NB algorithm searches for the best linear separator according to some criterion, SVM and RF have been used successfully and prominently for text classification [140]. We utilized Python's scikit-learn library [9] for implementation.

While RF and NB classifiers performed well with basic settings, SVM needed additional tuning on the hyper-parameters. Considering the adaptive capacity of SVM, Radial Basis Function (RBF) was selected as the kernel function, penalty parameter C=2.0 and kernel parameter gamma =2.0 to achieve better classification accuracy [86].

Evaluation

A good validation technique should not overestimate or underestimate the model performance on unlabelled data. Since the annotated data set was small (a balanced 300 data samples, where each data sample represents a pair of requirements), we utilized a more robust sampling technique called k times k-fold cross-validation (CV) technique to eliminate possible bias in the classifiers. This validation technique has been proven as the most unbiased validation technique, which can be effectively utilized in the event of smaller training data sets to avoid overfitting [153], [110], [86]. We utilized 10 times 10-fold CV technique for this study.

2.3.4 High level Dependency Extraction - RQ1.1

Data preparation

For this study, the sample data for evaluation were taken from the public data set: PURE [70]. The Ertms/ETCS [4] functional requirements specification document (SRS) consisted of 199 requirements with average sentence length of 10 words. With these $n = 199$ requirements, we ended up with $n(n-1)/2 = 19,701$ potential pairwise dependencies.

In order to generate the ground truth data, we manually annotated the 550 data samples since this data belonged to a public PURE library [70]. Two of the authors engaged in this activity independently first studied the project utilizing the description in the SRS document. For more effective manual annotation, the pair-wise cosine similarity was utilized as a starting point for the manual annotation. The final data set consisted of the samples on which both the annotators agreed upon the label. The term *original data set* is used to describe this balanced data set of 300 samples.

To ready the data for classification, it was first tokenized, then all the English stop words were removed, and each token was then lemmatized to perform classification. Further, the pairs with higher cosine similarity were picked first to check possible dependencies. Finally, a combination of higher and lower cosine similarity pairs was randomly chosen to generate 550 annotations. Out of them, 367 were annotated as independent, 150 as a dependent, 33 could not be classified. For multi-class annotations, 38 were of *Requires* type, 89 *Similar*, 19 *And*, 3 *Or* , and 1 *XOR*. At this stage, data was now ready for classification.

Stage 1: High level Dependency Extraction

For the Binary (high-level dependency) classifier, the initial data set (after annotation) had a class imbalance. We extracted all 150 data samples (with class = 1, dependent) and randomly extracted 150 independent data samples from the pool of 367 independent samples. Following the methodology explained in Section 2.3.3, we generated classifiers with NB, RF and SVM.

Averaged (from 10 times 10-fold CV) results for the classifiers are shown in Table 2.1 (a). Of the three, SVM has the highest precision and F1 score. It implies that SVM could classify the valid dependent pair of requirements most accurately of all the classifiers. However, all three classifiers have lower recall and higher precision rates, which means they cannot correctly identify the possible dependent pairs. Additionally, all three classifiers generated approximately 0.7 F1 accuracies. However, to improve accuracy, we analyzed the effect of our proposed WSL mechanism. We concluded that all present classifiers could be utilized as an efficient prediction model.

Table 2.1: RQ1.1: Average classifier accuracy from 10 times 10-fold CV before (a) and after (b) utilizing pseudo-labelled data using WSL (see Figure 2.7). Class 0: Independent, 1: Dependent

	Classifier	**Precision**	**Recall**	**F1 Score**
a) Original data set	RF	0.79	0.60	0.67
(#Class 0 = 150)	NB	0.77	0.70	0.73
(#Class 1 = 150)	SVM	**0.85**	0.68	0.74
b) Original data set + WSL sample	RF	0.90	0.81	0.85
(#Class 0 = 300)	NB	0.89	0.85	0.87
(#Class 1 = 300)	SVM	**0.94**	0.85	0.89

2.3.5 Fine granular Dependency Extraction - RQ1.2

Pseudo Labeling using Weakly Supervised Learning

In order to (pseudo) label the rest of the 19,000 requirement pairs, RF and NB classifiers created from the original data set were utilized (Figure 2.7). Since running classifier models have a non-deterministic characteristic, multiple instances for each of them were created. Only the samples (11,141) with consistent classification were finally selected. This new annotation resulted in *pseudo labelled samples*. The updated *original data set*, called as *updated data set*, was then tested using 10 times 10-fold CV through classification.

Results for various data sample sizes are shown in the Table 2.1 (b). We concluded that WSL classifiers performed better than those original learners from the results. The average CV showed more than 10%+ better performance utilizing 300 pseudo labelled data samples. Additionally, all three classifiers demonstrated notable improvement while SVM performed well overall.

Stage 2: Extraction of the Dependency Type

To address RQ1.2, we utilized 38 data samples for the three classes (114 in total). To demonstrate our findings, we annotated a multi-class subset of *original data set* is referred to as *baseline multi-class data set* in this paper. Utilizing this data set, baseline accuracies for RF, NB and SVM classifiers were generated. Results are shown in Table 2.2 (a).

We chose to report the weighted average results for the multi-class classifiers because when binary classification metric is extended to multi-class problems, the data is treated as a collection of binary problems, one for each class. There are then several ways to average binary metric calculations across the set of classes, each of which may be useful in some scenarios. Where available, it is recommended to select the weighted average parameter [8].

Analysis of the statistics revealed that the accuracy of all the classifiers remained in the same range (approx 0.6). However, SVM and RF performed well whereas NB demonstrated poor recall and a 10-fold CV score. The poor performance of the classifiers could be attributed to a limited and small data set used for classification.

The results of utilizing WSL on the *baseline Multi-class data set* are documented in Table 2.2 (b). The balanced data set for three classes was extracted randomly for classification from a pool of 4,460 data samples from WSL steps (Figure 3) on 19,000 data samples. This three class balanced data set consisted of 228 samples (=76*3). Statistical results show that NB and RF performed comparatively well, but the performance of SVM was exceptional. Although this improvement in accuracy is welcoming and supportive of WSL, closer evaluation is required to affirm the improvement in the performance.

2.3.6 Dependency aware release decisions - RQ1.3

Release planning is a wicked problem [132], i.e. the formulation of the problem is cognitively difficult, and there is no easy method to decide on a single solution. We approach the wickedness of the release problem by applying

Table 2.2: RQ1.2: Average classifier accuracy from 10 times 10-fold CV before (a) and after (b) utilizing pseudo-labelled data using WSL. Class 1: *Requires*, 2: *Similar*, 3: Others

	Classifier	**Precision**	**Recall**	**F1 Score**
a) Original data	RF	0.67	0.65	0.61
(#Classes 1,2 & 3 = 38)	NB	0.65	0.60	0.59
	SVM	0.76	0.69	0.69
b) Original & pseudo-labeled data	RF	0.85	0.81	0.81
(#Classes 1,2 & 3 = 76)	NB	0.83	0.81	0.81
	SVM	0.88	0.87	0.87

an evolutionary modelling and interactive problem-solving mechanism. This includes interaction with the user and decision-maker. Interactive optimization approaches acknowledge existing limitations of modeling and parameter settings. Also, they value the user's expertise in the application domain [108].

Swarm Intelligence is an emerging *bio-inspired* paradigm for optimization algorithms. As a form of collective intelligence, they have been proven successful in a vast number of applications in engineering, machine learning, image processing, and data mining [169]. Using randomization, swarm intelligence is also able for highly complex problems to overcome being trapped in local optima.

Overcoming the simplistic notion of maximizing a value function defined from isolated requirements, the new problem formulation examines requirements dependencies and makes them the drivers of decisions. The novelty of this optimization approach is to bundle highly dependent requirements and make them the core content of releases. Therefore, the optimization is multi-objective with three optimizing functions:

$$Minimize\ F1\ = \sum_{All\ structural\ dep's\ n} penalty(n)$$

$$Maximize\ F2 = \sum_{All\ value\ synergies\ n} values(n)$$

$$Minimize\ F3 = \sum_{All\ efforts\ synergies\ n} effort(n)$$

F1 is defined on the product set $R \times R$, which results in a quadratic function. The other two objectives, F2 and F3, are defined on the power set of R. These functions are complicated to formulate, and applying swarm intelligence is expected to handle both non-linearity and non-convexity of the set-defined synergy functions: *value* and *effort* respectively. The functions describe the added value and the reduced effort compared with an isolated selection of

requirements.

2.3.7 Threats to Validity

The results are preliminary as we studied the approach just for one data set. We will perform a more comprehensive analysis with data from PURE [70] in the future. In addition, we have attracted industrial data with access to domain experts to further check the validity of the approach.

For RQ1.1, the size of 500 samples is relatively small compared to the size of the overall sample set. We proposed WSL to overcome this deficit. For the actual annotations, each sample was annotated twice. In case of inconsistency, samples were not considered. Annotating test data from applying WSL is considered one way to overcome the annotation bottleneck [178]. We only considered samples with consistent classification from all the classification runs.

For RQ1.2, the annotated data is small, and the results might be biased. Also, this might have consequence on the WSL created output. We plan to evaluate our approach with additional annotations specific to a few selected classes.

k-fold CV is one way to validate results. There is a bias-variance trade-off associated with the choice of k in k-fold cross-validation. Given these considerations, one performs k-fold cross-validation with k=5 or k=10, as these values have been shown empirically to produce test error rate estimate [88]. We applied k=10. In addition, we applied it multiple times to overcome the impact of the fold selection. However, we need to test our approach with domain experts to see how valid the results are.

2.3.8 Discussion

In this study, we utilized a WSL based labeling approach, which exploits unlabeled data through the WSL method. This preliminary study showed that the quality of the textual content describing requirements is one of the key performance factors of our approach. Also, the general use of this approach in different domains needs thorough evaluation, which is part of the future research plan. In hindsight, while evaluating ML application for RDE automation, in this experiment, we encountered various difficulties, such as 1) Challenges due to natural language in the requirements, 2) Challenges with knowledge acquisition (labeled data), and 3) Lack of labeled data for ML training. Although there is a broad spectrum of advanced and conventional ML techniques to choose from for any automation, the nature of the problem and data-related challenges could influence the choice of ML technique largely. Additionally, the need for more extensive training data and computational power of the advanced ML technique incites another dimension to the ML technique selection criteria, which can not be ignored. In the next section, we elaborate more on these challenges and discuss various approaches to tackle them.

Summary

This chapter mainly explored three components listed to identify the open challenges. We also proposed solution approaches to tackle each one of them.

1. **Literature review:** We explored various existing literature to understand the status of advancements for RDE. As a result, it was found that RDE automation through ML exploration is in the nascent stage.
2. **Survey:** We conducted a state-of-the-art survey with the practitioners to understand the current challenges and open problems.
3. **Preliminary study:** To verify if the RDE automation is feasible and technical challenges in such an implementation, we conducted a preliminary study: WSL for RDE using publicly available dataset [70]. Results were encouraging; however, the ability of the conventional ML to generalize the information in the underlying dataset and the quality of the annotations itself demanded further scrutiny.

These challenges, when addressed, could provide a direction for effective RDE. In the next part, we discuss the approaches we took to address these challenges.

Part II

Addressing Data Challenge (RQ1 & RQ2)

Chapter 3

Active Learning for RDE

3.1 Introduction

We evaluated AL and compared it with Passive Learning (PL), also called random sampling, in various stages and multiple projects. Active Learning is a form of ML in which a learning algorithm interactively queries an *oracle* (typically a human expert) to obtain the desired label for new data points [141]. We performed 54 tests for three sampling techniques, namely MinMargin, LeastConfidence and Entropy, for the three selected ML algorithms: Random Forest, Naive Bayes, and Support Vector Machine for multi-class classification. In this study, we explain the results from NB and RF for the six Mozilla family projects in detail.

3.2 Related Work

Active Learning has been extensively used in various Software Engineering specific problems. For example, Du et al. [63] developed an Active Learning-based approach for trace link recovery between the artefacts of the same project and showed that it performed better than the conventional ML methods such as RF, NB and SVM. Ekrem et al. [91] used the Active Learning heuristic to classify software effort estimate data and showed that their method is effective compared to complex estimation methods in use. Xu et al. [168] and Lu et al. [101] successfully predicted defects for the upcoming version of the software using defects from previous versions using Active Learning-based prediction models.

Although Active Learning has been explored to classify clone anomaly reports [100], test reports [160] and user models [117], it has not been explored in dependency extraction so far. In the recent past, in Requirements Engineering (RE), Dhinakaran et al. [60] showed that AL could reduce the supervision effort without compromising classification accuracy. Similarly, C. Arora et al. [17] showed that the AL approach could improve the accuracy of automated domain model extraction.

Additionally, Olsson et al. [125] surveyed all the studies of NLP applications that used Active Learning and concluded that it is effective in addressing challenges in the NLP tasks such as information extraction, named entity recognition, text categorization, part-of-speech tagging, parsing, and word sense disambiguation. Thus we hypothesize that Active Learning could benefit RDE problem solving, which has natural language as its core part, thus making it a form NLP task.

3.3 Research Method

This section presents the problem statement and overview of the experimental setup for this empirical evaluation.

3.3.1 Research Question

We aim to evaluate AL for effective knowledge accumulation and improved classifier performance for RDE. Hence the research question we have formulated is as follows:

RQ1.4: Does Active Learning perform better than Passive Learning in terms of accuracy for RDE?

Justification: Supervised Learning mechanism heavily relies on manually annotated data. This is not only time consuming but also expensive in terms of effort. Additionally, for RE, annotation needs domain experts' involvement to a greater extent. Any mechanism that could bring in a reduction will be an added value. Active Learning is widely used in the areas where the domain expert's time is scarce. Through various empirical evaluations, we will explore how AL could address this inherent challenge of knowledge acquisition for RDE automation.

3.3.2 Experimental Setup

Random sampling (Passive Learning) randomly selects a training set. Active Learning selects the most informative instances using various sampling techniques such as MinMargin, LeastConfidence and Entropy [141]. We compare Passive Learning (PL) with AL using a chosen ML as a classifier for this scenario. The analysis was done by concurrently adding 20 training samples in every iteration to classify the unlabeled instances.

Figure 3.1 depicts the workflow and various steps of the AL-based method, which chooses annotations intelligently in every iteration.

- **Initialization:** All the textual requirements information is passed through an NLP pre-processing pipeline first for all the dependency candidates D (Step ❶). The data was first processed to eliminate stop words and then lemmatized following the traditional NLP pipeline [16]. We used the Term Frequency times Inverse Document Frequency (TF-IDF) [129] feature extraction before training ML methods.

  A domain expert - also referred to as $Oracle_1$ - randomly chooses a (typically small, in our test, we chose 20 samples) set of potentially dependent requirement pairs and annotates them (Step ❷) to generate a training

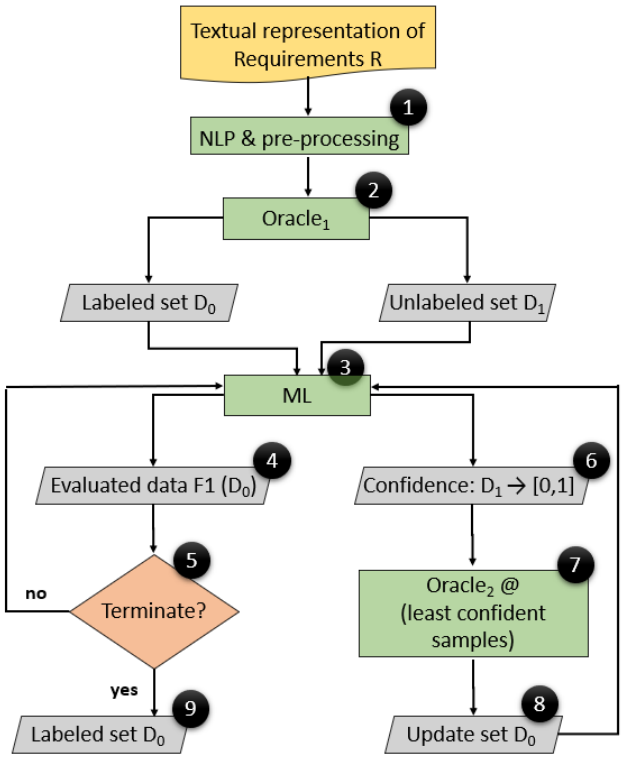


Figure 3.1: Step-wise logical flow of AL method for RDE

set.

- **Training**: Thus, the prepared training set is used for training the machine learner (Step ❸). Our method uses one of the three classifiers: Naive Bayes, Random Forest and Support Vector Machine as they are simple classifiers to work with textual data [121] [82]. Such trained ML (Step ❹) are measured based on F1-score. These models provide a probability of a test instance belonging to a class that is indicative of the uncertainty (e.g., for a binary classification problem, the closer the probability of a prediction is to 0.5, the higher the uncertainty of classifier's prediction) [60].

- **Choose least confident samples for manual labeling:** The ML learner is trained with an initially small amount of labelled samples which is then used to classify the unlabelled data applying *pool-based sampling* [141] technique. This is well-suited for sampling the most uncertain data point or instance(s). All requirement pairs in the unlabelled data pool are evaluated according to their informativeness. Informativeness is computed using one of the three *uncertainty sampling: MinMargin, Entropy, Least Confidence* [141] techniques, which selects the instances with the lowest confidence level (probability value) to be labelled next (Step ❻). In every iteration, n most uncertain predictions are shown to the $Oracle_2$ and labels are acquired (Step ❼). Such requirement pair with its label is then added to the training set (Step ❽).

  The selection of $confidence$ threshold and n is a trade-off between the accuracy of the classifier and the efficiency of the trainer. For example, for *least confidence* technique, in every iteration, Uncertainty sampling

recommends instances in which the model is least confident about to be explicitly labeled by $Oracle_2$. Such intelligently chosen instances labelled by $Oracle_2$ are added to the training set, and the process repeats until one of the termination criteria is met.

- **Termination criteria:** AL stops if either a predefined number of iterations is done, or if the degree of improvement is minimal, or if no more unlabelled samples are available (Step ❺, ❾).

We implemented this AL method as a tool in Python utilizing Scikit-learn libraries [127]. To enable comparison between AL and PL, in every iteration, when AL chose the most uncertain requirements pairs for annotation using oracle, PL chose these pairs randomly. Due to imbalanced class data, special care was taken to balance the dataset using the undersampling technique [150] for this multi-class classification.

3.4 Data

Online bug tracking system such as Bugzilla [2] is widely used in open source software (OSS) development. Feature requests and new requirements are logged into these systems in the form issue reports [145] [26] which help software developers to track them for effective implementation [146], testing and release planning [136]. Also, data from Bugzilla has been explored for bug report summary utilizing dependency related metadata from bug reports [103] [90] [130]. Kim et.al. [90] utilized *Blocks* and *Depends_on* association relationships to examine a method for summarizing the bug reports based on weighted-PageRank using these dependency relationships.

3.4.1 Data Collection

Collecting data from Bugzilla was a substantial effort carried out in multiple rounds. We extracted 16,239 requirements and related information for 15 Mozilla families of products from Bugzilla. All the information for requirements was collected using Bugzilla REST API [3] through a python script such that each one of the enhancements considered for retrieval is dependent on at least another one in the dataset. The data spanned from 08/05/2001 to 09/08/2019.

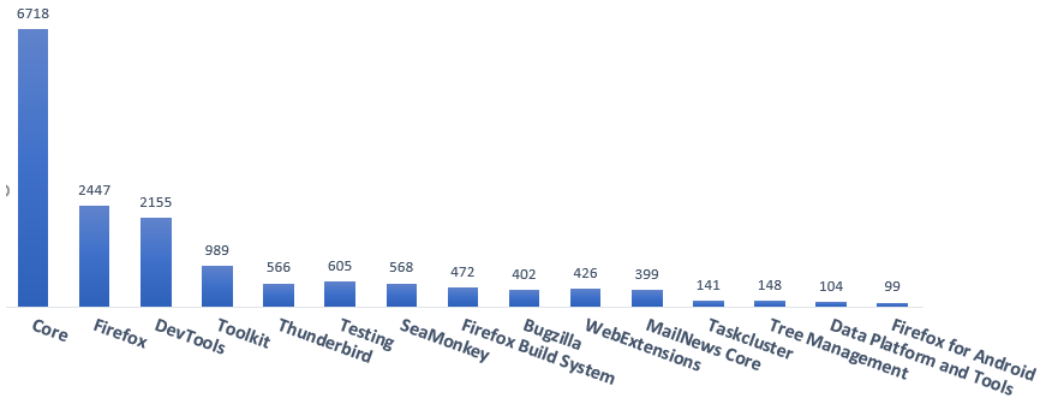


Figure 3.2: #Requirements mined from Bugzilla for the top 10 Mozilla projects

Figure 3.2 shows the list of projects and their respective count of requirements used in this section. The Core project is a kernel of other Mozilla software. Firefox is the Mozilla Foundation's web browser; Toolkit is a set of APIs. SeaMonkey is an all-in-one internet application suite, including web browser, email and newsgroup client, and HTML composer. Thunderbird is an Email client Application, Devtools is the developer tool within the Firefox web browser. Testing is the automated testing of Mozilla client code (Firefox, Thunderbird, Fennec, Gecko, etc), WebExtension is add-on API, Firefox for Android is a mobile version of Firefox for Android devices [3]. Figure 3.2 shows the number of requirements that were mined for this study. We have considered the top 6 projects (with more extensive data availability) to perform this empirical evaluation.

In Bugzilla, feature requests are a specific type of issue that is typically tagged as "enhancement" [116]. We retrieved these feature requests for the Mozilla family of products using the search engine in the Bugzilla issue tracking system. We exported all the related fields such as Title, Type, Priority, Product, Depends_on, See_also, and Blocks. Figure 3.3 depicts one such *enhancement* from Bugzilla. As shown in Figure 3.3, each issue report contains dependency relationships with other issue reports as references metadata [90].

3.4.2 Data Preparation

Further processing of this raw data resulted in 49,492 dependent pairs of various dependency types. We also generated 1,474,772 requirements pairs from these requirements, which had no dependency between them as a harmful sample data set of the class *independent*. We define the two relationships *Depends_on* and *Blocks*, which are inverse of each other as a *requires* structural dependency. For example: in Figure 3.3, the *Depends_on* implies that the bugs listed in this field (issue 1464506, 1464509, 1480430) must be resolved before this issue (1420347) [116]. So, issue 1420347 *requires* issues 1464506, 1464509 and 1480430. *Blocks* implies that an issue (1420347) must be resolved before the issues listed in this field (1527023) can be resolved [116]. So, issue 1527023 *requires* 1420347. Please note that the words *requirements* and *feature requests* are used interchangeably in this study. Table 3.1 shows statistics of project-wise dependency types. Core being a biggest project showed highest number of dependent pairs.

Similarly, for the relationship *See_also*, which implied *related to* dependency [3], was interpreted as *other* dependency type for this study.

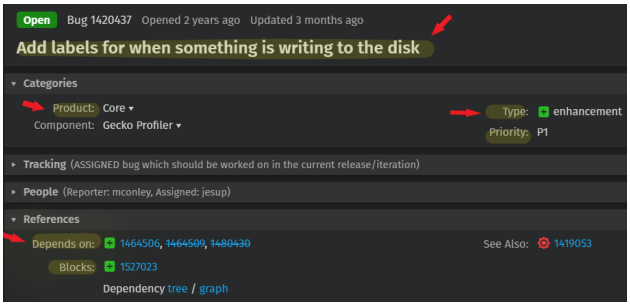


Figure 3.3: Bugzilla's *feature request (also referred to as requirement)* example. Red arrows indicate the information of interest in this study

Table 3.1: Statistics for project-wise dependency types

Project	**Requires**	**Other**	**Independent**
Core	13,645	1,535	218,926
DevTools	4,215	493	95,404
Firefox	5,429	690	167,884
Firefox Build System	1,080	204	104,334
Testing	1,155	145	62,528
Toolkit	2,054	279	141,362

3.5 Results: Comparison of AL with PL for RDE - RQ1.4

This section presents the results from empirical analysis of the six selected projects from the Mozilla family: Core, DevTools, Firefox, FirefoxBuildSystems and Testing. We compared AL with a conventional random sampling-based classification- *Passive Learning* - using the hyperparameter tuned RF ML algorithm since it showed better results compared to NB and SVM.

Beginning with 60 training samples of each class (*Requires Independent and Other)*, we developed multi-class classifiers for both AL and Passive Learning for this empirical study scenario. While AL employed MinMargin sampling technique[1] to identify 20[2] most uncertain instance (requirement pair) for oracle to label. Passive Learning randomly selected 20 instances and added them to the training set along with their label, thus, kept the two approaches comparable in all the 20 iterations. Since data is already labeled, for AL, we pretend they are unlabeled until queried and labeled by a simulated oracle in this scenario. As shown in Figure 3.4, for AL, Core, DeveTools, and Firefox projects achieved F1>0.7, whereas FireFoxBuild, Testing and Toolkit projects achieved F1<0.6. This disparity could be associated with data availability constraints. *Other* class samples were scarce for all the projects hence the results for this class did not yield good results. However, we are interested in *Requires* dependency type extraction mainly, thus as captured in the results, the AL's F1-score for this particular class (**Requires**) outperformed Passive Learning by a small margin for most of the projects except Testing.

Further looking closer into precision and recall of *Requires* class for the Core, DevTools, and Firefox provided nuanced insights. As shown in Figure 3.5, for all these projects, AL's precision and recall exceeded the Passive Learning by a good margin.

Results from AL for Core, DeveTools and Firefox projects achieved F1>0.7, whereas FireFoxBuild, Testing and Toolkit projects achieved F1<0.6. Also, AL's F1-score for *Requires* class outperformed Passive Learning (PL) by a small margin for most of the projects except for Testing. For Testing project PL excelled over AL However, overall F1-score for it was low. Small train set could be the reason for this performance.

[1]We also conducted tests using Least Confidence and Entropy, however, due to space constraint, only Least Confidence output are discussed for this study

[2]The tests were performed with#samples = 10, 15 and 20, only. We will discuss results related to #samples=20

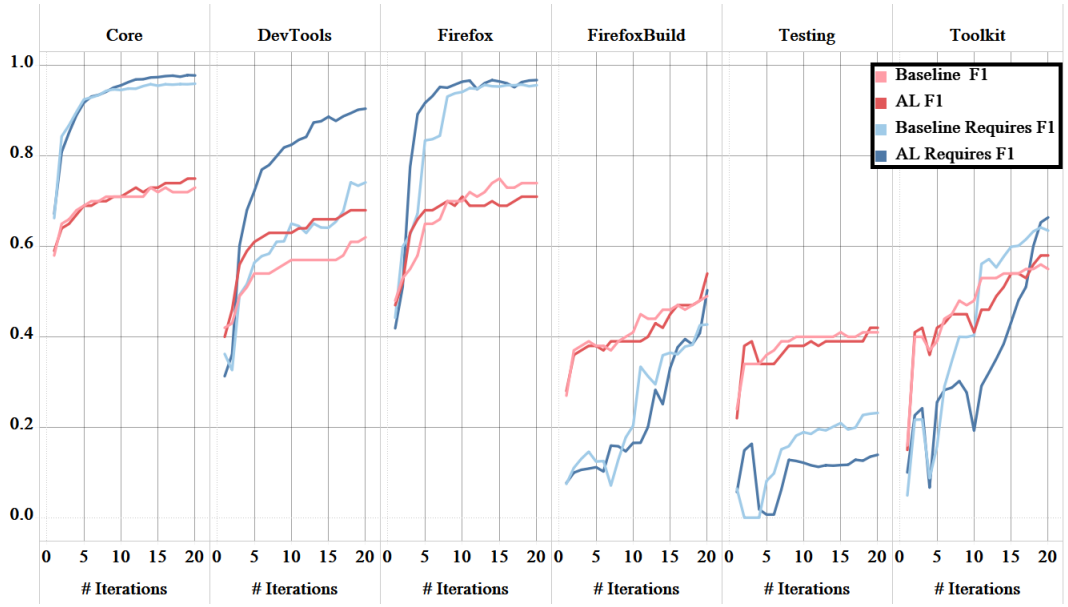


Figure 3.4: AL vs PL for Least Confidence sampling technique for all the selected projects for Multi class classification

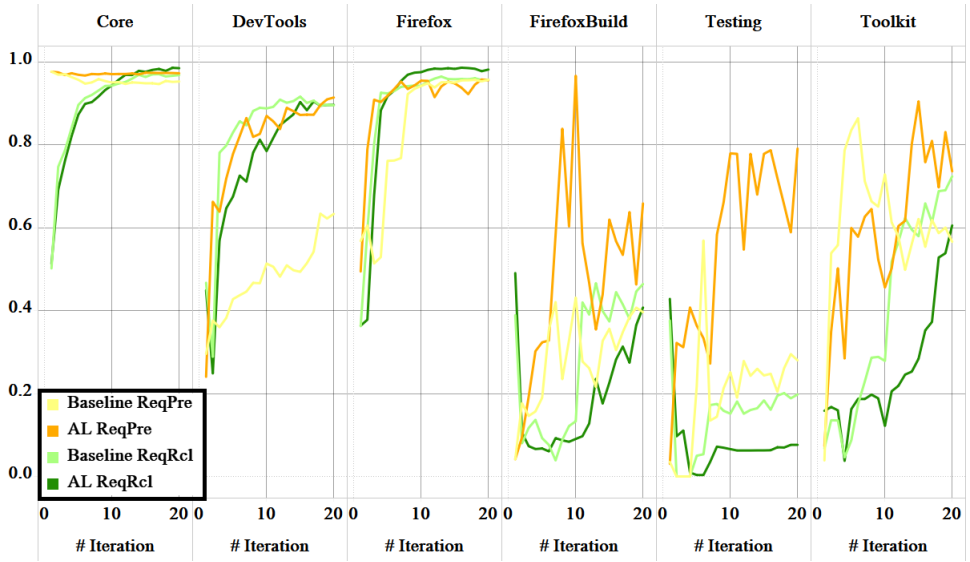


Figure 3.5: Precision and Recall of *Requires* class for AL vs PL using Least Confidence sampling technique for all the selected projects for multi-class classification. Baseline: Passive Learning, ReqPre: Precision of *Requires* class, ReqRcl: Recall for *Requires* class

3.6 Discussion

Most of the experiments with other sampling techniques with varying query sizes yielded similar results with negligible variations. Perhaps trying oversampling techniques or advanced feature extraction techniques could have altered the results for smaller projects such as FireFoxBuild, Testing and Toolkit. However, lack of data was a significant constraint to our knowledge since larger projects such as Core, Firefox, and DevTools showed encouraging results.

In hindsight, as highlighted by Burr Settles [141] in the AL literature survey, while using AL, it is assumed that often there is only one oracle or that the oracle is always correct or the cost of the labeling instances is free or relatively inexpensive. Thus, it ignores the expense part of utilizing AL. Consequently, AL which ignores cost, might perform no better than PL. Settles cautions that the pattern or feature set of the underlying natural language (of the dataset) should be known in advance to use AL safely. Also, the stopping criteria of AL should be mainly driven by the cost of acquiring instances rather than just improving the accuracy. As a result, we explored Ontologies to capture the details in the dataset, which could be further used to gather meaningful data. In the next section, we compare AL with Ontology-based Retrieval (OBR) and propose a hybrid solution for RDE. This solution is evaluated on two industry datasets.

Chapter 4

Hybrid Model for RDE

4.1 Requirements Dependency Classification by Integrating AL with Ontology-based Retrieval

Incomplete or incorrect detection of requirement dependencies has proven to result in reduced release quality and substantial rework. Additionally, the extraction of dependencies is challenging since requirements are mostly documented in natural language, which makes it a cognitively difficult task. Moreover, with ever-changing and new requirements, a manual analysis process must be repeated, which imposes extra hardship even for domain experts.

The three main objectives of this research are: 1) Proposing a new dependency extraction method using a variant of Active Learning (AL). 2) Evaluating this AL and Ontology-based Retrieval (OBR) as baseline methods for dependency extraction on the two industrial data sets. 3) Analyzing the value gained from integrating these diverse approaches to form two hybrid methods.

Building on the general AL, ensemble and self-training based machine learning, a variant of AL was developed, which was further integrated with OBR to form two hybrid methods (Hybrid1, Hybrid2) for extracting three types of dependencies (requires, refines, other): Hybrid1 used OBR as a substitute for human expert; Hybrid2 used dependencies extracted through the OBR as an additional input for training set in AL.

For two industrial case studies, AL extracted more dependencies than OBR. Hybrid1 showed improvement for both data sets. For one of them, $F1$ score increased to 82.6% compared to the AL baseline score of 49.9%. Hybrid2 increased the accuracy by 25% to the level of 75.8% compared to the AL baseline accuracy. OBR also complemented the AL approach by reducing 50% of the human effort.

4.2 Introduction

In this research, we compare the two baseline methods that take a radically different approach to the requirements dependency extraction problem. The first method utilizes a domain ontology to extract the dependency pairs and their dependency type. The second method uses Ensemble-based Active and self-training labeling learning [102]. This approach selects a good set of requirement pairs to form a labeled training data set - an active learning approach. Also, it utilizes both labeled and unlabeled data. This solution aims to use a small labeled data set to achieve a good classifier with high accuracy.

We also propose and explore two different hybrid methods in this paper which could overcome major limitations of the baseline methods. The first hybrid method replaces the human-in-the-loop component of AL with an ontology-based solution. The second uses dependencies extracted from the domain-specific ontology as training input to AL. These approaches are evaluated with two industrial data sets provided by two IT companies: Siemens, Austria and Blackline Safety Corp., Canada. In this paper, we refer to Siemens, Austria, as Company A and Blackline Safety Corp., Canada, as Company B.

The paper is organized as follows: Related work is discussed in Section 4.3, Section 4.4 presents the background on the dependency extraction problem. Section 4.5 describes the research method, research questions, baseline methods, industrial data sets, and the design of the whole empirical study in detail. Section 4.9 covers the empirical evaluation of the baseline methods. Section 4.10 introduces the two hybrid approaches and reports their evaluation. Section 4.11 discusses the overall results of the study. Section 4.12 details the threats to validity. Finally, Section 4.13 outlines the outcomes and the discussion.

4.3 Related Work

4.3.1 Approaches for Requirements Dependency Extraction

Dependency extraction has been explored as a special case of traceability in the past [44] [20]. J.N.och Dag et al. [124] explored "similar" dependency identification based on similarity measures. Chichyan et al. [39] analyzed the mechanism of how the semantics of unstructured requirements can be evaluated to identify the dependency types using NLP techniques.

Recently many studies have explored requirement dependencies and utilized NLP and ML to a great extent on requirements artefacts [78] [51] [19] [139]. Guo et al.' s study [78] is limited to extract the trace and do not explore the structural type of dependencies, which is a focus of our research.

Deshpande et al. [50] [51] used NLP and ML methods to extract dependencies. Weakly supervised learning methods were explored and various machine learners were utilized in this study on a public data set [70]. Samer et al. [139] analyzed small industry data sets and utilized Latent Semantic Analysis to extract the dependency types. However, these studies lack emphasis on domain-specific knowledge and need a large number of training sets.

4.3.2 Ontology-based Approaches

Ontologies are used in different requirements engineering activities [48]. As part of a holistic approach to requirement analysis, Verma and Kass [156] used a semantic graph expressed in OWL to represent a core requirements ontology. The dependencies are just one type ("affects") and are discovered using SPARQL queries. Other works [149] [172] use a domain ontology to create a requirements dependency graph and support some analysis. None of these papers provide empirical evidence on their benefits nor make any attempt to combine ontologies with ML techniques.

Guo et al. [79] proposed the use of manually generated ontologies in a related RE problem, namely term mismatch problem in trace retrieval solutions. The construction of an ontology followed a guided approach by augmenting the ontology with existing traceability knowledge. To alleviate the considerably high cost of constructing such an ontology, the authors suggested that an ontology created through leveraging trace links for one project can be next re-used in other projects.

In a similar vein, Li and Cleland-Huang [96] combined general and domain-specific ontologies to trace requirements with better accuracy than standard information retrieval techniques. To this end, the authors used a syntax tree and considered only noun phrases (representing mostly identifiers' names) and verbs (representing actions) for computing similarity scores between source and target artefacts. Nevertheless, the approach devised by Li and Cleland-Huang [96] lacks higher-level reasoning, as it is unable to capture more sophisticated concepts from the ontology.

Assawamekin et al. [18] utilized an ontology as a knowledge management mechanism to automatically generate traceability relationships. Guan et al. [77] showed how ontology and semantic web technology could be used to automate RDE.

Although these studies show an exciting outlook for Ontology in RDE automation, the effort needed to generate such ontology and their ability to handle complex grammatical patterns in requirement sentences written using natural language is not known; hence needs closer scrutiny.

4.4 Background

Requirement dependencies establish a relationship between requirements such that "progress of action on one requirement assumes the timely outcome of action on another requirement or the presence of a specific condition" [12]. Some authors refer to them as "interdependencies" [44] [95] to clearly distinguish from relationships between requirements and other artefacts such as code or test suites. Although the study of requirement dependencies is framed in the Requirements Engineering (RE) discipline, other communities have targeted the problem extensively with respect to aspect identification [131] or extracting dependencies among features such as variability modelling [151] and architectural analysis [157].

Research in the area has a long history that studied dependencies related to both syntactic and semantic criteria. While some authors have proposed taxonomies focusing on requirements engineering in general [128] [44], others focused on application areas such as release planning [37]. For example, Dahlstedt [44] provided the classification

of most fundamental dependency types such as Requires, Refines, Similar_to, Increases/Decreases_value_of etc. and classified them into Structural and Cost/Value interdependencies.

Recently, Zhang et al. [174] consolidated these taxonomies in the context of change propagation analysis and proposed a model that included nine types of dependencies. In this paper, we are particularly interested in two types of dependencies: *refines*, *requires*, whereas the remaining ones are labelled as *other*. Refinement was defined by Pohl's seminal proposal of taxonomy as "target object [*requirement*] is defined in more detail by another requirement" [128], whereas the requires relationship can be defined according to Carlshamre et al., as "R1 requires R2 to function" [37].

Not only academics but also software industry professionals have a voice on the topic. Deshpande et al. [55] report the results of a recent survey for requirements dependency extraction and maintenance, with 76% responses (out of 70) from practitioners. More than 80% of the participants agreed or strongly agreed that dependency type extraction is difficult in practice; dependency information has implications on maintenance, and ignoring dependencies has a significant impact on project success [55].

Previous studies have explored diverse computational methods that used Natural Language Processing (NLP) [124], fuzzy logic [123], predicate logic [176] and deep learning [78] techniques in the past. By considering their diverse nature, strengths and weaknesses, it is natural to explore the option of combining them into hybrid methods to obtain better results. In this paper, we choose two approaches (elaborated in Section 4.5.2):

- **Ontology:** Ontologies are widely used in RE research (see [48] for a survey). They are a well-suited conceptual artefact to manage knowledge. Particularly, domain ontologies [107] provide a formal representation of a specific domain serving as a means for communication and agreement. Hence, their use in activities such as dependency extraction is a reasonable choice to get domain-specific solutions.
- **Active Learning:** AL is a form of ML in which a learning algorithm interactively queries an *oracle* (typically a human expert) to obtain the desired label for new data points [141]. It has been effective in reducing human efforts in the data analysis process [60] [17]; therefore, it can be considered a good candidate for the dependency extraction problem.

Our motivation for selecting these two methods is that they are radically different approaches. While AL needs to start training from scratch for every new set of requirements, the use of a domain ontology enables knowledge reuse for projects of the same domain.

4.5 Research Method

In this section, we present the problem statement and then formulate the two research questions. Additionally, we introduce the two baseline methods in detail. Since this research is completely application-oriented, we describe the two industrial data sets first, followed by the configuration of the two baseline methods and the design of the empirical evaluation.

4.5.1 Problem Statement

The requirements dependency extraction problem aims to find and explicitly describe all existing dependencies between pairs of requirements. In this study, we aim to understand the effectiveness of two baseline methods that are variants of Active Learning (AL) and Ontology-based Retrieval (OBR). We analyze the outcomes and further evaluate the hybrid approaches by combining these two diverse methods, which are then evaluated on the two industry data sets. This study focuses on two different dependency types: *requires* and *refines*. All other extracted dependencies, i.e. those dependencies which are not classified as *requires* or *refines*, are subsumed as *other*. This categorization implies that these dependencies would require further analysis to determine their specific type.

Research Questions

RQ1.5: Are AL and OBR valid approaches for requirements dependency extraction from the perspective of industrial applications?

Justification: The two baseline methods have been explored in the existing literature; however, they lack industrial evaluation. From an application perspective, besides precision and recall measure, the amount of effort required to run the methods is an important selection criterion. The results obtained to answer this RQ are not just useful by themselves, but also pave the road to the following second research question.

RQ1.6: Can AL and OBR be integrated to form an improved dependency extraction method?

Justification: Results from RQ1 provide insights to both baseline methods, which enable us to design a hybrid approach that could reconcile their diverse perspectives and utilize their strengths to yield improved results.

4.5.2 Baseline Methods

In this subsection, we describe our AL method in-depth, which is designed specifically for this study and summarize the OBR method, which is already reported in the previous research by Motger et al. [115]. Also, we describe a NLP pre-processor that is common to both the methods.

Requirements Dependency Extraction by Active Learning (RD-AL)

Figure 4.1 depicts the workflow and various steps of the specialized AL-based method. We have utilized the general AL method [60], ensemble machine learning [100] and self-training method [102] [171] mechanism[1], to design a variant of AL-based tool which is explained as follows.

- **Initialization:** All the textual requirements information is passed through an NLP pre-processing pipeline first that generates a unigram representation of the textual data fields of all dependency candidates D (Step ❶),

[1]Self-training: A learner is first trained with a small labelled data set, and then it is used to classify the unlabelled data. Typically the most confident unlabelled instances, together with their predicted labels, are added to the training set, and the process repeats.

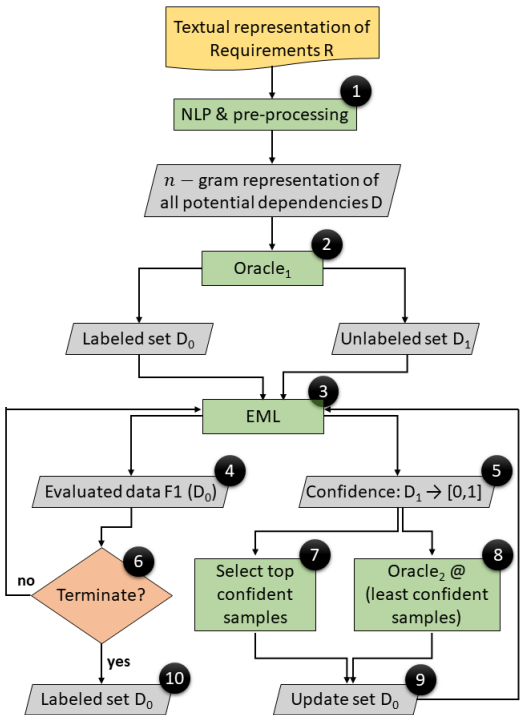


Figure 4.1: Step-wise flow diagram of Ensemble-based AL method for RDE

which is discussed further in Section 4.5.4. The output of this pipeline is used by baseline methods to build more complex n-gram representations (bigrams, trigrams, etc.) according to the specific needs of the algorithms. A domain expert - also referred to as $Oracle_1$ - randomly chooses a (typically small) set of potentially dependent requirement pairs and annotates them (Step ❷) to generate a training set.

- **Training using ensemble machine learning**: Thus prepared training set is used for training the machine learner. We utilize yet another ML technique called *ensemble machine learning* (EML) [61] in this step. EML combines individual classifiers' predictions by using their strengths and diluting their weaknesses, which improves the prediction performance of individual classifiers (Step ❸). EML has proven to be effective compared to individual classifiers in problems such as app review classification [82] and software defects prediction [112]. Our method uses three classifiers: Naive Bayes, Random Forest and Support Vector Machine as they are simple classifiers to work with textual data [121] [82] and provide a probability of an instance belonging to a class which is indicative of the uncertainty (e.g., for a binary classification problem, the closer the probability of a prediction is to 0.5, the higher the uncertainty of classifier's prediction) [60].

- **Choose least confident samples for manual labeling:** The EML learner is trained with an initially small amount of labelled samples which is then used to classify the unlabelled data (Step ❺) applying *pool-based sampling* [141] technique. This is well-suited for sampling the most uncertain data point or instance(s). All requirement pairs in the unlabelled data pool are then evaluated according to their informativeness. Informativeness is computed using *least confidence uncertainty sampling* [141], which selects the instances with the

lowest confidence level (probability value) to be labelled next (Step ❺, ❽). In every iteration, n most uncertain predictions are shown to the $Oracle_2$ and labels are acquired (Step ❾).

- **Choose additional labels (a SSL method):** A unlabled pair of requirement for which the classification probability value (confidence value) is higher than the (preset) *confidence* threshold, is chosen to be added to the training set [100] (Step ❼, ❾).

  The selection of $confidence$ threshold and n is a trade-off between the accuracy of the classifier and the efficiency of the trainer. In every iteration, the additional labels selected through SSL and instances labelled by $Oracle_2$ are added to the training set and the process repeats until one of the termination criteria is met.

- **Termination criteria:** AL stops if either a predefined number of iterations is done, or if the degree of improvement is minimal, or if no more unlabelled samples are available (Step ❻, ❿).

We implemented this specialized AL method as a tool called *RD-AL*. RD-AL has been implemented in Python utilizing Scikit-learn libraries [127] for voting classifier based ensemble classifier. Ensemble VotingClassifier combines conceptually different machine learning classifiers and uses a majority vote (hard voting) or the average predicted probabilities (soft voting) to predict the class labels. Such a classifier can be useful for a set of equally well-performing models to balance out their individual weaknesses. Since AL requires probability values to pick the most uncertain and confident instances in every iteration, we have utilized *soft voting classifier*, which generates the averaged predicted probabilities for classification in this research.

Requirements Dependency Extraction by Active Learning (RD-AL)

For RD-AL we utilize the AL already defined in the section above but with an Ensemble ML.

4.5.3 Requirements Dependency Extraction by Ontologies

For the OBR method, we utilize the OpenReq-DD tool that uses domain ontologies [115]. The ontology defines dependency relationships between specific terms related to the domain of the requirements. Using this information, it is possible to apply NLP techniques to extract meaning from these requirements and relations. Further, ML techniques could also be applied for conceptual clustering to classify the requirements into the defined ontology.

OpenReq-DD is exposed as a RESTful service with an API[2] to provide the required data and perform the dependency extraction. The ontology is built using the W3C Web Ontology Language (OWL[3]). The output of the OpenReq-DD service is a set of extracted dependencies as a JSON response.

Figure 4.2 shows the sequence of steps that OpenReq-DD performs to extract requirement dependencies. After data pre-processing (see Subsection 4.5.4), the semantic analysis performs two operations: 1) It generates a dependency tree where each node is a token of the input sentence and edges are the relations between parent words and child words; 2) extracts keywords to categorize each requirement, using a TF-IDF_BASED algorithm which uses

[2]https://api.openreq.eu/dependency-detection/swagger-ui.html
[3]https://www.w3.org/OWL

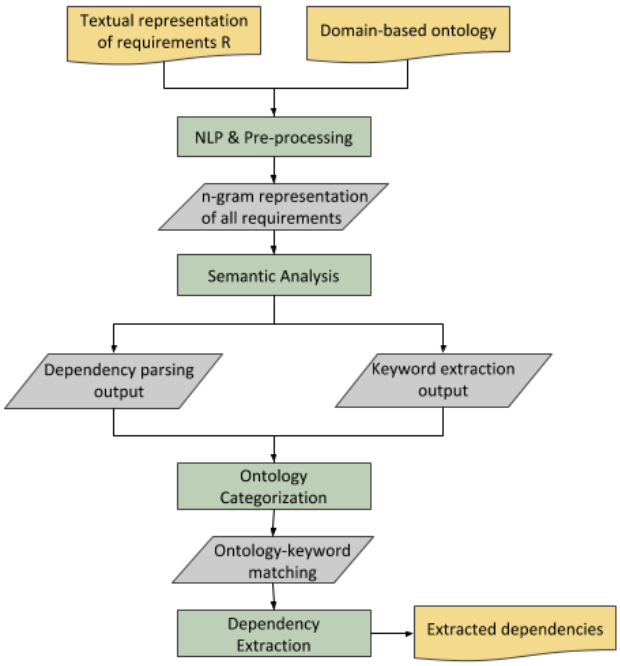


Figure 4.2: OpenReq-DD technical workflow representation

frequencies to find most relevant, significant keywords among the requirement corpus. Next, the ontological categorization step classifies requirements into the different concepts of the domain ontology, which is the input to the final dependency extraction step, based on matching ontology concepts and relationships with clustered requirement keywords. See [115] for more details.

4.5.4 Natural Language Pre-processor Pipeline

To make both methods comparable and avoid bias in the initial step, we have implemented a state-of-the-art NLP pre-processor component which is deployed as a decoupled Java-based tool. This component applies a set of NLP techniques to improve the quality of textual requirements data and to apply a lexical analysis. Thus generated output is then fed to both the RD-AL and the OpenReq-DD tools for further processing.

The NLP pre-processor pipeline implements the following techniques:

- *Noisy Text Cleaning.* A set of 14 sentence cleaning rules which include the removal of non-relevant elements like list pointers or escape sequence characters.
- *Stop-word Removal.* Most common words that are not relevant for the syntactic and semantic analysis of a sentence are filtered out of the requirements text.
- *Standardization.* Words or phrases that are not recognized by standard dictionaries (such as acronyms) are replaced for understandable words.
- *Tokenization.* Each requirement sentence is split into a bag of words or tokens using the Apache OpenNLP

toolkit[4].

- *Stemming.* Each token is converted into its root or stem using the KStemmer from the NLP4J toolkit[5].

Table 4.1: Information about industry data input from the two companies A and B

	Company A	**Company B**
Product	Train control system	Safety critical system
# Requirements (n)	310	93
# Potential dependent pairs: n(n-1)/2	47,985	4,278
# Annotations aquired	273	191
# Ontology classes	28	21
Rough Ontology construction effort (hrs)	5hrs	10 - 12 hrs

4.6 Industrial Data Sets

Research questions are investigated on the two industrial data sets, as summarized in Table 4.1 and described below.

4.6.1 Company A - Siemens, Austria

This is a multinational conglomerate and the largest industrial manufacturing company in Europe. For this research, we use a collection of Request For Proposals (RFPs) documents of the railway domain. These requirements refer to specific technical features of the railway domain, including the reconstruction, rehabilitation and maintenance of tracks, industrial materials, related legislation, software systems and electronic interlocking installations.

For our study, we selected 310 requirements from the RFPs documents. This selection was assessed by Company A's domain experts, who chose a highly related subset of requirements from the Radio Block Center (RBC) railway sub-domain to guarantee the extraction of relevant dependencies. From these 310 requirements, 47,895 requirement pairs were generated, from which 273 pairs are manually annotated (53 as requires, 65 as refines, 30 as others, 125 as non-dependent). These requirement pairs are candidates for the dependency extraction process.

4.6.2 Company B - Blackline Safety Corp., Canada

This is a world leader in the development and manufacturing of wireless safety products. For this study, we use the requirements of an area monitoring safety product built to monitor confined and open areas for safety hazards. It is a complex product that uses hardware, firmware, and software technologies. This product has intricate and inconspicuous dependencies in the manufacturing and release levels.

Currently, company B extracts dependencies exclusively manually. The domain knowledge is distributed over different documents and owned by different people, hence, the complete extraction and dependency management

[4]http://opennlp.apache.org
[5]https://emorynlp.github.io/nlp4j

Table 4.2: Sample requirement pair examples from the two data sets and their dependency type

	Requirement 1	**Requirement 2**	**Dependency**
A	TND_2366: All acceptance tests for the ETCS level 2 system shall check the content of the telegrams emitted/received by the RBC and the balises	TND_2145: Shunting routes need not be recorded in the RBC	*Requires*
B	Movement detection setting or enabling movement detection should be initiated from the device's menu and also remotely from the portal	For anti-theft, special locking mechanism will not be provided. If the owners are worried about theft they can bolt it	*Refines*

process is not only effort intensive but also time consuming. Additionally, there is a risk of missing dependencies. For our study, this company provided a document of 93 requirements which resulted in 4,278 potential requirement pairs. Of these, 191 pairs were manually annotated (43 as requires, 68 as refines, 4 as others, 80 as non-dependent). Table 4.2 shows examples from companies A and B.

4.7 Configuration of the Methods

4.7.1 Configuring the Ensemble-based AL Approach

For classifier model training, we randomly chose 20% of the data and retained it as the test set; the remaining 80% is used as the training set. The same test set (unseen data) is used in each iteration of AL to compute the classification ($F1$) scores. There are three possible stopping criteria to terminate the learning process [141]: 1) Desired classification accuracy is obtained or accuracy start to degrade, 2) Labelling budget is exhausted, 3) There are no more unlabelled requirement pairs. We chose to terminate if the accuracy does not improve over a certain number of iterations (up to 10).

In order to strike a balance between the classifier's accuracy and trainer's efficiency, we set the number of annotations per iteration to three (n=3). Also, to minimize the addition of high confidence dependent pairs to the training, we set the $confidence$ threshold to 0.9.

4.7.2 Configuring the OBR Approach

We built domain ontologies for both the products[6] in collaboration with the respective product's domain experts. For Company A, the ontology was developed collaboratively by some of the authors of this paper who are experts in ontologies in general, and domain experts. Keyword elements and relationships were established based on an analysis of the data set provided by Company A. The result takes the form of a formatted ontology file (using OWL syntax).

Figure 4.3 shows an excerpt of the ontology for Company A, comprising a small subset of key concepts and their dependency relationships, including two dependency types: *requires* and *refines*. We could see how this ontology

[6]Due to confidentiality reasons, we refrain from providing ontology details for Company B

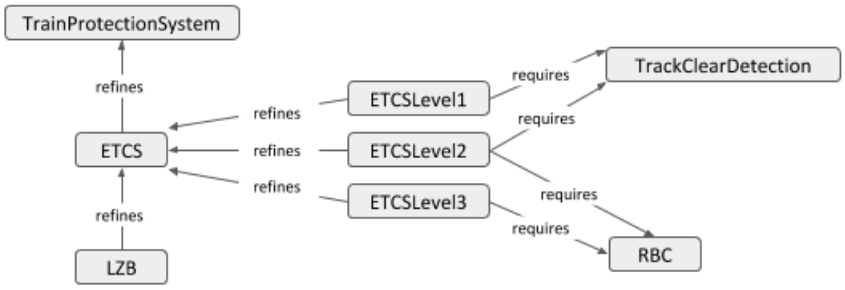


Figure 4.3: Railway domain ontology example of Company A depicting various key concepts and their relationships and types

conveys the necessary information to automatically extract the dependency among requirements TND_2366 and TND_2145 listed in Table 4.2, considering the relationship "ETCSLevel2 *requires* RBC".

The design and elaboration of the domain-specific ontology for Company A (consisting of 28 classes) took 5hrs of effort as it was designed alongside ontology experts from the development team. Company B's ontology (consisting of 21 classes) was also developed in collaboration with the domain expert of this product over 12hrs in multiple iterations and meetings.

4.8 Design of the Empirical Investigations

The overall design of the study has been outlined in Figure 4.4. We use Company A and Company B requirements as the two industrial data sets to build two independent empirical evaluation scenarios. These include a set of pre-processed requirements using the preprocessor pipeline and a domain ontology file designed by domain experts from each company. This input has been in three different evaluation set-ups: (a) The OpenReq-DD baseline evaluation, (b) The AL-RD baseline evaluation, (c) The two different Hybrid approaches, discussed in detail in Section 4.10. For each one of the three evaluation scenarios, we provide two independent empirical analysis using Company A and Company B data sets. Each combination provides a set of dependencies as a result.

RQ1.5 is devoted to analyzing the performance of the two baseline methods for the two industrial data sets whereas RQ1.6 is for the evaluation of the two hybrid methods stemming from AL and OBR approach.

4.9 Empirical Evaluation - RQ1.5

In this section, we present the evaluation, results and analysis for RQ1.5. The results are explained through accuracy values which are supported with qualitative and comparative analysis of the two methods from an efficiency perspective.

For the evaluation, we have used 10 times 10-fold cross-validation. Each of the classifier statistics has been analyzed using precision, recall and $F1$ measures. as defined in Equations 1, 2, and 3, respectively.

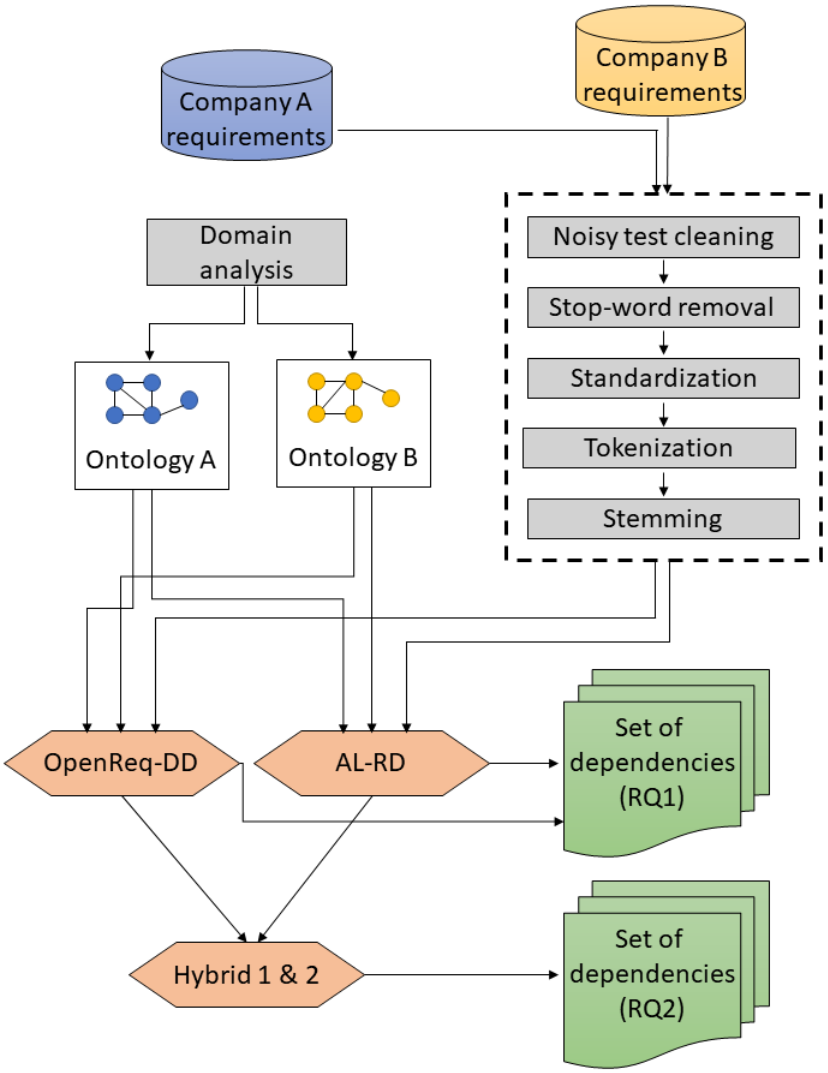


Figure 4.4: Research design of our study which utilizes two industry datasets, their respective ontolgies, and OpenReq-DD and AL-RD methods

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4.3}$$

In the formulae above, TP represents the # of requirement pairs that are truly classified positive. Similarly, FP and FN represent the # of requirement pairs that are classified as false positives respectively false negatives.

The evaluation results are summarized in Table 4.3. $F1$ is computed for the test data (unseen data). The standard deviation (STD) has been specified for the cross-validation score to provide information regarding the variance between multiple runs. Special care has been taken to have balanced data sets to ensure an unbiased analysis.

The ensemble-based AL approach (see Figure 4.1) generates the final output in two stages. Firstly, a coarse classification of dependent and independent requirements is performed. Secondly, only the dependent requirements pairs are used further to classify into fine-grained dependency types in the second stage.

The evaluation results are summarized in Table 4.3. $F1$ is computed for the test data (unseen data). The standard deviation (STD) has been specified for the cross-validation score to provide information regarding the variance between multiple runs. Special care has been taken to have balanced data sets to ensure an unbiased analysis.

The ensemble-based AL approach (see Figure 4.1) generates the final output in two stages. Firstly, a coarse classification of dependent and independent requirements is performed. Secondly, only the dependent requirements pairs are used further to classify into fine-grained dependency types in the second stage.

4.9.1 Results Company A

As summarized in Table 4.3, OpenReq-DD extracted 1,608 dependencies. Among them, there were 501 *refines* dependencies, 1,107 *requires* dependencies and zero *other*. The 10×10 cross-fold validation showed an average $F1$ score of 85% and a standard deviation (STD) of ∓ 0.07.

The RD-AL tool was fed with all 47,985 potentially dependent pairs of requirements. Of these pairs, 273 pairs were randomly selected and annotated by a domain expert. The multi-class classification stage, RD-AL extracted 1,656 *requires* dependencies, 1,758 *refines* dependencies and 4,871 *other* dependencies. The multi-class classification $F1$ score was 72% and the 10×10 cross-fold validation average was a 92% $F1$ score with ∓ 0.02 STD.

4.9.2 Results Company B

The OpenReq-DD tool extracted 154 *refines* and 44 *requires* dependencies. The 10×10 cross-validation resulted in an average $F1$ score of 76% with a ∓ 0.21 STD. The RD-AL tool was fed with all potentially dependent pairs of requirements generated out of 93 requirements. Of these, 200 pairs were randomly picked and annotated by a domain expert to form a seed training set. In the second stage of fine-grained multi-class classification, the tool extracted 871 *refines*, 416 *requires* and 65 *Other* dependencies. Once again, the operation was terminated when the accuracy plateaued at 49.87% $F1$ score, while the 10×10 cross-validation accuracy showed 86% accuracy and ∓ 0.14 STD.

4.9.3 Analysis

Due to the varying size of the two industry data sets, the amount of effort and time taken for executing the RD-AL approach were different. Company A's data set needed about 3hrs of domain expert's time of which 0.5hrs was to provide the annotations in every iteration. Similarly, Company B's data set needed one hour of domain expert's time including 0.25hrs for annotation.

OpenReq-DD approach is solely based on the ontology, thus, the efforts and time needed for its construction was a one-time activity. Conversely, RD-AL approach needs greater effort since a domain expert was actively involved in the process. However, the findings indicate that the number of dependencies extracted by RD-AL was higher compared

Table 4.3: Baseline results for the two approaches

	Annotated data	**RD-AL**	**OpenReq-DD**
Company A	273 pairs	**Refines:** 1,758 **Requires:** 1,656 **Other:** 4,871 **F1**: 72.0% **10×10:** 92%(∓0.02)	**Refines:** 501 **Requires:** 1107 **10×10:** 85%(∓0.07)
Company B	191 pairs	**Refines:** 871 **Requires:** 416 **Other:** 65 **F1:** 49.9% **10×10:** 86%(∓0.14)	**Refines:** 154 **Requires:** 44 **10×10:** 76%(∓0.21)

to OpenReq-DD. RD-AL uses a top-down approach where it considers all the potential pairs of the requirements and uses an oracle to acquire additional knowledge. In contrast, OpenReq-DD tool used a bottom-up approach using ontology as its direction to achieve the same objective. As a result, the dependencies extracted by the OpenReq-DD tool are smaller in number (higher chances that a few of the dependencies could be missed), whereas RD-AL extracted them in a larger numbers (higher chances of false positives).

RQ1.5: For both case studies, RD-AL was extracting more dependencies than OBR. The results naturally lead to the evaluation of the hybrid approach of the two methods, which could complement each other to improve performance and reduce efforts.

4.10 Empirical Evaluation - RQ1.6

This section elaborates the two hybrid approaches and their evaluation. Ontologies in general help capturing the domain-specific knowledge which can be further analysed to provide a deeper analysis of requirements documents [156] [80] [149]. However, constructing an ontology is time-consuming [79] and has been proven to be difficult to automate for repeated use in the advent of changing and evolving software products [80].

Conversely, AL works on the uncertainty sampling (active sampling) instead of random sampling selection strategy [141]. Hence, AL seeks to minimize the human effort required for training a classifier by intelligently selecting an unlabelled sample for labelling via uncertainty sampling over multiple iterations. However, manual annotation by an oracle and seed train set are the most important aspects of the AL approach. Hence, these can have immense impact on how fast the AL could converge to stopping criteria [60] [100].

Also, there have been studies which evaluate alternatives to *oracles* (which are typically domain experts otherwise) to overcome the noise that a single human expert could add due to fatigue, boredom, inconsistency etc. Such as crowdsourcing or multiple annotators etc. [141].

Building on the strength of domain-specific ontologies and the power of AL, in this subsection, we discuss two

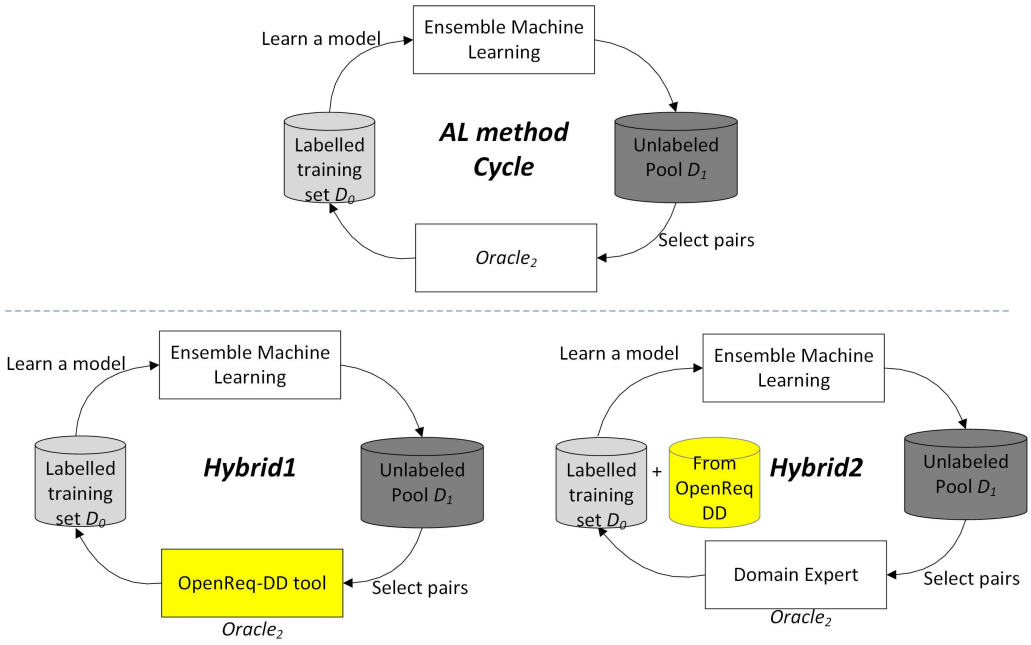


Figure 4.5: Comparison of AL with two hybrid approaches. In the first (Hybrid1) we use OpenReq-DD tool as an oracle and in the second (Hybrid2) we used its output as an additional dataset for training AL-RD

hybrid approaches and leverage on their respective benefits. These approaches use ontology reasoning as (i) a substitute to the human expert ($Oracle_2$: Figure 4.1) and (ii) as an extension for the training set for running the AL approach. Although the AL tool was developed in Python and OBR in Java, python wrapper was developed to communicate among the two tools through API. Figure 4.5 shows how the OBR tool was used as a plug-in into the AL tool to construct the hybrid approaches.

4.10.1 OBR as Oracle for Classification (Hybrid1)

In the AL approach (Figure 4.1), $Oracle_2$ (domain expert) assigns a label to each of the least confident requirements pairs in every iteration. The Hybrid1 approach replaces the human expert by the OpenReq-DD tool as a $Oracle_2$. At each iteration of the AL cycle, the least confident requirement pairs are automatically sent to the OpenReq-DD tool using its REST interface to autonomously predict whether the requirement pairs are dependent or not (stage 1) and which type of dependency are they classified to (stage 2). In this way, we examine the effectiveness of the tool support for classification.

4.10.2 Dependencies Extracted from the Domain-specific Ontology as Training Input to AL (Hybrid2)

To create a synergy between the two baseline methods, Hybrid2 uses the dependencies extracted using ontology-retrieval (Figure 4.2) method as an additional input for the training set (D_0, in Figure 4.1). This is achieved by

Table 4.4: Hybrid1 results: OpenReq-DD tool as an oracle in RD-AL approach

	Annotated data	**RD-AL output**
Company A	273 requirement pairs (positive samples)	**Refines:** 1,783 **Requires:** 5,448 **Other:** 854 **F1:** 75.0% **10×10:** 96%(∓0.02)
Company B	191 requirement pairs (positive samples)	**Refines:** 102 **Requires:**231 **Other:** 219 **F1:** 82.6% **10×10:** 92%(∓0.17)

running a prior analysis using the OpenReq-DD baseline approach, and this output is then fed as the labelled training set for the AL-RD baseline method. As a result, initial labelled data turns out to be a combination of the human expert annotation process and the automatic OpenReq-DD classification output.

As a consequence, the chances of learning are improved by gathering input from two different types of oracles: human expert and ontology retrieval. As shown in Figure 4.5, this approach compares with the AL method described in Figure 4.1.

4.10.3 Results Company A

Tables 4.4 and 4.5 provide a summary of the results explained in this section. The tables describe the impact of variants of hybrid methods which try to strike synergy between RD-AL and OBR. For Hybrid1, where OpenReq-DD replaced the human oracle in RD-AL, the $F1$ score increased from 72.0% to 75.0% when compared to the baseline method. The 10×10 cross-validation analysis reported a 96% $F1$ score with STD of ∓0.02.

For Hybrid2, where the output from the OpenReqDD-tool (1,608 dependencies) was added to the training set, showed an $F1$ score of 76.7% which is about 3% higher than the baseline accuracy. The 10×10 cross-validation $F1$ score improved to 95% from 92% with a STD of ∓0.02. The human effort required by Hybrid2 is a combination of the effort required by OBR and the human effort of the AL baseline method, which include the design of the ontology and the feedback provided to the active learner cycle as $Oracle_2$.

4.10.4 Results Company B

For Hybrid1, the $F1$ score showed an increase to 82.6% compared to the baseline 49.9%. Also, the 10×10 validation score showed an improvement of 6% to become 92%.

For Hybrid2, when the output from the OpenReq-DD (191 dependencies) was added to the existing training set

(200 dependencies), the accuracy increased by 25% to the level of 75.8% in the multi-class classification compared to baseline accuracy. of the RD-AL tool at 49.87%. The 10×10 cross-validation accuracy was 92% with STD of ∓0.05.

Additionally, to measure the potential of the ontology to replace the domain expert as an oracle (Hybrid1), intersection of the annotations, provided by the domain expert (in the baseline results), with OBR was conducted. The results showed that the Company A and B's tests had 56% and 50% overlap, respectively.

Table 4.5: Hybrid2 results: Training set combined with dependencies from OpenReq-DD tool

	Annotated data	**RD-AL output**
Company A	273 pairs + 1609 OpenReq-DD dependencies (positive samples)	**Refines:** 4,292 **Requires:** 7,530 **Other:** 1,151 **F1:** 76.7% **10×10:** 95%(∓0.02)
Company B	191 pairs + 198 OpenReq-DD dependencies (positive samples)	**Refines:** 585 **Requires:** 311 **Other:** 207 **F1:** 75.8% **10×10:** 92%(∓0.05)

4.10.5 Analysis

Firstly, when comparing Hybrid1 and Hybrid2, the former generated better results for Company B, whereas the latter showed better results for Company A. Secondly, Hybrid1 results indicate that the domain-specific ontology could cover for approx. 50% of the efforts of a domain expert. More concretely, this implies that OpenReq-DD does not extract all the dependencies present in the data set and provides partial knowledge of the complete representative data of the unlabelled data set. Conversely, considering the sum of the estimated effort required by $Oracle_2$ in every iteration during RD-AL execution (as reported in Section 4.9), Hybrid1 reduces half of the human effort required for dependency extraction for these two companies.

Lastly, Hybrid2 outcomes show more promising results. There is a clear improvement in the $F1$ score: 3% for company A at 76.7% and an increase of 25% for company B at 75.8%. This disparity could be attributed to the diverse nature of these two data sets. While one is a pure software-based product, the other belongs to a more complex and multi-component project. However, the results indicate that OBR output helped classifier to learn the representative of the data set with this approach.

RQ1.6: For Hybrid1, the $F1$ score showed an increase to 82.6% compared to the baseline 49.9%. For Hybrid2, the accuracy increased by 25% to the level of 75.8% in the multi-class classification compared to baseline accuracy. OBR also complemented the AL approach by reducing 50% of the human effort.

4.11 Discussion of results

Despite its exploratory nature, this study offers some interesting insight into the relevance of the two baseline methods to the real-world (industry) data sets. The findings show that our AL-based implementation tends to create false positives. However, this could be a positive aspect when weighed against the adverse impact it could have had on the overall product success (at least in the real-world context) if dependencies were overlooked or omitted. Additionally, the ontology-based approach appeared to be conservative in the extraction process; however, this does not claim external validity.

Although there are different approaches to extract requirements dependencies, there is limited or no evidence of their suitability in real-world settings. In fact, we think that there are no easy answers to the question: how different methods perform under different circumstances? Through this research, we hope to answer this question and make progress, even though it may appear minuscule, towards reaching the bigger goal.

Based on the results from just the two industrial case studies, we do not claim to answer the fundamental question, "Which method works better and in what circumstances?". Instead, we argue that the industrial perspective goes beyond simple $F1$ measurement and includes other crucial aspects that are especially related to effort and impact. Thus, we could draw one such strong conclusion from this research, which is that creating ontologies becomes more valuable, the more it is re-used subsequently in different ways. However, measuring this value was outside the scope of our investigation.

The return of investment (ROI) is defined as the ratio of the current value of the investment and the cost of investment. ROI $\geqslant 1$ indicates that the value (in a given time interval) exceeds the cost of investment and thus is profitable. Consequently, we believe that even a small improvement in extracting requirements dependencies is valuable considering the adverse impact of missing some of them.

In summary, the investment into the hybrid approach is justified by the investigation of ROI: How much rework effort is saved and quality improvement is achieved from the cost of extracting more number of requirements dependencies? The answer is context-specific, but we expect a bigger than one ROI from investing in a hybrid extraction method, in particular, in the case of acquiring additional training samples to generate better models.

Ontologies are intended to provide knowledge engineers with reusable pieces of declarative knowledge, which can be – together with problem-solving methods and reasoning services – easily assembled into high-quality and cost-effective systems [122]. Ontology is created by synthesizing the knowledge elicitated from domain experts into comprehensible mapping.

While it can not be determined if a specific hybrid system integrating ontology-based techniques in the AL cycle is

better than an alternate approach, empirical evaluation proves that both baseline methods improved when integrated to counterbalance the main weaknesses of each one of them.

Finally, as the product manager of Company B summarized the value addition of the proposed hybrid methodology, " The safety-critical functionalities of the product and the evolution of the requirements as part of the product incremental development cycles exert tremendous pressure to identify as many dependencies as possible. Hence, decisions at the fine granular level have a larger impact on quality, cost, resource utilization and time-to-market".

4.12 Threats to validity

4.12.1 Internal Validity

1. Our EML model comprises of the basic and most used ML algorithms in text-based research. However, EML can be enhanced to incorporate additional algorithms and tested for its impact on the classifier performance.
2. RD-AL utilizes the least confidence measure to compute the uncertainty index. It remains to be seen how other measures, such as Min Margin and Label entropy [141], could alter the results.
3. We used the SSL method to select and add a small portion of the most confident sample (step ❼, Figure 4.1) data to the training set in every iteration. However, some of these requirement pairs may be wrongly labelled which could have an impact on the classifier performance.
4. Hyper-parameter tuning for the machine learners has not been explored in this study. Since this needs additional computational time and resources and thus was not feasible to be carried out in every iteration of AL. However, we do not rule out its impact on the results.
5. Similarly, to have the same measures, we chose $F1$ in both companies but in the case of Company A, since they are more interested in minimizing false negatives, other values of F_k would be a more precise measure for their business case.
6. In this study, synonymy evaluation is not used. Exploring intelligent mechanisms such as semantic similarity using WordNet database could have a significant impact on NL procedures like ontology categorization.
7. Data annotation was carried out by domain experts from companies A and B. Thus, we do not rule out the adverse impact of bias in the initial training set.
8. We did not consider directionality in the dependency set. Additionally, we did not consider the feature engineering process of extracting word and sentence-level features. While these two treatments could have improved the prediction power of our approach, reported results are not compromised.

4.12.2 Construct Validity

The level of detail and the completeness of the ontology are known to have a substantial impact on the overall retrieval results. It remains open how complete these ontologies for the two industry data sets are. As a matter

of fact, ontology design was not specifically focused on a three-type dependency approach and, hence, the generic *other* dependency type is not mapped in the ontologies. A refinement iteration of this third generic type and its design on the ontology could have an impact on the OBR baseline method results. Additionally, we suspect that the language-based noise in the raw data in the form of grammar and semantics could have an impact in general on the outcomes.

4.12.3 External Validity

The experiments carried out for this research are for just two diverse and different data sets. Hence, the results cannot be generalized. Large scale empirical studies, either on industry or open-source software repositories, could benefit to arrive at more general conclusions.

4.13 Discussion

In this study, we proposed a variant of AL, that combined AL, ensemble and SSL to extract requirement dependencies. We also compared it with the OBR approach for two industry data sets. Results showed that AL extracted more dependencies compared to OBR, thus, we designed two hybrid approaches to evaluate how well these baseline methods could complement each other to yield nuanced results. Hybrid1 results showed that it is possible to reduce the human effort required in the AL while improving the reliability of the classification output. Hybrid2 demonstrated that conservative dependency extraction results could be used as input for AL to improve results and provide visibility to new and undetected dependencies.

Requirement dependencies extraction is a difficult task, and no single solution is expected to solve the problem. In fact, there is no "solution" to this problem. Instead, a few existing industrial studies confirmed that each evolutionary improvement could help to improve the product development process from a real-world perspective. Our research on industry data sets shows that improvement not only refers to the formal accuracy ($F1$ value) but also includes the decreasing effort to extract dependencies and the ability to support knowledge management of the company. The latter aspects are hard to quantify but are proven highly relevant [137].

Chapter 5

Cross Project Dependency Extraction

5.1 Introduction

Transfer Learning aims to extract the knowledge from one or more source tasks and applies the knowledge to a target task [120]. A cross-project dependency extraction (CPDE) is a form of Transfer Learning when an ML model is trained using sufficient training data from existing source projects and then used on a target project to predict[1] requirements dependencies. This definition is similar to cross-project defect prediction [170] [109] which has been adapted to the context of RDE. The target project could be a new project or a project with limited training data. In other words, Cross Project Dependency Extraction (CPDE) uses data from one project (source) to build the predictive model. It is then used to predict dependencies in another project. Essentially, CPDE aims to solve the problem of limited data available to train within a project. Hence, we utilize conventional ML methods such as NB, RF and SVM for evaluating Transfer Learning in the context of RDE.

Recently, BERT, a Deep Learning (DL) language model, pre-trained on a large textual corpus such as English Wikipedia, has proven to effectively address the Transfer Learning challenges when fine-tuned with task-specific dataset [84] [98]. Thus, to evaluate CPDE applicability for RDE, we fine-tuned the base BERT model using RDE specific dataset of a larger (source) project and evaluated its performance for dependencies extraction for smaller (target) projects.

5.2 Research Method

As a natural progression to addressing the lack of training data for dependency extraction, we explored RF, NB and SVM for CPDE. Utilizing data from six different projects of Bugzilla, we analyzed classification for binary and multi-class classification problems related to RDE. Also, we evaluate fine-tuned BERT using binary class RDE dataset as multi-class data used in this study is scarce (please refer to the statistics in Table 3.1. After filtering requirements

[1]We use prediction and extraction interchangeably to align with the ML terminologies for better readability.

sentences with less than five words, *Other* class samples for most of the projects were left with fewer dependent pairs) to train DL methods.

5.2.1 Data

We chose the top six projects from the Mozilla family of products: Core, Firefox, DevTools, Toolkit, Testing, and Firefox Build Systems. This dataset has been described in Section 3.4 in detail. We extracted 17,337 requirements [26] and their related information for 15 Mozilla family of products from Bugzilla. We interpreted *depends_on* and *blocks* fields as "*Requires*" dependency type. So, further processing resulted in 49,492 pairs being interdependent where the dependency type was "*Requires*". For the *see_also* dependency type, which is documented as a relationship of the type "related issue" has been interpreted as *others* dependency type. Please refer to Section 3.4.2 for more details regarding this interpretation explained with an example. We also generated 1,474,772 requirements pairs from these requirements, which had no dependency between them as a negative sample data set.

To determine the similarity between the projects we utilized number of intersecting dependency pairs for any two given projects from the chosen 6 projects for this study. Figure 5.2.1 shows the distribution of the pure and intersecting requirements pairs. If a pair of requirements belong to same project then they are pure requirements, if not then they are termed as intersecting requirement pairs. We further grouped them project-wise and closely analyzed further. The heatmap depicted in 5.2 shows this information for all the projects. Firefox project is a biggest of the six. However, Core project intersects maximally with most of the projects as shown in the Figure

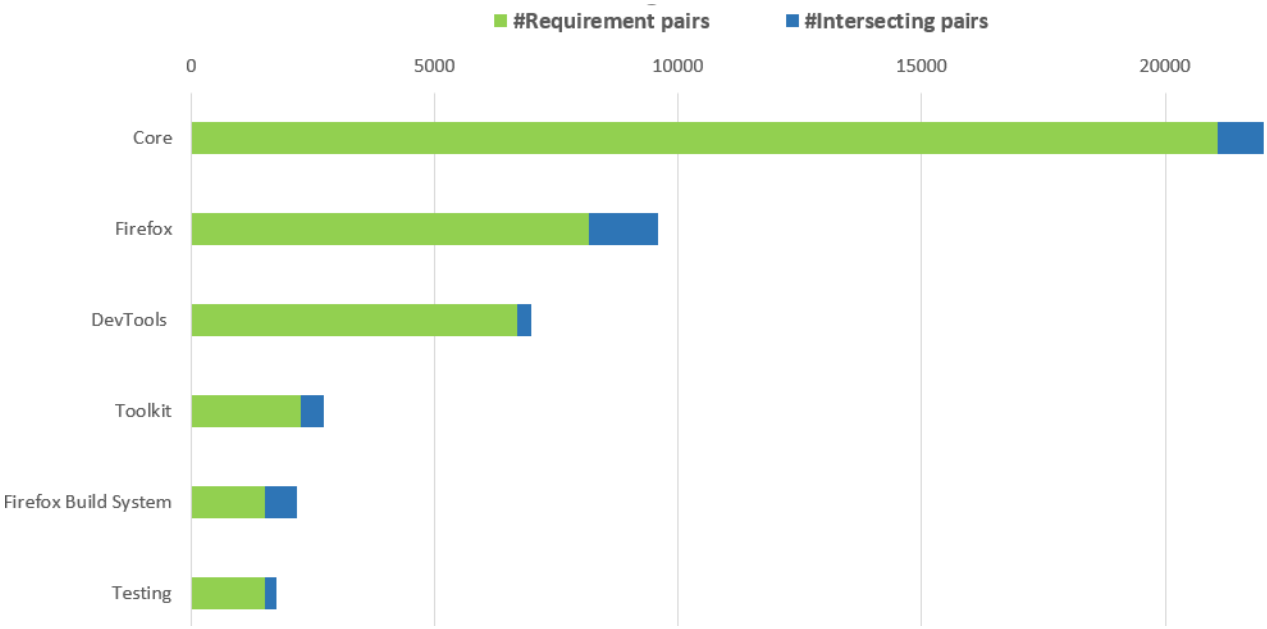


Figure 5.1: For the six projects selected for CPDE study, count of total requirements pairs are depicted using green color. Blue color depicts the number of pairs that have one requirement belonging to other project

	Core	Firefox	DevTools	Toolkit	Testing	Firefox Build System	#Intersecting pairs
Core		46.8%	4.0%	24.9%	10.9%	13.3%	952
Firefox	51.8%		3.8%	36.2%	1.0%	7.2%	1095
DevTools	91.5%	2.4%		1.6%	2.0%	2.4%	246
Toolkit	48.4%	42.2%	2.8%		3.8%	2.8%	289
Testing	49.5%	8.3%	2.6%	7.3%		32.3%	192
Firefox Build System	76.9%	5.4%	0.3%	7.5%	9.9%		577

Figure 5.2: For a pair of requirement tuple say *(A,B)*, *A* belongs to project on x-axis and *B* on y-axis. This heatmap shows the percent-wise intersection for all the six projects

5.2.2 Research Questions

We explored answers to four research questions as follows:

RQ2.1 In terms of accuracy, how do Within Project Dependency Extraction (WPDE) models compare with CPDE for binary (Dependent/Independent) classification?

Justification: To verify the applicability of transfer learning in the context of RDE, we chose to perform a high-level test - Binary dependency extraction - to identify the best performing ML algorithms first. RF, NB and SVM were chosen for building extraction models and tested on the six Mozilla family projects.

RQ2.2 In terms of accuracy, how does WPDE models compare with CPDE models for fine granular (multiclass: *Requires*, *Independent*, *Others*) dependencies?

Justification: Based on the results from RQ2.1, we then chose the best performing ML algorithm to further classify fine granular dependencies as the second stage of transfer learning application in the context of RDE.

RQ2.3 For version wise classification, does CPDE outperform WPDE in terms of accuracy?

Justification: Since software development is incremental and iterative, it was essential to explore Transfer Learning in the scenario of lack of data for training in the initial stages of software version development. Hence, we explored strengthening Transfer Learning by utilizing knowledge of dependencies over the versions as an additional data source.

RQ2.4 How does fine-tuned BERT perform for CPDE in terms of accuracy compared to other conventional ML techniques?

Justification: BERT models are pre-trained on a massive dataset from Wikipedia. Thus they have proven to understand the context of textual content. Essentially, BERT can automatically detect the features (textual representations) needed for classification or detection, which, when learned by the classifier, allows better generalization even over new or (and) unseen data [177] [84]. Thus, we test it for CPDE. As such, the BERT model is further fine-tuned using domain-specific data, which is RDE data in our case.

5.2.3 Experiment setup

This study aims to evaluate CPDE for RDE and compare it with WPDE for feasibility. As such, we analyze the extent to which TL could address the limited training set problem for RDE.

We first preprocessed the data set and used Natural Language Processing to remove stop words, special characters and numbers. Finally, lemmatization was carried using the Stanford NLTK tool kit [1]. We implemented the following binary and multiclass classification algorithms for various empirical studies.

CPDE pseudo code

Train set: S consisting of requirements pairs and their class labels where, label can be 0:*Independent*, 1:*Requires* and 2:*Others*.
Test set: T consisting of requirements pairs. Although label is known it is only used during performance evaluation.

1 Using labeled training set S, obtain a classifier C
2 Capture 10×10 cross validation score.
3 Apply C to T and obtain class labels for each dependent pair.
4 Capture F1-score, precision and recall.
5 Repeat 3 & 4 for other target projects T

Cross project dependency extraction (CPDE):

CPDE also called the Transfer Learning model, is developed for a source project and used for predicting the dependencies for another project (target project). CDPE pseudo code details the logic of CDPE implementation. For WPDE, logic is identical to CDPE, but unlike CDPE, source S and target project T are one and the same.

Version-wise CPDE:

In the advent of data availability, it is practical to use the historical labeled dependency data of the prior versions within the same project to build classifiers and apply them to predict dependencies in the current version. However, such data might be limited or scarce, and CPDP could be effective in this scenario. Therefore, we supplement the dependency information from another project (source) until version-wise dependencies data for the target project is accumulated. We hypothesize that this will improve the performance of dependency prediction between successive versions.

Cross-version defect prediction [105] has been explored in the literature wherein the training data from different versions of a product are used to train a classifier and predict the defects for a newer or updated version of the product. Taking inspiration from it, we developed a solution in the first iteration of which we utilize labelled data (say, S) from a source project, build a classifier and predict for the oldest version (available), say, t_i of a target project T. In the next iteration, we include the labelled data from this older version t_i^l to S (i.e. now $S = S + t_i^l$) and once again build a classifier to test it for the next in line t_{i+1}, this process is repeated for all the version wise available data. This is our version-wise CPDE mechanism. It essentially simulates a scenario of CPDE for a target project

under development, and the data for training is available only over a software development life cycle. Pseudocode of Version-wise CPDE below details the logic of version-wise CPDE.

Version-wise CPDE pseudo code	
	Train set: S consisting of requirements pairs and their class labels where, label can be 0:*Independent*, 1:*Requires* and 2:*Others*. **Test set:** t_i consisting of requirements pairs for the oldest version i of the target project T. Although labels are known (as this is a completely labeled dataset) it is only used during performance evaluation.
1	Using labeled training set S, obtain a classifier C
2	Capture 10×10 cross validation score.
3	Apply C to t_i and obtain classified dataset t_i^l
4	Capture F1-score, precision and recall.
5	Move t_i^l to S
6	Increment i to i+1 (i.e. next version)
7	Go to 1 and repeat until all the versions i of T are classified

CPDE using fine-tuned BERT

It was essential to have a large train set to utilize BERT for classification. Hence, we utilized the Core project's data for this test as it had a larger dataset to perform this hypothesis. Fine-tuning BERT essentially means that domain-specific data is used to train the BERT models (off the shelf) and used as an extraction model for other projects.

5.3 Results

5.3.1 WPDE vs CPDE models for binary dependencies classification - RQ2.1

Using WPDE and CPDE algorithms explained previously, we developed three extraction models using NB, RF and SVM algorithms for the six selected projects with varying sizes (18 extraction models). We randomly selected 800 samples for training and 200 samples for testing for each one of the projects. Special care was taken for class balancing for both train and test sets. We then repeated the training and testing operations 10 times to finally capture the averaged results of F1, precision and recall.

As shown in Figure 5.3, WPDE models of Core and Firefox projects for all the three models performed well with F1-score>=0.8, however not so much for the other four projects. Firefox and Core are projects with larger datasets than others; hence we speculate it is the prime reason for this behaviour. A closer look at the precision and recall for all these models are as shown in Figure 5.4. Analysis shows that, in general, RF and NB excelled, and SVM did not yield good results comparatively.

For CPDE, we randomly selected 320 samples from each project and tested them with other project's extraction models. Heatmap of this empirical test is as shown in 5.5. While most projects could perform well when tested with Firefox data, others did not yield stellar results. The only interesting factor was that the Core and Firefox project

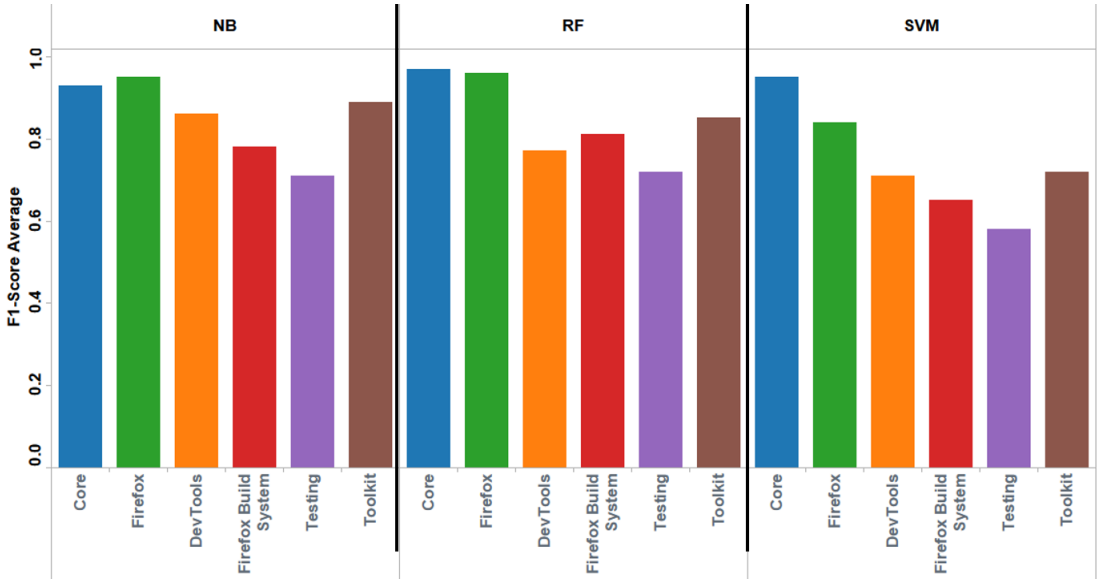


Figure 5.3: Average F1 score of WPDE for the six selected projects from Mozilla for the three conventional ML methods executed 10 times

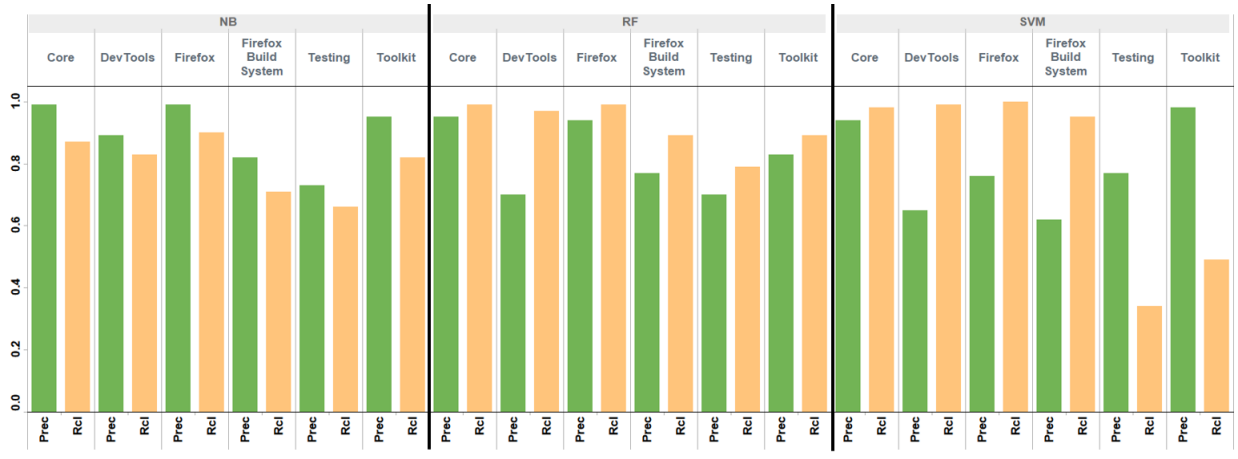


Figure 5.4: Average Precision and Recall score for WPDE for the six selected projects from Mozilla which were executed 10 times

could predict mutually and showed F1-score=0.6. Yet again we speculate the dataset size to be the main constraint which could have triggered such behavior.

Train Project \ Test Project	Core	DevTools	Firefox	Firefox Build System	Testing	Toolkit
Core		0.52	0.62	0.41	0.42	0.46
DevTools	0.5		0.56	0.54	0.44	0.57
Firefox	0.57	0.51		0.41	0.41	0.6
Firefox Build S..	0.46	0.44	0.55		0.45	0.45
Testing	0.51	0.48	0.51	0.44		0.54
Toolkit	0.53	0.55	0.58	0.45	0.47	

Figure 5.5: Average F1 score of 10 times execution for Binary class CPDE for six selected projects

RQ2.1: Within Project Dependency Extraction (WPDE) for all the six projects showed F1>=0.8 and Cross Project Dependency Extraction (CPDE) stood F1<=0.6. Clearly binary class WPDE performed better than CPDE for RDE.

5.3.2 WPDE vs CPDE models for multi-class dependencies classification - RQ2.2

In order to answer the second research question, we further dived deep into fine-granular dependency classification. For these empirical tests, we chose RF algorithm and three dependency types: *Requires*, *Other* and *Independent*.

Table 5.1: Statistical information of the six projects used for WPDE multi-class classification experiments

Project	**Train set**	**Test set**
Core	750 (= 250 × 3 classes)	150 (= 50 × 3 classes)
DevTools	630 (= 210 × 3 classes)	126 (= 42 × 3 classes)
Firefox	669 (= 223 × 3 classes)	135 (= 45 × 3 classes)
Firefox Build System	177 (= 59 × 3 classes)	36 (= 12 × 3 classes)
Testing	126 (= 42 × 3 classes)	27 (= 9 × 3 classes)
Toolkit	174 (= 58 × 3 classes)	36 (= 12 × 3 classes)

Table 5.1 shows the statistics for the data that was used for classification and testing. Due to the lack of data for the *Others* class, and use of undersampling technique introduced a major hurdle for multi class classification.

Figure 5.6 shows the results from average F1-score from 10 times execution of the WPDE models for the six projects. As shown, F1-score of the Independent class was the highest for Core and Firefox projects. Whereas, for the other two classes, results were not encouraging as the F1 score<0.6 in the most cases.

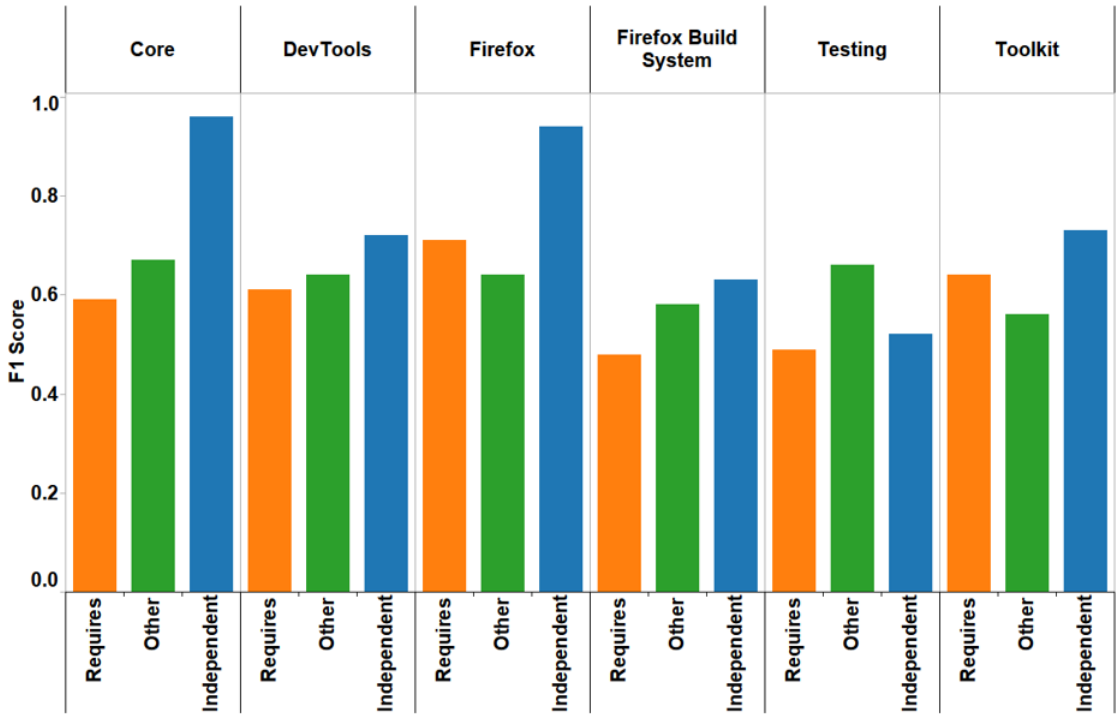


Figure 5.6: WPDE for multiclass class for six selected projects shows that F1-score for *Independent* class was the highest for Core and Firefox projects

For CPDE for multi-class classification, we utilized all of the data available for each project for training (source project). We then tested with all of the data available for each target project. For example, from Table 5.1, we used 900 samples of Core for training and tested it against 630 samples of DevTools, 669 of Firefox etc. Results of these extraction models using NB and RF are as shown in Figure 5.7a and 5.7b respectively. Results showed that CPDE for multi-class classification was not useful. None of the F1 scores could cross beyond the 0.4 mark barring a couple of exceptions as highlighted through heat maps.

RQ2.2: Within Project Dependency Extraction (WPDE) for all the six projects showed F1<=0.6, and Cross Project Dependency Extraction (CPDE) stood F1<=0.4. Clearly, multi-class WPDE performed better than CPDE for RDE. However, overall results for both did not yield encouraging outlook

5.3.3 CPDE models for version wise dependencies classification - RQ2.3

Since the CPDE from multi-class classification was pessimistic, we reduced the scope of the empirical analysis to binary classification. However, we chose *Requires* and *Independent* as the two classes and explored CPDE in the context of version-wise data. In this context, we first chunked the target project's (Bugzilla) data based on versions

Train Project \ Test Project	Core	DevTools	Firefox	Firefox Build System	Testing	Toolkit
Core		0.36	0.42	0.35	0.35	0.36
DevTools	0.32		0.42	0.39	0.29	0.41
Firefox	0.48	0.4		0.32	0.3	0.41
Firefox Build System	0.33	0.32	0.34		0.37	0.29
Testing	0.36	0.33	0.28	0.45		0.28
Toolkit	0.37	0.34	0.38	0.31	0.41	

(a) CPDE for multiclass using Naive Bayes shows that overall results were pessimistic

Train Project \ Test Project	Core	DevTools	Firefox	Firefox Build System	Testing	Toolkit
Core		0.25	0.33	0.3	0.28	0.22
DevTools	0.25		0.33	0.26	0.21	0.27
Firefox	0.37	0.35		0.21	0.3	0.32
Firefox Build System	0.27	0.28	0.22		0.39	0.22
Testing	0.28	0.2	0.23	0.32		0.27
Toolkit	0.24	0.25	0.28	0.27	0.33	

(b) CPDE for for multiclass using Random forest

Figure 5.7: CPDE output for multi-class classification

and then utilized data from the source project (Firefox) to evaluate CPDE. We repeated the CPDE and, in each instance, added the version data of the target project to provide the target project's knowledge incrementally. This imitated the scenario where CPDE is used for a project with no training data, and incrementally version-wise data is gradually used for training as and when the data becomes available as software versions are released.

For this experiment, we used Bugzilla project as target project and Firefox as source project. version wise data distribution for Bugzilla is as shown in Figure 5.8. The data collection stage was challenging, and we had limited data

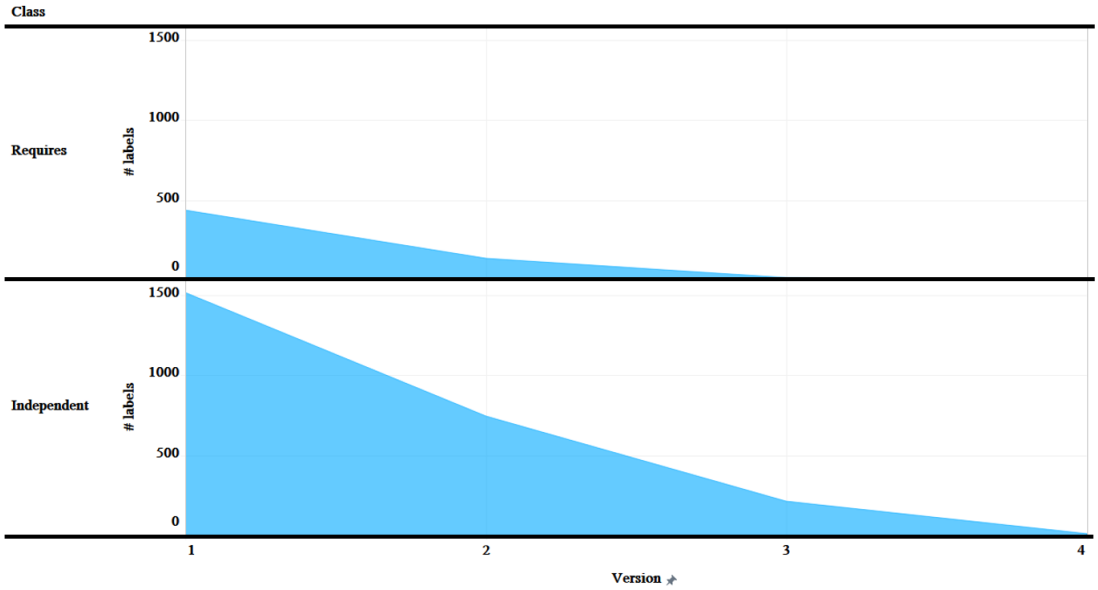


Figure 5.8: Version wise dataset of the Bugzilla project shows data availability for the earlier version is comprehensive compared to others

to perform these tasks. For versions 1 and 2, data was 500 and 138, and for others, it was just a very few samples.

We used 1500 samples of each class from the Firefox project for training and tested V1 (1st version) of Bugzilla. In the second iteration, we added V1 to Firefox data and trained the model to predict for V2 and so on. Results of

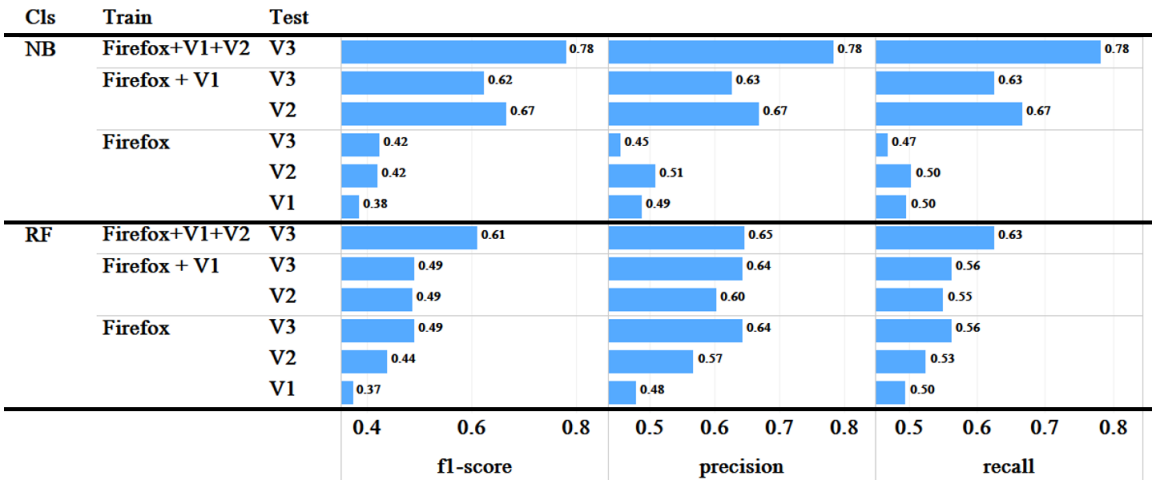


Figure 5.9: version wise incremental CPDE for Bugzilla project

this experiment are as shown in Figure 5.9 for the Naive Bayes and Random forest classifiers.

There was no clear winner among the two ML methods. RF showed better accuracy and recall when Firefox only data was used for training compared to NB. Also, when training data was incrementally supplemented utilizing Bugzilla's data, RF showed the highest recall=0.65. In comparison, NB could achieve the highest F1=0.8 in the last iteration of the CPDE testing.

Results showed an exciting and encouraging trend for this experiment. However, we speculate that these results can not be generalized as the version-wise data was inadequate.

RQ2.3: Using Firefox's data, version-wise CPDE for the Bugzilla project showed encouraging results for RF and NB. However, this needs to be tested with larger datasets to arrive at concrete outcomes.

5.3.4 CPDE using fine-tuned BERT - RQ2.4

Firstly, to keep the results comparable, we developed WPDE models using NB for the four target projects: Firefox, DevTools, Testing and Toolkit. Results of these models are listed in Table 5.2

We utilized 2900 samples of each class: *Requires* and *Independent* belonging to the Core project, the source project, for fine-tuning the BERT model. The train and test loss are shown in Figure 5.10 over the 4 epochs. The results showed that up to 3 epochs of training would suffice as the loss is negligible over iterations there off.

Thus trained model was then used to predict dependencies for other four target projects. The results are as shown in Table 5.3.

WPDE results from Table 5.2 show that F1>0.7. However, NB's CPDE model using Core project data did not match this performance for any of the projects and F1<0.6 for Firefox, Testing and Toolkit projects. For DevTools the performance was F1=0.38. We also tested RF CPDE model and the results showed marginal improvements only and did not match the WPDE F1.

Table 5.2: WPDE using Naive Bayes

	Data points	Precision	Recall	F1	Accuracy	Confusion Matrix
Firefox	0: 1116 1: 1116 Train: 893 each Test: 224 each	0.76 0.66	0.58 0.82	0.66 0.73	0.70	[[120 94] [40 184]]
Dev Tools	0: 783 1: 783 Train:626 each Test:127 each	0.82 0.77	0.75 0.83	0.78 0.80	0.79	[[117 40] [26 131]]
Testing	0: 206 1: 206 Train: 165 each Test: 42 each	0.73 0.85	0.88 0.69	0.80 0.76	0.78	[[36 5] [13 29]]
Toolkit	0: 423 1: 423 Train:338 each Test: 85 each	0.76 0.72	0.69 0.78	0.72 0.75	0.74	[[59 26] [19 66]]

Figure 5.10: Train and Test loss curves for BERT when fine-tuned using Core project's data

Table 5.3: Accuracy, F1 and other measures using CPDE model of NB, RF and fine-tuned BERT on Core project's data for other target projects. Class 0 is *Independent* and Class 1 is *Requires*

Target project	**Test size**	**Algorithm**	**Precision**	**Recall**	**F1**	**Accuracy**	**Confusion Matrix**
Firefox	0: 1116 1: 1116	Naive Bayes	0.55 **0.89**	0.98 0.20	0.70 0.33	0.59	[[1089 27] [888 228]]
		Random Forest	0.63 **0.94**	**0.97** 0.42	**0.76** 0.58	**0.70**	[[1086 30] [643 473]]
		Fine tuned BERT	**0.74** **1.00**	**1.00** **0.67**	**0.86** **0.80**	**0.83**	[[1114 2] [364 752]]
Dev Tools	0: 783 1: 783	Naive Bayes	0.41 0.33	0.53 0.23	0.46 0.27	0.38	[[412 371] [603 180]]
		Random Forest	0.53 0.57	**0.75** 0.33	0.62 0.42	0.54	[[588 195] [521 262]]
		Fine tuned BERT	0.62 **0.68**	**0.75** 0.54	**0.68** 0.61	0.65	[[584 199] [357 126]]
Testing	0: 206 1: 206	Naive Bayes	0.54 0.65	**0.84** 0.29	**0.66** 0.40	0.57	[[174 32] [147 59]]
		Random Forest	0.60 **0.67**	**0.75** 0.50	**0.67** 0.57	0.63	[[154 52] [102 104]]
		Fine tuned BERT	**0.74** **0.77**	**0.79** **0.72**	**0.76** **0.75**	**0.76**	[[163 43] [58 148]]
Toolkit	0: 423 1: 423	Naive Bayes	0.52 0.56	**0.78** 0.28	0.62 0.37	0.53	[[331 92] [306 117]]
		Random Forest	0.51 0.51	0.49 0.53	0.50 0.52	0.51	[[206 217] [199 224]]
		Fine tuned BERT	**0.72** **0.82**	**0.84** **0.71**	**0.79** **0.76**	**0.78**	[[357 66] [123 300]]

On the other side, fine-tuned BERT CPDE models showed encouraging results and outperformed Firefox, Toolkit and Testing's NB and RF CPDE F1 scores. It was evident that this deep learning-based technique could be used for transfer learning for smaller target projects (Testing and Toolkit target projects are the smallest of the 6 selected projects). Due to this positive outlook, we further analyzed the applicability of fine-tuned BERT models for RDE, which is explained in the next chapter in detail.

RQ2.4: Since Transfer Learning is also enabled through fine-tuning the pre-trained BERT models on the task-specific dataset, we explored fine-tuned BERT for RDE. Results showed encouraging results as the CPDE using fine-tuned BERT outperformed within project's conventional models by 27% to 50% (on F1-score measure scale).

5.4 Discussion

We analyzed how cross-project dependency could benefit RDE in the three scenarios. Although results from binary class extraction showed encouraging results, multi-class classification struggled due to lack of data, undiscovered advanced sampling techniques, feature extraction and ML techniques. We speculate that identifying these techniques could benefit operationalizing TL in RDE to a great extent. Hence, we explored BERT, a state-of-the-art DL method that also overcomes the need for feature extraction in NLP-based classification and extraction tasks as a next step.

As anticipated, fine-tuned BERT showed encouraging results in the preliminary empirical evaluations (RQ2.4). Thus, the next chapter focuses on exploring BERT in RDE and further analyzing various empirical experiments utilizing Mozilla family of products datasets.

Summary

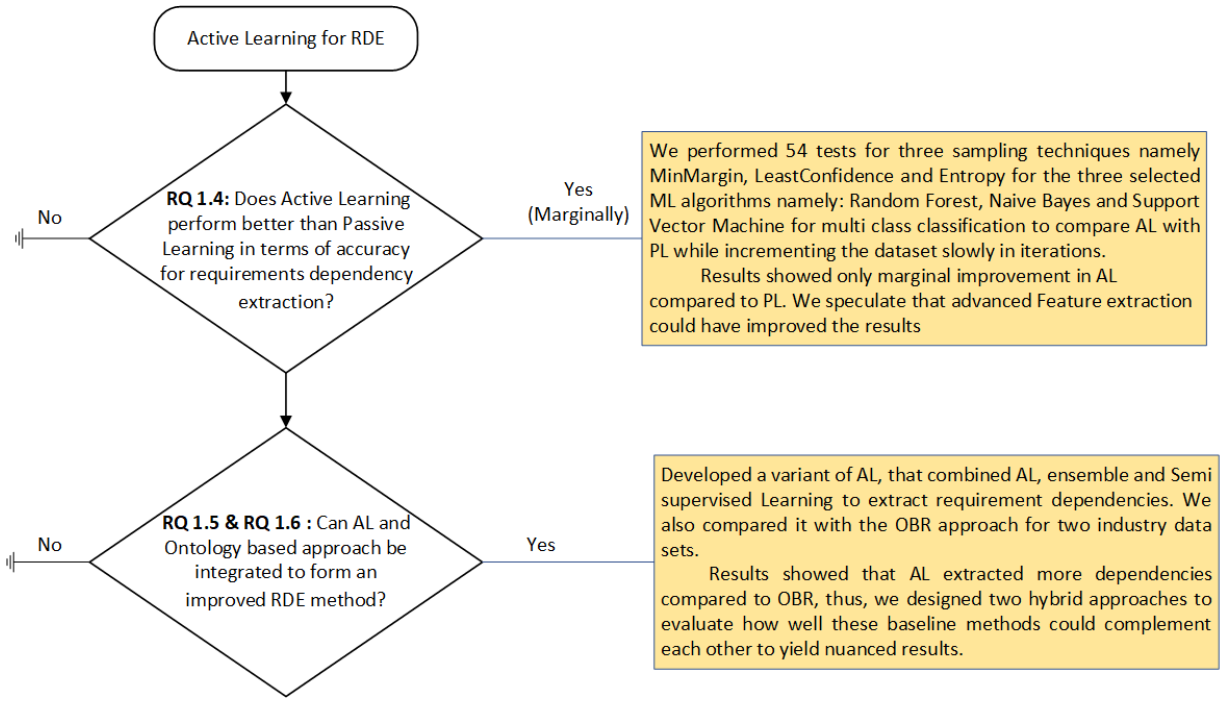


Figure 5.11: Bird eye view of the research questions, their logical connection and the briefly explained results

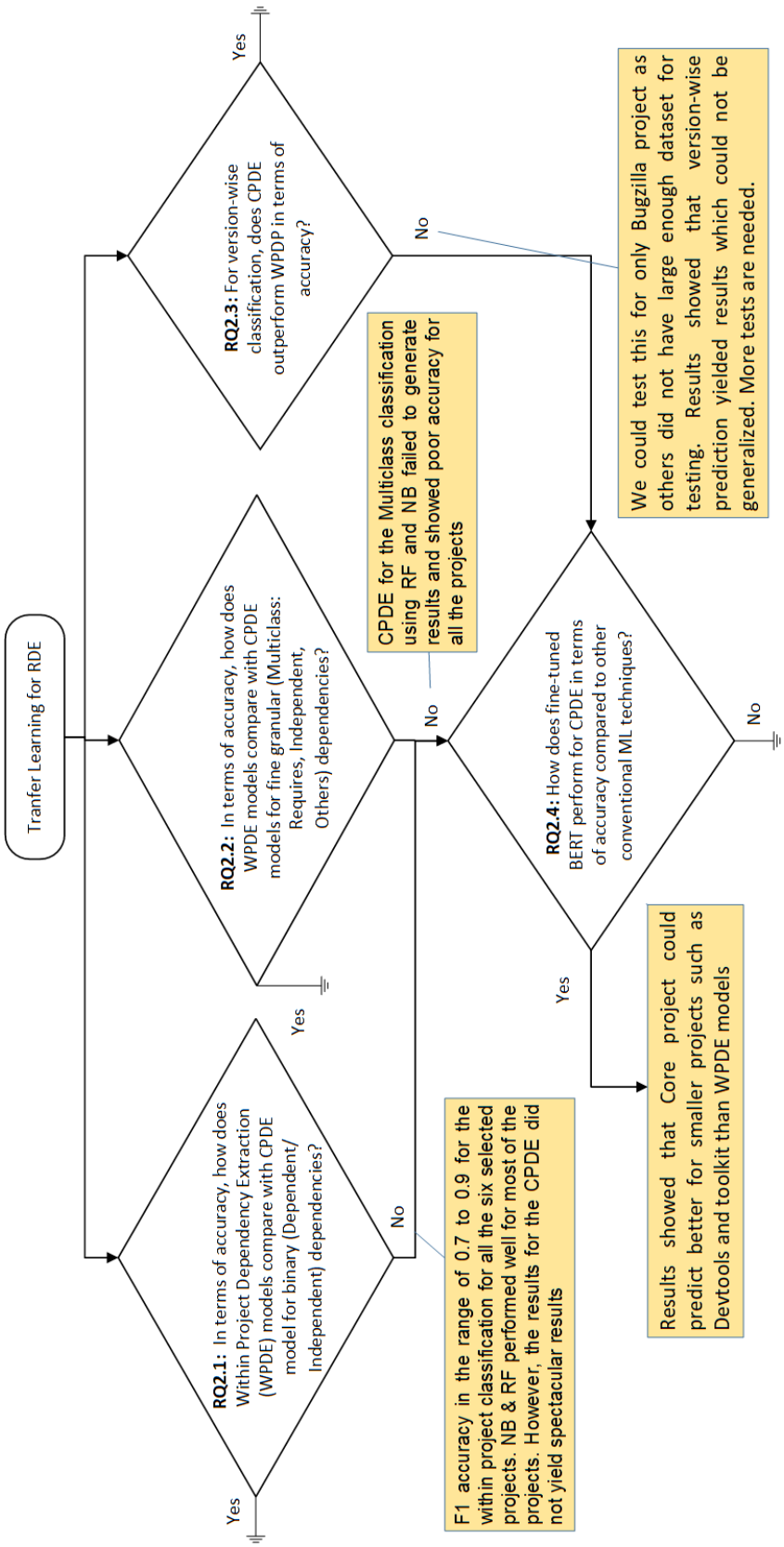


Figure 5.12: Bird eye view of the research questions, their logical connection and the briefly explained results

Part III

Addressing Feature Extraction Challenge (RQ3)

Chapter 6

Fine-tuned BERT for RDE

6.1 Introduction

Fine-tuning BERT is an ML step wherein off-the-shelf BERT (pre-trained BERT model) is further trained on task-specific data. This task is RDE in our context; hence we term the outcome of fine-tuning the BERT model using RDE data as RDE-BERT. In order to determine the effectiveness of RDE-BERT, we performed mainly two empirical studies. 1) Evaluation of RDE-BERT in terms of accuracy against other conventional ML methods such as Naive Bayes and Random Forest. 2) Evaluate if RDE-BERT can detect the dependency direction.

6.2 Related Work

6.2.1 Requirements Dependency Classification

In the recent past, many empirical studies have explored diverse computational methods that used Natural Language Processing (NLP) [124] [139], WSL technique [51], hybrid techniques [52] and DL [78] to analyze requirement dependencies.

Deshpande [52] proposed a new method for extracting requirement dependencies. This method integrated active learning with ontology-based retrieval to extract requires, refine 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.

In the advent of recent advancements in NLP and ML, RDE automation has garnered serious interest. Recently, Atas et al. [19] used POS-tag (Parts-Of-Speach Tagging) and n-grams for feature extraction in the textual requirements of an industry dataset (annotated by students) before using supervised ML methods for classification. However, Atas

et al. highlight the small size of the dataset and the need for domain experts in annotations as serious threats to the validity of the results.

6.2.2 Applications of BERT in Requirements Engineering

Araujo et al. [47] explored the application of BERT to automate the identification of software requirements from app reviews. In a similar vein, [15] 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 more significant advantage overall.

Hey, et al. [84] Used a fine-tuned BERT model to classify requirements into functional and Non-functional and showed that it improves requirements classification and can be used for unseen projects with convincing results. Sainiani et al. [138] Utilized the BERT model to extract and classify requirements from large software engineering contracts into predefined classes. Their results showed that a higher F-score could be achieved for classifying requirements with BERT.

Das et al. analyzed [46] 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. [11] 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 dataset's structure.

Fishbach et al. [73] 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. [98] 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 has been 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? It needs to be studied further.

6.3 Research Method

Since we are utilizing Bugzilla data, discussed in Section 3.4, we elaborate on the research questions in this section. This section evaluates how efficient BERT is in tackling RD specific problems. Thus, we evaluate two research questions as follows.

6.3.1 Research Questions

RQ3.1 How does RDE-BERT compare with Naive Bayes and Random Forest in terms of accuracy?

Why: Existing methods utilized for dependency type prediction suffer from a lack of annotated data and

the need for advanced NLP feature engineering techniques, which makes it hard to draw firm conclusions. Bugzilla data provides a massive set of annotated data for over 69 projects which consist of dependency type and requirement pairs. Fine-tuned BERT models have been successful at addressing sentence pairs related problems.

How: By fine-tuning the BERT model for the randomly selected subset from Bugzilla, we can verify and compare its effectiveness with the BOW method utilized in our previous research so far and test if dependency types for a dataset belonging to a particular project could be predicted.

RQ3.2 In terms of accuracy, how does RDE-BERT compare with others to predict the direction of the dependency?

Why: Modeling requirement pairs has been a core NLP research problem under active study. Additionally, for a long time, dependency direction prediction has been elusive due to the nature of problems exhibited in the natural language used while documenting requirements. Dependency types such as *Requires* which are a ordered pair of requirements exhibit direction [139]. Automating such direction extraction is a challenge as ML methods need to understand the context underlying the content.

How: BERT has been evaluated for its efficiency in modeling a pair of sentences and capturing the relationship between them. This is also referred to as "Next sentence prediction" NLP task [59]. Thus, we speculate that utilizing the NextSentencePrediction BERT model and further fine-tuning it for Bugzilla data could effectively tackle this challenge.

6.3.2 Data and Experiment Setup

The Firefox project dataset was considered for evaluation. Care was taken to analyze the sentence length and utilize sentences with over 5 words only. For RQ3.1, experiments focused on binary classification and *Requires* and *Independent* dependency classes were considered. We sampled 4,590 data points which were balanced for classes. Then, the complete dataset was split into 80:20 train and test ratios for all the tests. Then ML models were developed starting with 10% of the train set, and various performance measures such as F1 score, precision and recall were captured. The process was repeated, and the train set was incremented by 5% in every iteration until all the train set data was exhausted. In every iteration, performance measures were captured and documented.

Since RQ3.2 focused on dependency direction evaluation, we considered *Requires* dependency type and inverted these dependencies to generate *Not dependent* dependencies. We then randomly sampled *Requires* and non-dependent pairs to create a dataset. Irrespective of the project, we randomly chose 7,722 *Requires* dependency and *Independent* pairs after shuffling all the data that belonged to 69 Mozilla projects. For training and testing, split the data into an 80:20 ratio.

Experiments were repeated 10 times before documenting the average performance measures. Also, the confusion matrix for the last experiment is documented.

6.4 Results

6.4.1 RDE-BERT Vs Others - RQ3.1

Figure 6.1 shows the plot of the F1 score captured at various instances while gradually increasing the training set size over iterations. As shown in the results, Hyperparameter tuning of RF and NB did not yield drastically nuanced results as they stood at F1=0.76 and F1=0.68, respectively. Whereas RDE-BERT struggled initially, increasing the train set showed stellar results at F1>0.87. However, all the methods hit a plateau beyond 70% train set and with 80% train set final F1=0.85, 0.75 and 0.67 for RDE-BERT, hyperparameter tuned RF and NB respectively. Scrutiny

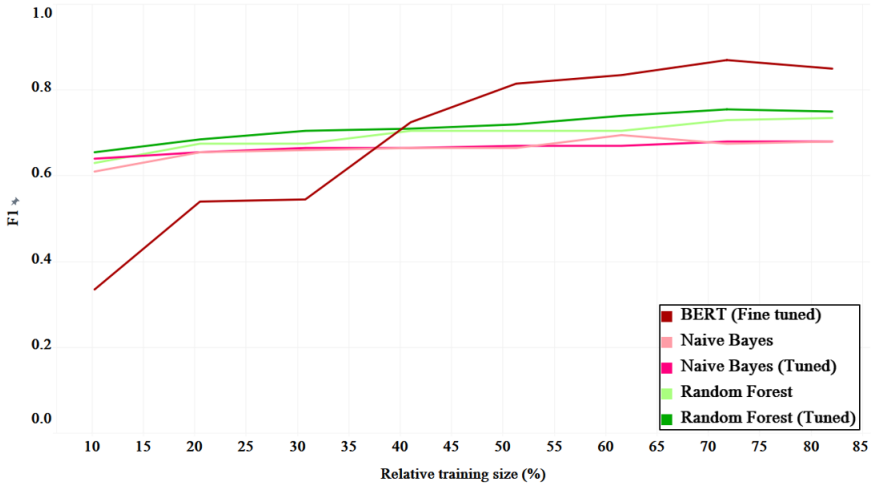


Figure 6.1: Comparison of BERT, Naive Bayes and Random Forest methods on F1-score for RDE: Results show that BERT outperforms others by 13% to 27%

of precision and recall helped to shed light on acceptable behaviour as shown in Figure 6.2. This chart is for *Requires* dependency, which is the focus of our interest.

For the RDE problem, False Negative is more detrimental than False Positive because rework or effort and time investment when dependencies are missed are expensive than ignoring/validating the falsely identified independent ones as a dependent (false positives). Thus better recall is important for RDE. Following are various observations

- Precision of all the ML methods was comparatively better than the recall when the train set was up to 70%.
- Hyperparameter tuned RF could achieve max recall=0.7 at precision=0.76 with 60% of the train set.
- Hyperparameter tuned NB could achieve max recall=0.71 at precision=0.65 with 40% of the train set.
- RDE-BERT could achieve max recall=0.82 at precision=0.91 with 70% of the train set.
- However, with just a 30% train set, RDE-BERT achieved the highest recall=0.85 and precision=0.55, indicating a very high number of false positives.

These observations show a trade-off between the F1 score and the train set size. A more extensive train set assures higher recall and precision, but how much investment does it incur to procure such massive data set? How

feasible is it in the real-world where unlabelled data is abundant and labeled scarce? These questions need scrutiny of various ML methods' cost and benefit analysis.

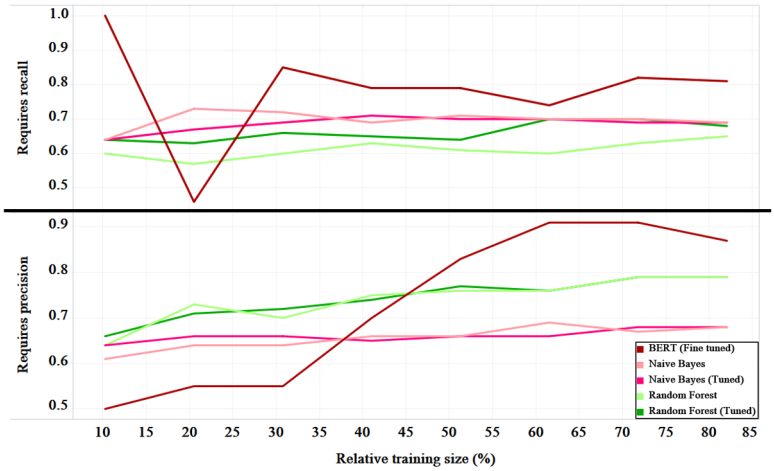


Figure 6.2: Precision and Recall comparison for *Requires* dependency type when various ML methods are used for RDE.

6.4.2 Dependency Direction Identification - RQ3.2

In order to evaluate this research question, we once again performed experiments focusing on binary classification where the two classes were *Requires* and *Independent*. As an experiment, we added a few of the *Requires* dependency samples to *Independent* data pool by labeling them independent while inverting the pair order [59]. For example, $\{R1, R2, Requires\}$ was converted to $\{R2, R1, Independent\}$. This was to test if ML methods would learn the direction information in the ordered pair of requirements.

Table 6.1 shows the results for five chosen ML methods and their respective confusion matrix. We tested hyperparameter tuned RF and NB; however, the results were not interesting. Thus we also evaluated hyperparameter tuned Logistic regression, which slightly improved the results yet for good. Whereas RDE-BERT showed exceptional results and performance with F1 and precision, recall at 0.76 compared to other ML methods as highlighted in the table.

Fine-tuned BERT outperformed conventional ML methods by 13% to 27% on the F1-score scale for Firefox project dataset. Also, we showed that fine-tuned BERT successfully predicted the dependency direction and outperformed conventional ML methods by 90%.

6.5 Discussion

In this empirical study, we evaluated the applicability of Fine-tuned BERT for RDE specific two tasks. Results showed that RDE-BERT excelled and outperformed others in performance measures such as F1, precision and recall. Despite the stellar performance of RDE-BERT, it is crucial to notice that it is computationally expensive and needs

Table 6.1: Results for directionality test: trained on 5405 of each type: Independent and Requires

	Dependency	**Precision**	**Recall**	**F1**	**Accuracy**	**Confusion matrix Test set: 2317 balanced**
Naive Bayes	Requires	0.27	0.26	0.27	0.27	[[645 1672]
	Independent	0.27	0.28	0.28		[1706 611]]
Random Forest	Requires	0.15	0.15	0.15	0.16	[[373 1944]
	Independent	0.16	0.16	0.16		[1964 353]]
Logistic Regression	Requires	0.27	0.27	0.27	0.27	[[628 1689]
	Independent	0.27	0.26	0.26		[1696 621]]
LR+ Hyper parameter tuning (C=0.0001)	Requires	0.36	0.35	0.35	0.37	[[900 1417]
	Independent	0.37	0.39	0.38		[1517 800]]
Fine tuned BERT	Requires	**0.75**	**0.76**	**0.76**	**0.76**	[[1752 565]
	Independent	**0.75**	**0.76**	**0.76**		[548 1769]

an extensive training set. Gathering such a train set is a cost and effort-intensive task in the real world. Thus, it is essential to critically evaluate various ML methods based on performance measures and the cost and benefits it would rake in the ML process. Hence, we evaluated if BERT is a new silver bullet for RDE and presented its outcome in the next chapter.

Chapter 7

Is BERT the New Silver Bullet?

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 Extraction (RDE). In general, based on various criteria used for evaluation, decisions on classification method preference may vary. For RDE, we go beyond conventional metrics such as the F1-score and consider Return-on-Investment (ROI) to evaluate two techniques for such decision making. We study RDE-BERT (fine-tuned BERT) using data specific to requirements dependency extraction) and compare with Random Forest, our baseline. For RDE and data from Free OSS 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.

7.1 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 [179] 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 [34], 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 [11], [73], [46], [84]. 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 [59]. This task is Requirements Dependency Extraction (RDE) for our study.

The fine-tuned BERT model using RDE-specific data from the Redmine FOSS project [5] is referred to as *RDE-BERT* in our study. We compare RDE-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 [135]. 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, 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 [111] objectively. Since for decision-makers, it is vital to have an affinity to analytics rather than programming and modeling [166], 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 [56]. 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 study makes the following contributions

- Compare and evaluate ML technique: Random Forest (baseline) and RDE-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, related work is elaborated in Section 7.2. We describe basic concepts and the research questions addressed in this paper in section 7.3. This is followed by the approach to answer the proposed research questions in section 7.4. Section 7.5 provides more information on the dataset used, while Section 7.6 elaborates our ROI model. Results are then discussed in Section 7.7. Section 7.8 discusses the validity of results and we conclude the paper with a discussion in Section 7.9.

[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 RDE-BERT.

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

7.2 Related Work

7.2.1 ROI-based Decision-making in Software Engineering

Boehm et al. [33] [32] 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. [89] 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. [99] 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. [71] 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.

Weiss et al. [163] answers the question regarding what quality of external data one must aim for, when such data is available at a premium and translated the total cost in term of CPU time and treatment of the subject is limited to a static setting. One of the main goals of this paper is to analyze the results achieved from ROI analysis focusing on extraction of requirement dependencies by Deshpande and Ruhe [56] and compare if the results from Firefox project where, learning can be stopped around 20 for RF and around 40 for BERT technique would be similar to the projects Redmine and Ruby.

Ruhe and Nayebi [3] proposed the Analytics Design Sheet as a means to sketch the skeleton of data analytics process. The four quadrants would help to understand the data analytics methods, techniques and available data related to a problem statement in a better and simpler way. Nagrecha et al. [119] 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.

7.3 Concepts & Research Questions

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

7.3.1 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 [6] 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*.

7.3.2 Research Questions

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

RQ3.3: In terms of accuracy and varying training set size, how does RDE-BERT compare with Random Forest?

RQ3.4: How does the ROI of RDE-BERT compare with the ROI of Random Forest?

RQ3.5: How sensitive are the results of RQ3.4 for varying cost estimates?

7.3.3 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 7.1 shows the principal entries of CM for a binary class classification.

Table 7.1: 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 (7.1).

$$F1 = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{7.1}$$

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 (7.2). Both $Benefit$ and $Cost$ are measured as human effort in person-hours.

$$ROI = (Benefit - Cost)/Cost \tag{7.2}$$

[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

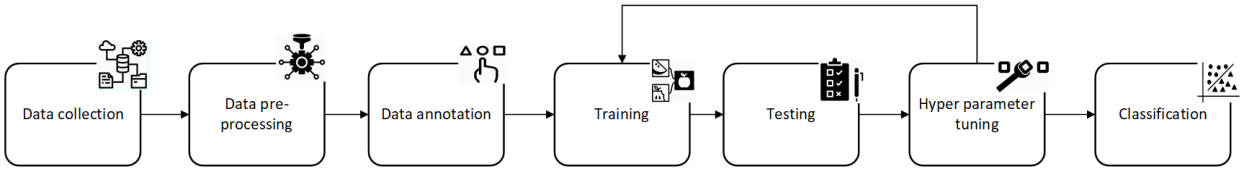


Figure 7.1: Various steps that formulate the Machine Learning process (images curtsy NounProject.com)

In the crux, our study evaluates various conditions under which RDE-BERT could be preferred over the baseline ML (i.e. RF). Specifically, overriding the exclusive accuracy considerations, we compare F1 and ROI from RF and RDE-BERT and analyze different results depending on the preference criterion such as train set and varying cost factors.

7.4 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 RDE-BERT with the RF on varying values from the confusion matrix (7.3.3). In this section, the methodology of this study for CM values of varying sizes of the training set is presented.

7.4.1 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 7.1. 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 [59].

7.4.2 Research Design

Figure 7.2 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 7.2, 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 RDE-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 (RDE-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 7.2). Later in step ❽, we also compute the ROI for RDE-BERT and compare it with the ROI of RF. Finally, sensitivity analysis is performed to understand how factors impact the ROI computation (❾).

Table 7.2: Confusion matrix terminology specified for RDE

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)

7.4.3 RDE-BERT

We use a pre-trained BERT model in combination with our RDE-specific dataset. The result is a fine-tuning BERT model called *RDE-BERT*. We use BertForSequenceClassification from the huggingface PyTorch library [87] for this implementation.

In every instance, for a given training set size, RDE-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, RDE-BERT was used to classify the test set and the resulting F1-score and confusion matrix were captured.

7.4.4 Baseline: Random Forest

While for RDE-BERT, the data were retained in their original form, it was further passed through the NLP pipeline (explained in detail in section 7.5. B) [162] for stopword removal, lemmatization, and TF-IDF vectorization [56], before feeding it to RF for classification. The same train and test split dataset were utilized in both RDE-BERT and RF to retain data authenticity.

For a given training set, hyper-parameter tuning was performed using random search [24]. 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.

7.4.5 Evaluation Setup

Classification: For RDE, we denote requirement pairs having *Relates_to* dependencies as *positive classes* and *Independent* pairs as *negative classes*. Dataset was balanced using under-sampling technique [150].

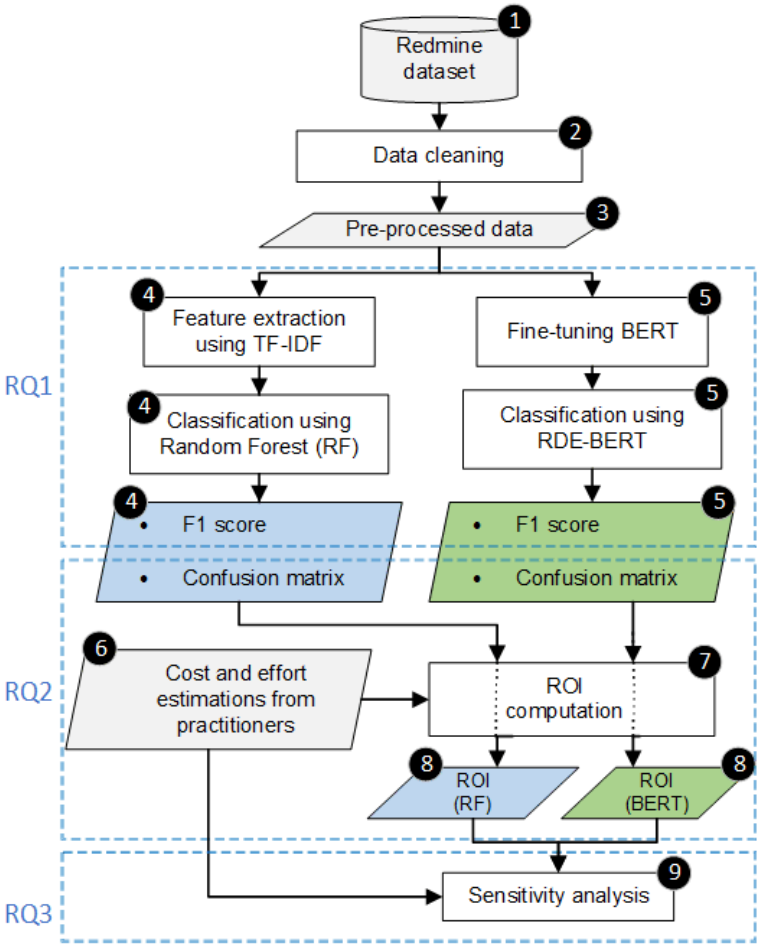


Figure 7.2: Research design of the study show 9 different steps to evaluate three RQs using Redmine dataset

Iteration: First, for both RF and RDE-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.

7.5 Data

7.5.1 Data Collection

Redmine [5] 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.

7.5.2 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.

Table 7.3: 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

For this study, we exclusively analyzed the *Relates_to* dependency since it is the most frequently occurring dependency in this dataset. Table 7.3 shows sample pairs of this dependency type. We used the 790 requirements that had empty "links" field (meaning no dependencies) to generate *Independent* pairs. A pair of id was randomly picked from this pool to generate 10,000 *Independent* pairs.

7.5.3 Data Pre-processing

We do not need to perform pre-processing for RDE-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 [129].

7.6 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 RDE 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.

Table 7.4: 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	$

7.6.1 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).

7.6.2 Return: The Value Factor

In the context RDE 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

[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

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 7.4 lists the relevant cost components and their corresponding units.

7.6.3 Cost and Benefit Estimation

As explained in Section 7.4.5, 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 7.4. $Cost$ 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 7.3.

$$Cost = n * \frac{(C_{dg} + C_{pp} + C_{ft} + C_e + C_l)}{60} * N_H * C_{res} \tag{7.3}$$

Return computations for RDE, assumes reward ($Cost_{FP}$) for misidentifying independent requirements (FP) and heavily penalizing ($Cost_{FN}$) instances that were falsely classified as independent (FN). Equations (7.4) and (7.5) show these computations.

$$TotalPenalty = FP * Cost_{FP} + FN * Cost_{FN} \tag{7.4}$$

$$Return = Value_{product} - TotalPenalty \tag{7.5}$$

7.6.4 ROI Sensitivity Analysis

We perform sensitivity analysis to investigate the effects of various principal parameters on the ROI computation. Harman et al. [83] 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 7.5 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.

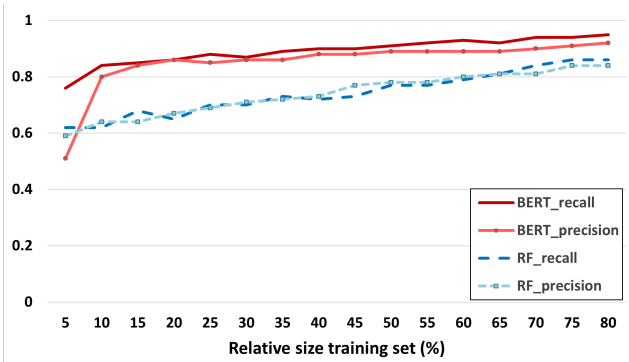


Figure 7.3: Precision and recall of RDE-BERT and RF shows that the precision and recall for BERT were low in the beginning, thy pick up dramatically with increasing train set size

7.7 Results

7.7.1 Comparison Based on Accuracy - RQ3.3

Figure 7.3 shows the precision and recall curves for the two methods. Although precision and recall of the RDE-BERT are low at the beginning, they pick up dramatically after initial slow growth and outperform RF. As shown in Figure 7.4, over incremental training set size, F1 of RDE-BERT outperformed the RF by a margin of 10%. The highest F1-score that RDE-BERT could achieve was 0.93 whereas the RF peaked at 0.83. Right from the beginning, the F1-score of RDE-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.

RDE-BERT outperforms RF in terms of F1 right from the beginning. RDE-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.

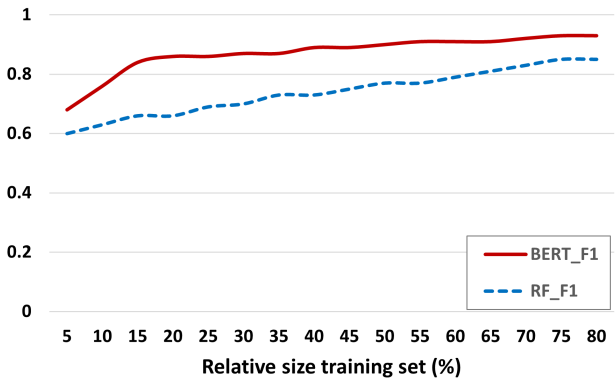


Figure 7.4: F1 of RDE-BERT vs RF for varying training set sizes show that RDC-BERT outperforms RF on F1-score measure

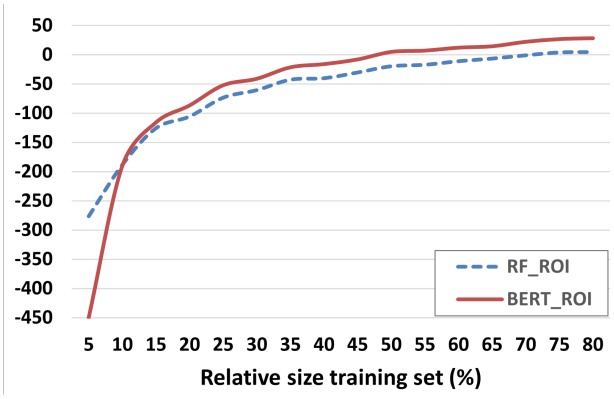


Figure 7.5: ROI of RDE-BERT vs RF for varying training set sizes show that with over 45% train set RDC-BERT starts to generate positive ROI

7.7.2 Comparison Based on ROI - RQ3.4

To compare RDE-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 [10], an OSS application, through an interview. Table 7.5 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 7.5: 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 7.5 shows a comparison of the ROI between RF and RDE-BERT. RF performs poorly and achieves positive ROI only with 75% or more training set size. However, RDE-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 RDE-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 and RDE-BERT does not make a huge difference. In hindsight, choosing RF could be more economical as it is computationally inexpensive comparatively.

RDE-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 RDE-BERT perform about the same and generate negative ROI.

7.7.3 Sensitivity Analysis of Fine-tuned BERT - RQ3.5

In this question, we were interested how much the results will change for changing (cost) values. We computed the ROI for both RDE-BERT and RF for varying $Cost_{FN}$ and $Cost_{FP}$ around the values defined in Table 7.5. In particular, we studied two scenarios. We visualized the results as a heat-map. If the difference ROI(RDE-BERT)

- ROI(RF) is positive then this means that RDE-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 RDE-BERT utilized 5% of the dataset for training. We kept the $Value_{product}$ static at 4M. As shown in Figure 7.6, the RF was resilient to higher $Cost_{FN}$ values comparatively (yellow hue).

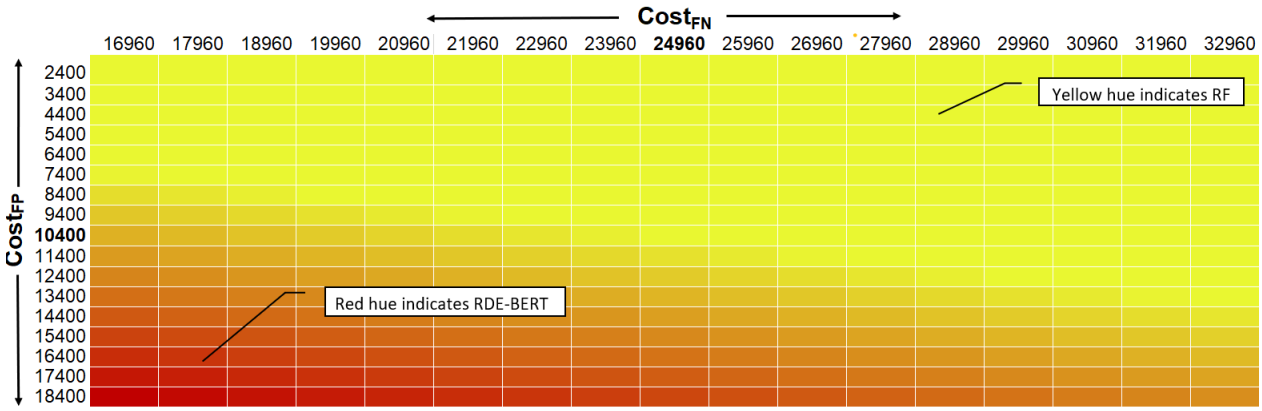


Figure 7.6: Difference in ROI values of RDE-BERT and RF (RDE-BERT minus RF) for varying $Cost_{FN}$ and $Cost_{FP}$. The comparison is for 5% training set size and $Value_{product}$ = 4M. The yellow hue in heat-map (upper right triangle) shows that RF performs better than RDE-BERT for higher values of $Cost_{FN}$ in conjunction with lower values of $Cost_{FP}$

Secondly, we fixed the $Value_{product}$ to 3M and re-evaluated the ROI at 10% of the dataset like before. As shown in the 7.7, results show that RF was also very effective for higher values of $Cost_{FN}$ cost factors compared to RDE-BERT which held strong for $Cost_{FP}$ mostly.

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.

In summary, as shown in Table 7.6, RDE-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 7.6: 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		🏆

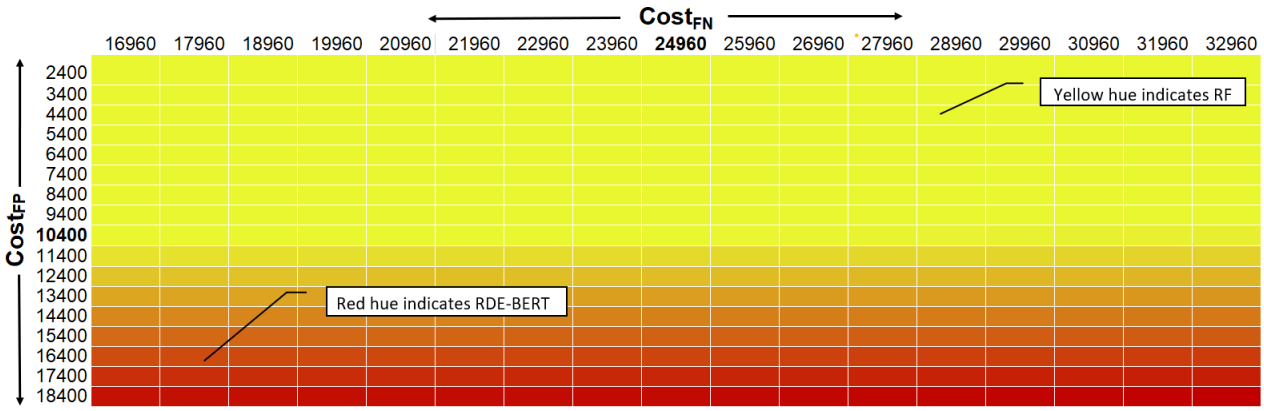


Figure 7.7: Difference in ROI between RDE-BERT and RF for varying $Cost_{FN}$ and $Cost_{FP}$. The comparison is for 10% training set size and $Value_{product}$ = 3M. The yellow hue in heatmap (upper part of rectangle) shows that RF performed better for growing $Cost_{FN}$ and not so much for $Cost_{FP}$ compared to RDE-BERT when the dataset is smaller.

7.8 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.

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 [56] 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.

7.9 Discussion

In this study, we used a fine-tuned variant of the breakthrough BERT technology, called RDE-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. [28]. 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 study is the first step in this direction.

Summary

Figure 7.8 provides pictorial gist of this study. Utilizing flow diagram, the interconnection between various research questions explored in this study are explained in this Figure.

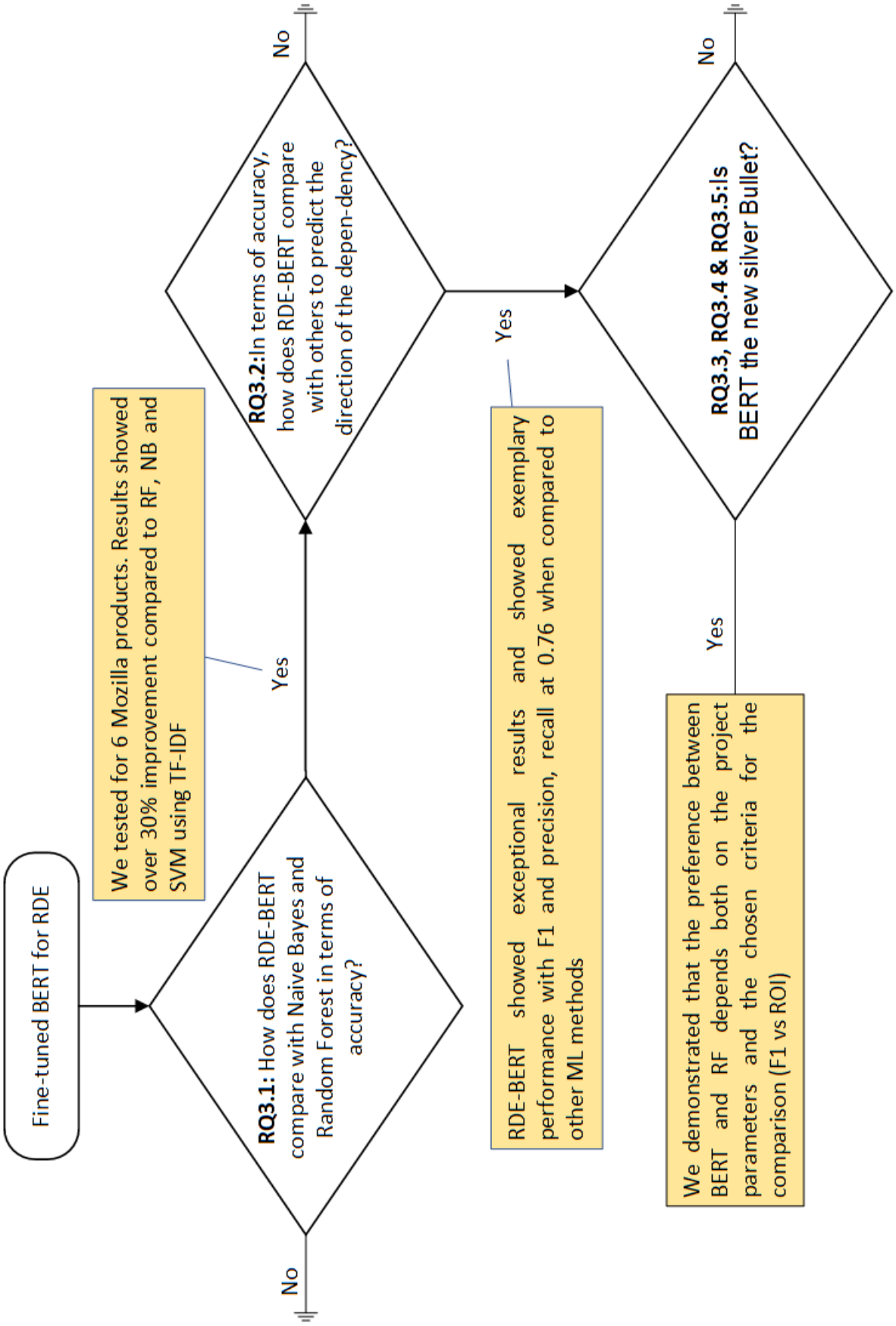


Figure 7.8: Birds eye view of the logical connection between various RQs and the results from their evaluation shown in this figure.

Part IV

Addressing Performance Evaluation Challenge (RQ4)

Chapter 8

Beyond Accuracy: ROI of Data Analytics

The unprecedented access to data has rendered a remarkable opportunity to analyze, understand, and optimize the investigation approaches in almost all the areas of (Empirical) Software Engineering. However, data analytics is time and effort consuming, thus, expensive, and not automatically valuable.

This work demonstrates that it is crucial to consider Return-on-Investment (ROI) when performing Data Analytics. Decisions on "How much analytics is needed"? are hard to answer. ROI could guide for decision support on the What?, How?, and How Much? analytics for a given problem.

The proposed conceptual framework is validated through two empirical studies that focus on requirements dependencies extraction in the Mozilla Firefox project. The two case studies are (i) Evaluation of fine-tuned BERT against Naive Bayes and Random Forest machine learners for binary dependency classification and (ii) Active Learning against passive Learning (random sampling) for *Requires* dependency extraction. For both the cases, their analysis investment (cost) is estimated, and the achievable benefit from Data Analytics (DA) is predicted, to determine a break-even point of the investigation.

For the first study, fine-tuned BERT performed superior to the Random Forest, provided that more than 40% of training data is available. For the second, Active Learning achieved higher F1 accuracy within fewer iterations and higher ROI compared to Baseline (Random sampling based RF classifier). In both the studies, estimate on, How much analysis likely would pay off for the invested efforts?, was indicated by the break-even point.

Decisions for the depth and breadth of DA of empirical data should not be made solely based on the accuracy measures. Since ROI-driven Data Analytics provides a simple yet effective direction to discover when to stop further investigation while considering the cost and value of the various types of analysis, it helps to avoid over-analyzing empirical data.

8.1 Introduction

Return-on-Investment (ROI) is of great interest in engineering and business for arriving at decisions. This is true in Software Engineering (SE) as well. For example, Silverio et al. [106] evaluated cost-benefit analysis for the adoption of software reference architectures for optimizing architectural decision-making. Cleland et al. [40] studied the ROI of heterogeneous solutions for the improvement of the ROI of requirements traceability. Recent data explosion in the form of big data and advances in Machine Learning (ML) have posed questions on the efficiency and effectiveness of these processes that have become more relevant. In this paper, we present a retrospective evaluation of two empirical studies taken from the field of requirements dependency analysis for the benefit of ROI.

Data Analytics in SE (also called "Software Analytics" by Bird et al. [30]) is a term widely used, sometimes with a slightly different meaning. We subsume all efforts devoted to collecting, cleaning, preparing, classifying, analyzing data, and interpreting the results as *Data Analytics (DA)*. In SE, the goal of DA is to provide better insights into some aspects of the software development life-cycle, which could facilitate some form of understanding, monitoring, or improvement of processes, products or projects.

SE is uncertain in various ways. SE is highly human-centric, and processes are not strictly repeatable. The goals and constraints of software development are dynamically changing. Experimentation and DA are inherently arduous under such circumstances. The famous Aristotle [21] is widely attributed with a saying, "It is the mark of an educated mind to rest satisfied with the degree of precision which the nature of the subject admits and not to seek exactness where only an approximation is possible". Figure 8.1 shows a typical ROI (cost-benefit) curve of technology usage. Following some phase of increase, the curve reaches saturation, so, beyond that point, further investment does not pay off. We contemplate that a similar behaviour holds true for applying DA. **Our research hypothesis** is that ROI-driven DA helps to determine the break-even point of investment and thus optimizes resources spent in this process.

Paper structure: Section 8.2 discusses related work. The problem formulation is detailed in Section 8.3. Section 8.4 explains the empirical ROI investigation approach for the two problems. A discussion of the applicability of the results is elaborated in Section 8.6.

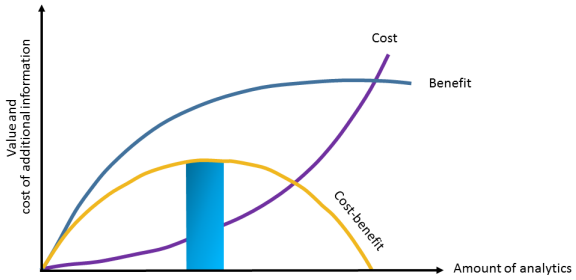


Figure 8.1: Break-even point from cost-benefit analysis of technology investment.

8.2 Related Work

8.2.1 ROI Analysis in Software Engineering

Evaluating the profitability of expenditure helps to measure success over a period of time thus takes the guesswork away from the concrete decision-making process. For instance, Erdogmus et al. [65] analyzed the ROI of quality investment to bring its importance in perspective and posed important questions, "We generally want to increase a software products quality because fixing existing software takes valuable time away from developing new software. But how much investment in software quality is desirable? When should we invest, and where?".

Begel & Zimmermann [22] composed a set of 145 questions - based on a survey with more than 200 developers and testers - that are considered relevant for DA at Microsoft. One of the questions:"How important is it to have a software DA team answer this question?", expected answer on a five-point scale (*Essential* to *I don't understand*). Although it provides a sneak peek of the development and testing environments of Microsoft, it does not prove any emphasis on any form of ROI. Essentially, we speculate that the ROI aspect was softened into asking for the perceived subjective importance through this question.

Boehm et al. [33] presented quantitative results on the ROI of Systems Engineering based on the analysis of the 161 software projects in the COCOMO II database. Van Solingen [155] analyzed the ROI of software process improvement and took a macro perspective to evaluate corporate programs targeting the improvement of organizational maturity. Ferrari et al. [71] studied the ROI for text mining and showed that it has not only a tangible impact in terms of ROI but also an intangible benefits - which occur from the investment in the knowledge management solution that is not directly translated into returns, but that 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.

Ruhe and Nayebi [135] proposed the *Analytics Design Sheet* as a means to sketch the skeleton of the main components of the DA process. The four-quadrant template provides direction to brainstorm candidate DA methods and techniques in response to the problem statement and the data available. In its nature, the sheet is qualitative. ROI analysis goes further and adds a quantitative perspective for outlining DA.

8.2.2 Empirical Analysis for Requirements Dependency Extraction

The extraction of dependencies among requirements is an active field of SE research. The practical importance of the topic was confirmed by our survey [55]. More than 80% of the participants agreed or strongly agreed that (i) dependency type extraction 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) [124] [139], WSL technique [51], hybrid techniques [52] and DL [78]. However, none of

the approaches considered ROI to decide among techniques and the depth and breadth of their execution level.

8.3 Methodology

Different models exist that provide guidance to perform DA. Wieringa [165] provides a checklist for what he calls the design cycle and the empirical cycle. In this study, we use the term *Scoping* for defining the problem and the analysis objectives. Scoping also means defining the boundaries that help to exclude non-essential parts of the investigation. Analysis of the projected *Return-on-Investment (ROI)* serves as an input for scoping.

8.3.1 Research Question

DA follows a resource and computation-intensive process constituting data gathering and processing components that are the non-trivial proportion of the total research cost. Thus, it is essential to account for these to compute the overall cost-benefit and optimize it further.

Our aim is to look at DA for empirical studies retrospectively (already conducted studies in the past). In particular, we are interested in Requirements Dependency Analysis (RDA) based studies. Through this research, we define and validate the principal concepts needed for ROI-driven DA. Our research question is:

RQ4.1: What are the benefits of ROI-driven Data Analytics in the studies focusing on Requirements Dependency Analysis?

Justification: As for any investment, it is most important to know how much is enough. There is no incentive to invest in analytics just for the sake of performing some analysis. Although one cannot claim exactness from this, it is worthwhile to get some form of guidance on where (which techniques) and how far (how much of it) one should go. To make the analysis concrete, we have selected RDA as the area of our specific investigations.

8.3.2 Cost Factors

Data processing is an umbrella term used to combine data collection (C_{dg}), pre-processing (C_{pp}) and labeling (C_l) 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. For example, supervised Machine Learning (ML) requires a large amount of annotated data, to begin with, whereas Active Learning acquires these annotations over a period of time in iterations until a stopping condition for classification operation is reached [141]. Additionally, there is a cost associated with modeling and evaluation (C_e).

8.3.3 Value Factors

The value returns or "benefits" are defined based on the needs of the decision problem. In the context of dependency extraction, the benefit could be modeled in terms of the ability of the ML model to identify a larger number of

Table 8.1: Parameters used for ROI computation

	Symbol	Meaning	Unit
Cost	C_{dg}	Data gathering time	Minutes
	C_{pp}	Pre-processing time	Minutes
	C_e	Evaluation time	Minutes
	C_l	Labeling time	Minutes
	$C_{resource}$	Human resource cost	$ per hour
Benefit	B_{reward}	Value per TP	$
	$B_{penalty}$	Penalty per FN	$
	$BF1_{iteration}$	F1 difference	Number
	$PValue$	Projected value per 1% F1 improvement	$
Others	H	#Human resources	Number
	N_{train}	Size of the training set	Number
	N_{test}	Size of the test set	Number
	N	$N_{train} + N_{test}$	Number

dependencies correctly (higher # of True Positives TP: B_{reward}) while limiting misclassification (reduced # of False Negatives FN: $B_{penalty}$). Conversely, the benefit could also be determined based on the net value ($PValue$) of change of accuracy ($BF1_{iteration}$) in every iteration, especially when using Active Learning. Table 8.1 lists the relevant cost components and their corresponding units. These will be utilized to compute the ROI later for the two different problems in Section 4.4.

8.3.4 ROI

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

$$ROI = (Benefit - Cost)/Cost \tag{8.1}$$

Costa et al. [41] distinguished the "hard ROI" from the "soft ROI". The former refers to the direct additional revenue generated and cost savings. The latter improved productivity, customer satisfaction, technological leadership, and efficiencies.

8.4 ROI of Techniques

We have selected the area of requirements dependency analysis (RDA) to illustrate and initially validate our former conceptual framework. In what follows, we introduce the key terms needed to formulate two Empirical Analysis Studies called EAS 1 resp. EAS 2.

8.4.1 Problem Statement

Following are the definitions of dependency types that are used to state the two studies. For a set of requirements R and a pair of requirements $(r, s)\ \epsilon\ R \times R$

1) An ***Independent*** relationship is defined as the absence of any form of relationship between a pair of requirements.

2) A ***Dependent*** relationship is defined as the complement set of Independent. i.e., there exists at least one type of the dependency types such as *Requires, Similar, Or, And, XOR, value synergy, effort synergy* etc. between r and s.

3) ***Requires*** is a special form of *Dependent* relationship. If r requires s, or s requires r, then, r and s are in a *Requires* relationship

4) ***Other*** type of dependency is when (r, s) is *Dependent* and the dependency type is not Requires (could be any of the other dependency types mentioned in (2))

Problem 1- Binary requirements dependency extraction: For a given set R of requirements and their textual description, the binary requirements dependency extraction problem aims to classify each pair (r,s) ϵ $R \times R$ as *Dependent* or *Independent*.

Problem 2- Specific requirements dependency extraction of the type *Requires*:

For a given set R of requirements and their textual description, the Requires dependency extraction problem aims to classify for each pair (r,s) ϵ $R \times R$ if they are in a *Requires* relationship.

8.4.2 Empirical Analysis Studies (EAS)

In this section, we formulate two Empirical Analysis Studies, EAS 1 and EAS 2, to investigate the two problems explained above. We aim to analyze and compare Bidirectional Encoder Representations from Transformers (BERT), and Active Learning (AL), both proven to be of interest in general and pre-evaluated for their applicability to the stated problems, with conventional ML. For the two studies, we examine the (F1) accuracy and the ROI of the whole process of DA.

EAS 1: We compare two supervised classification algorithms: Naive Bayes (NB) and Random Forest (RF) - ML algorithms successfully and prominently used for text classification [140] in the past, with a fine-tuned BERT model [59]. The analysis was performed for an incrementally growing training set size to capture its impact on F1 and ROI.

BERT (Bidirectional Encoder Representations from Transformers) [59] is a recent technique published by researchers from Google. BERT is applying bidirectional training of Transformer, a popular attention model, to language modeling, which claims to be state-of-the-art for NLP tasks. In this study scenario, we explore the question, "How does fine-tune BERT compare with conventional algorithms on an economical scale?" by comparing models' effectiveness with incurred ROI.

EAS 2: Random sampling (Passive Learning) randomly selects a training set - referred to as *Baseline* in the rest of the paper. Active Learning selects the most informative instances using various sampling techniques such as MinMargin and LeastConfidence [141]. We compare *Baseline* with AL using RF as a classifier for this scenario. The analysis was done by adding a few training samples in every iteration concurrently to classify the unlabeled instances.

Active Learning (AL) is a ML method that guides a selection of the instances to be labeled by an oracle (e.g., human domain expert or a program) [141]. While this mechanism has been proven to positively address the question, "Can machines learn with fewer labeled training instances if they are allowed to ask questions?", through this exploration, we try to answer the question,"Can machines learn more economically if they are allowed to ask questions?" [142].

8.4.3 Data

The online bug tracking system Bugzilla [2] is widely used in open-source software development. New requirements are logged into these systems in the form issue reports [145] [26] which help software developers to track them for effective implementation [146], testing, and release planning. In Bugzilla, feature requests are a specific type of issue that is typically tagged as "enhancement" [116]. We retrieved these feature requests or requirements from *Firefox* and exported all related fields such as Title, Type, Priority, Product, Depends_on, and See_also.

Data collection: Collecting data from Bugzilla was a substantial effort that was carried out in multiple rounds. We collected 3,704 enhancements from *Firefox* using REST API through a python script such that each one of the enhancements considered for retrieval is dependent on at least another one in the dataset. The data spanned from 08/05/2001 to 09/08/2019.

Data preparation: The complete data was analyzed to eliminate special characters and numbers. Then dependent requirement pairs were created based on the depends_on (interpreted as *Requires* dependency) field information for each one of the enhancements. Requirements with no dependency between them were paired to generate *Independent* class dataset. Further, sentence pairs that had fewer than three words in them were filtered out resulting in 3,373 *Requires*, 219 *Other* and 21,358 *Independent* pairs.

Pre-processing and feature extraction: The data was first processed to eliminate stop words and then lemmatized following the traditional NLP pipeline [16]. For supervised and AL ML, we used the Bag Of Words (BOW) [129] feature extraction method, which groups textual elements as tokens. For applying BERT, we retained sentence pairs in their original form (without stop word removal and lemmatization).

Classifiers: For both NB and RF, the data was split into train and test (80:20) and balanced between classes. Also, hyper-parameter tuning was performed and the results for 10-fold cross-validation were computed, followed by testing (on unseen data).

To fine-tune the BERT model, we used *NextSentencePrediction*[1], a sentence pair classification pre-trained BERT model, and further fine-tuned it for the RDA specific dataset on Tesla K80 GPU on Google Colab[2].

[1]https://huggingface.co/transformers/model_doc/bert.html#bertfornextsentenceprediction
[2]https://colab.research.google.com/

8.4.4 ROI Modeling

EAS1

The classification algorithms such as RF and NB, have been explored in NLP based SE problems. These algorithms are driven by the feature extraction aspect to a great extent. Thus, could influence their effectiveness on classification outcomes. However, feature extraction is problem specific and incurs substantial cost and access to domain expertise.

On the other hand, BERT eliminates the need for feature extraction since it is a language model based on deep learning. BERT, pre-trained on a large text corpus, can be fine-tuned on specific tasks by providing only a small amount of domain-specific data.

In this empirical analysis, we conducted classification by utilizing a fraction of the whole dataset for training and testing for a small fixed data set. This was repeated by slowly increasing the fraction of the training set and results were captured.

During every classification, $Cost$ and $Benefit$ were computed using various parameters explained in Table 8.1. $Cost$ is the sum of the data processing costs ($(C_{dg} + C_{pp} + C_e + C_l)/60$) (in hours) for a fraction (N%) of training set. This is further translated into dollar cost based on hourly charges ($C_{resource}$) of H human resources.

$$Cost = N\% * \frac{(C_{dg} + C_{pp} + C_e + C_l)}{60} * H * C_{resource} \tag{8.2}$$

$Return$ computations for RDA, assumes reward (B_{reward}) for identifying the dependent requirements (TP) while penalizing ($B_{penalty}$) instances that were falsely identified as independent (FN).

$$Benefit = TP * B_{reward} - FN * B_{penalty} \tag{8.3}$$

Table 8.2: Parameter settings for the two empirical analysis scenarios

Parameters	**Values**
$C_{fixed} = C_{dg} + C_{pp} + C_e$	1 min/sample
C_l	0.5 min/sample
$C_{resource}$	\$400/hr
H	1
N	4,586
B_{reward}	\$500/TP
$B_{penalty}$	\$500/FN
$BF1_{iteration}$	$=F_{cur} - F_{prev}$
$PValue$	\$10,000 per percent F1 improvement

EAS 2

In this empirical analysis, we compared AL with a traditional random sampling based classification- *Baseline* - using the RF ML algorithm.

Beginning with 60 training samples of each class (*Requires, Independent and Other)*, we developed multi-class classifiers for both AL and Baseline for this empirical study scenario. When AL used MinMargin sampling technique[3] to identify 20[4] most uncertain instance (requirement pair) for oracle to label, baseline randomly selected 20 instances and added to the training set along with their label, thus, kept the two approaches comparable in all the 20 iterations. Since data is already labeled, for AL, we pretend they are unlabeled until queried and labeled by a simulated oracle in this scenario.

The *Cost* is determined by first computing the sum of total processing time in person hours (= *Cost*) taken for data processing ($C_{fixed} = C_{dg} + C_{pp} + C_e$)), labeling ($C_l$) of train set ($N_{train}$) and data processing cost (C_{fixed}) for testing. This is further translated into dollar cost (=C_{total}) based on hourly charges ($C_{resource}$) of H human resources.

$$Cost = \frac{N_{train} * (C_{fixed} + C_l) + N_{test} * C_{fixed}}{60}$$

$$C_{total} = Cost * H * C_{resource} \tag{8.4}$$

Likewise, $Benefit$ is defined as the monetary value associated with a 1% improvement in F1 score ($BF1_{iteration}$) between subsequent iterations.

$$Benefit = BF1_{iteration} * PValue \tag{8.5}$$

8.5 Results: Benefits of ROI-driven Data Analytics - RQ4.1

In the real-world, cost and benefit values are hard to get and are uncertain. All the results presented in this section are based on the parameter settings given in Table 8.2. The settings reflect practical experience but are not taken from a specific data collection procedure. We claim that the principal arguments made in our study are independent of these settings.

8.5.1 EAS 1

Figure 8.2 provides the "accuracy only view" and shows that F1 gradually increases with the increasing training size for the three ML algorithms: NB, RF, and BERT. However, all three ML algorithms reach a saturation towards larger training set sizes. While BERT performed exceptionally well when training set size exceeded 42%, it could have been ideal to pre-determine "How much training is enough?". Thus we selected the top two classifiers (Figure

[3]MinMargin sampling technique performed well compared to Least Confidence and Entropy thus, we utilized MinMargin for this study

[4]The tests were performed with#samples = 10, 15 and 20. In this study, we will discuss results related to #samples=20

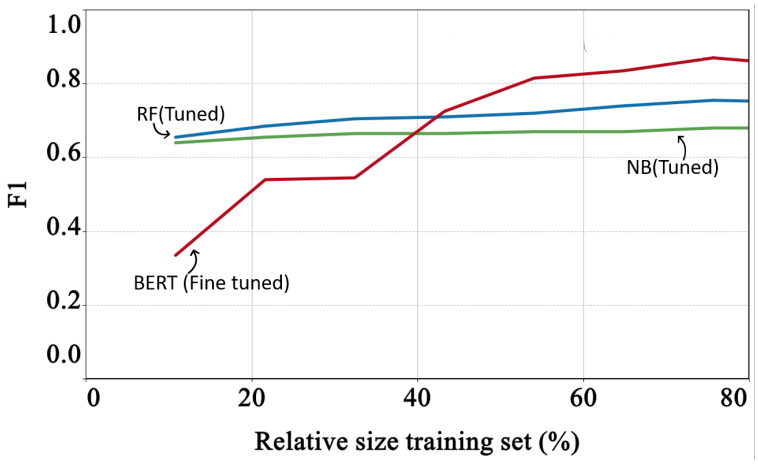
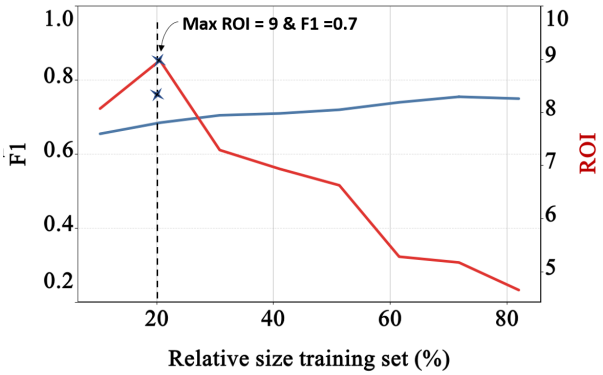
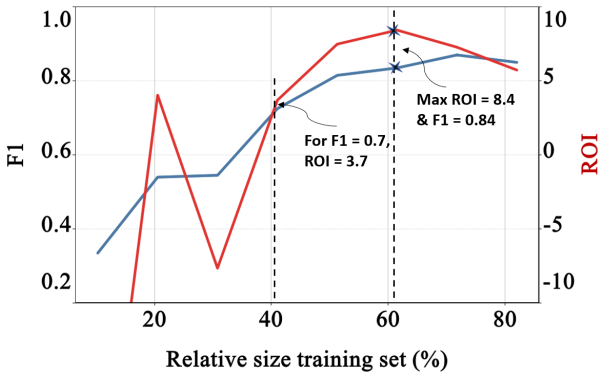


Figure 8.2: F1 score plot for NB, RF and BERT trained over increasing training set size, F1 improves, but plateaus beyond 70% train set



(a) F1 vs ROI for Random Forest shows learning can stop at 20% train set



(b) F1 vs ROI for Fine-tuned BERT shows that beyond 60% ROI deteriorates

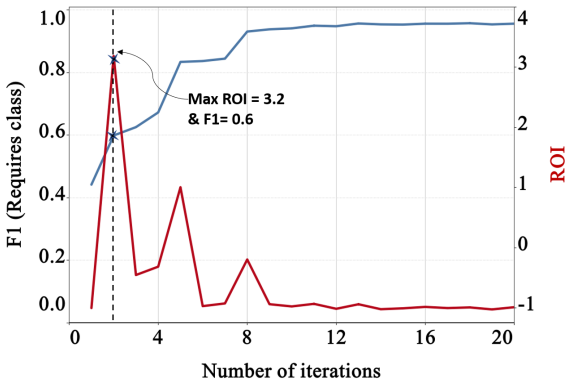
Figure 8.3: Empirical Analysis Scenario 1 (EAS 1)

8.2): BERT and RF and applied the monetary values (Table 8.2) for the various cost and benefit factors defined in Table 8.1 and computed the ROI.

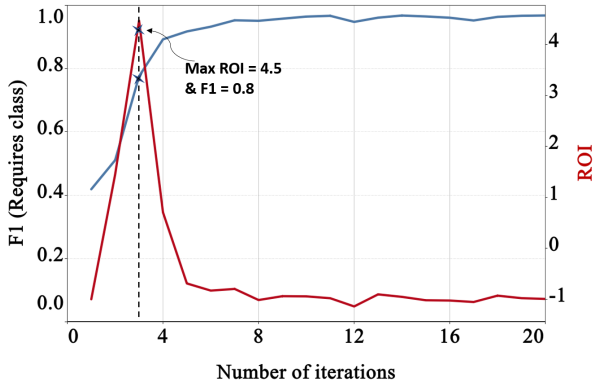
Figure 8.3a and 8.3b show the results for RF and BERT, respectively. The ROI behaviour is not monotonous and peaks for both cases. Although RF classification achieved the highest ROI with just 20% of training set and accuracy of F1 = 0.7, highest F1 value of 0.75 was achieved along with the lowest ROI of 3.7 as annotated in Figure 8.3b.

For RF classification and applying ROI arguments, learning can be stopped with 20% of the training set.

Now looking at BERT classification, the best ROI-driven results: F1 = 0.84 and an ROI = 8.43, were achieved with the 60% training set. Although F1 rose to 0.9 with 70% training set size, ROI dropped to 7.27. For the recom-



(a) F1 vs ROI for Baseline shows that learning beyond 2nd iteration does not generate positive ROI



(b) F1 vs ROI for AL shows that at the 3rd iteration, ROI peaks and starts to deteriorates on wards

Figure 8.4: Empirical Analysis Scenario 2 (EAS2)

mendation of 20% of training set size, ROI has a local optimum. BERT in general performs well on the F1, however, is it worth the ROI? needs to be explored.

For training set sizes of at least 40% of the size of the whole set, BERT performed better than RF in terms of both accuracy and ROI.

8.5.2 EAS 2

We analyzed the ROI for *Baseline* against AL for classifying the *Requires* class. The results are shown in Figure 8.4a and Figure 8.4b. Similar to EAS 1, we applied the values from Table 8.2 and equations (8.4) and (8.5) to compute cost and benefit at every iteration for both the approaches. For the *Baseline* approach, ROI peaked at 3.2 and F1 = 0.6, in the very 2nd iteration. Onwards, ROI drastically decreased which indicated lesser value for increasing training set by random sampling (*Baseline*) method.

Similar behavior was observed for the AL approach. shown in Figure 8.4b. The peak here was after three iterations with values ROI = 4.5 and F1 = 0.8.

Both Baseline and AL showed the best ROI performance in the early iterations. Higher F1 accuracy needs additional human resources and reduces the ROI.

8.6 Discussion

For the problem of RDA, we explored the potential value of ROI-driven decisions. When chasing higher accuracy, there is a risk of over analyzing empirical data. In the sense that the value added due to increased accuracy is not justifiable by the additional effort needed in achieving it.

What does a high or low ROI mean for DA? : If available, a high ROI ratio indicates that there is a

substantial benefit expected from following the recommendations derived from DA. Assuming that the ROI-driven suggestions are implemented, the small improvements achieved for solving the decision problems with high impact could justify the effort invested. Analysis related to effort and benefit, targeting high ROI, also implies simplicity first. Advanced methods are needed, but they are hard to justify practical application if a similar type of insight could be reached from a much simpler analysis, e.g., from descriptive statistics.

What is the risk of ignoring analysis?: The calculation of ROI is based on the value and effort estimates and thus only provides an approximation. In all types of exploratory data analysis, the emphasis is mainly on creating new research hypotheses or validating existing assumptions. In these cases, the notion of ROI is not the primary concern. Also, estimates for value and effort needed are highly dependent; hence, the ROI might only serve as a soft recommendation. On the other hand, whenever the ROI can be determined as a reasonable estimate, even after using intervals of best and worst-case performances, then ignoring ROI means to potentially waste effort for analysis that does not pay off the investment made. For EAS 1, if the training size set was limited to 30%, RF could be considered as a better choice over BERT. However, with the possibility to increase the training set size, the BERT approach could be favored.

Acknowledgement

We thank Atharva Naik and Venessa Chan for useful comments. This work is supported by the Natural Sciences and Engineering Research Council of Canada, Discovery Grant RGPIN-2017-03948.

Chapter 9

ROI of ML Classification

Machine Learning (ML) can substantially improve the efficiency and effectiveness of organizations and is widely used for different purposes within Software Engineering. However, the selection and implementation of ML techniques rely almost exclusively on accuracy criteria. Thus, for organizations wishing to realize the benefits of ML investments, this narrow approach ignores crucial considerations around the anticipated costs of the ML activities across the ML life-cycle, while failing to account for the benefits that are likely to accrue from the proposed activity. We present findings for an approach that addresses this gap by enhancing the accuracy criterion with return on investment (ROI) considerations. Specifically, we analyze the performance of the two state-of-the-art ML techniques: Random Forest and Bidirectional Encoder Representations from Transformers (BERT), based on accuracy and ROI for two publicly available data sets. Specifically, we compare decision-making on requirements dependency extraction (i) exclusively based on accuracy and (ii) extended to include ROI analysis. As a result, we propose recommendations for selecting ML classification techniques based on the degree of training data used. Our findings indicate that considering ROI as additional criteria can drastically influence ML selection when compared to decisions based on accuracy as the sole criterion.

9.1 Introduction

Machine Learning (ML) includes methods, tools, and techniques for inferring models from data and has provided successful applications of classification and prediction algorithms. In the area of software development and evolution, a recent study [143] revealed that there is a spectrum of applications of ML across the software development life-cycle, with most of the applications belonging to the category of *Quality Assurance and Analytics*.

There exists an extensive variety of ML algorithms and this pool is growing steadily. A recent study [143] listed Decision Trees, Naive Bayes, and Random Forrest as the techniques most frequently applied in Software Engineering. However, it is important to determine which algorithm works well for a given problem and which are less effective. The performance of any ML technique is generally measured in terms of accuracy (or similar measures). However,

the success of ML does not only depend on the algorithms used because ML is a process with various interdependent steps and the investments made in this process need to be related to the return gained from its results. This paper puts estimating the return-on-investment (ROI) of ML in the spotlight. ROI is most widely used in the context of business analysis, which we extend to ML classification problems. In particular, we focus on the decision-making of ML method selection, i.e., to determine when to stop the process and how much additional investment is needed to achieve a target goal (result).

The most important prerequisite for generating accurate ML models is high-quality training data, however securing such data is often an arduous task. Additionally, engineering and selecting appropriate features is especially time-consuming and requires a vast amount of effort and resources [72]. The benefits gained from the application of ML can be dramatically offset due to data collection and data pre-processing activities, which incur substantial costs and effort.

ROI is of great interest in engineering and business, where it is widely used as a guide for decision-making. This is true in Software Engineering (SE) as well. For example, Martinez Fernandez et al. [106] evaluated cost-benefit analysis for the adoption of software reference architectures for optimizing architectural decision-making. Cleland Huang et al. [40] studied the ROI of heterogeneous solutions for the improvement of requirements traceability. However, the recent data explosion in the form of big data and advances in Machine Learning (ML) have posed questions on the efficiency and effectiveness of these processes that have become more relevant.

In this paper, we present two empirical studies from the field of requirements engineering. While it serves as one sample topic for a broader problem, Requirements Dependency Extraction (RDE) has been a topic of interest for both researchers and practitioners. In particular, we study a fine-tuned *BERT* (Bidirectional Encoder Representations from Transformers) [59], a recent technique published by researchers from Google, with Random Forest for solving RDE. BERT uses bidirectional training of transformer, a popular attention model, to language modelling, which claims to be state-of-the-art for NLP tasks. We compare BERT with Random Forest (RF), a widely used ML technique that serves as a baseline for comparison.

The objective of this study is to present an alternative method to evaluate ML algorithms. In that sense, we demonstrate the perspective of the returns ML algorithms would generate for the investment done while choosing a particular method for a given problem. Our research contributions are as follows

- Describe an ML process model for ML classification and perform related ROI modeling.
- Empirically evaluate Random Forest and fine-tuned BERT for textual classification in the context of requirements dependency classification (RDE) using accuracy and ROI.

The remainder of the paper is structured as follows: Section 9.2 provides a motivating example of this study, followed by Section 9.4 which explains requirements dependency, its extraction, practical relevance, and research questions. Section 9.5 elaborates our ROI modeling of the ML process. Data used in this study are detailed in Section 9.6 followed by empirical results in Section 9.7. The discussion Section 9.8 details implications and limitations of this study before providing a discussion in Section 9.9.

9.2 Motivating Example

Figure 8.1 shows a prototypical ROI curve for technology investment [134]. When trying to achieve better results, the investment's cost (or effort) is growing over time, typically non-linearly. However, the benefit achieved from that investment eventually reaches some saturation point beyond which almost no further improvement is achieved. In total, a saturation point is achieved, after which further investment does not pay off anymore (i.e., point of diminishing returns).

Thus, the most crucial question arises-*Do similar arguments apply for ML classification in Software Engineering?* While this could be true in general, we study it in the context of the requirements dependency classification problem.

Deshpande et al. [55] report the results of a recent survey for requirements dependency classification and maintenance, with 76% of responses (out of 70) from practitioners. More than 80% of the participants agreed or strongly agreed that dependency type classification is difficult in practice; dependency information has implications for maintenance, and ignoring dependencies has a significant impact on project success [55].

Applying the advanced NLP technique BERT, we performed an ROI analysis on the requirements dependency classification. Automating this process saves time, and making the classification more effective helps better align the development process with the existing dependencies. For example, if a requirement r depends on another requirement s, then the implementation of s should precede implementing r. Violating this logical dependency will not only delay the usage of r but also decrease the effectiveness of testing.

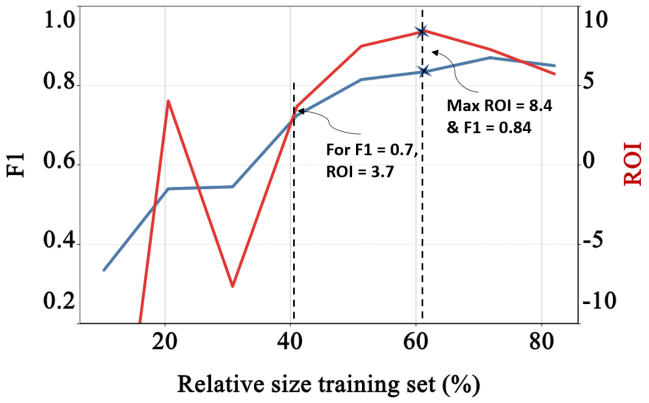


Figure 9.1: ROI vs F1 of BERT for Firefox dataset [56] shows that beond 60% train set, ROI deteriorates although F1-score continues to improve

Figure 9.1 shows that there is an early peak in the ROI of using BERT. Since it is a very data-intensive technique, the ROI goes down with increasing training set size before the ROI reaches the global maximum. By comparison, considering only the harmonic mean (F1) of precision and recall gives a different recommendation for training set size. We discuss this in detail in Section 5.

9.3 Related Work

Although ROI is used in various contexts in Software Engineering and Data analytics, we discuss noted findings from the literature in the context of our proposed research.

9.3.1 Exploration of ROI in Software Engineering

Farbey et al. [66] explained that as a product moves through its life cycle, various evaluation methods such as ROI, Multi-Objective multi-criteria, Value analysis etc. play an important role in decision making. In this study, ROI was recommended either as a strategy to decrease uncertainty in the business area or to improve knowledge of how technology would operate.

Khoshgoftaar et al. [89] presented an interesting case study of a large telecommunication software system and demonstrated a methodology 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) values of classification models. Although these cost-benefit models were ahead of their time, they did not consider the time and effort investment done on data and metrics gathering for cost computation. In another study on calculating ROI in the software product line, Bockle et al. [31] derived cost and benefit estimates based on organization level criteria, such as cost to the organization and cost of reuse. However, this did not involve data analytics of any form.

The guesswork could be eliminated from the decision-making process while evaluating the profitability of expenditure, which could help measure success over time. For instance, Erdogmus et al. [65] analyzed the ROI of quality investment to bring its value into perspective; posed an important question, "We generally want to increase a software product's quality because fixing existing software takes valuable time away from developing new software. But how much investment in software quality is desirable? When should we invest, and where?", which we think is difficult to quantify yet crucial for the success of software-based products.

Begel & Zimmermann [22] gathered and listed a set of 145 questions in a survey of 200 Microsoft developers and testers and termed them relevant for DA at Microsoft. One of the questions: "How important is it to have a software DA team answer this question?", expected answer on a five-point scale (*Essential* to *I don't understand*). Although this analysis provides a sneak peek of the development and testing environments of Microsoft, it does not provide emphasis on any form of ROI. Essentially, we speculate that the ROI aspect was softened into asking for the perceived subjective importance through this question.

Boehm et al. [33] [32] presented quantitative results on the ROI of Systems Engineering based on the analysis of the 161 software projects in the COCOMO II database. Ruhe and Nayebi [135] proposed the *Analytics Design Sheet* as a means to sketch the skeleton of the main components of the DA process. The four-quadrant template provides direction to brainstorm candidate DA methods and techniques in response to the problem statement and the available data. In its nature, the sheet is qualitative, while ROI analysis goes further and adds a quantitative perspective for outlining DA.

Ling et al. [99] 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).

9.3.2 Exploration of ROI in Data Analytics

Ferrari et al. [71] studied the ROI for text mining and showed that it not only has a tangible impact in terms of ROI but also intangible benefits, which arise from the investment in the knowledge management solution. This solution translates the returns directly that must be considered while integrating the financial perspective of analysis with the non-financial ones.

Weiss et al. [163] emphasized how the quality of external data influence the results and quantified the effort of gathering and using such data when it is available at a premium into cost in terms of CPU time, even though the treatment of the subject is limited to a static setting. In a similar vein, Nagrecha et al. [119] proposed a Net Present Value model to determine the cost and impact of analytics programs for an organization.

Taking inspiration from these studies in our research, we not only consider data pre-processing costs as an additional cost aspect but also transform machine learning metrics to dollar amounts, with derived costs and benefits being also validated by industry experts.

9.3.3 Empirical Analysis for Requirements Dependency Classification

Requirements dependencies classification is an active field of SE research. The practical importance of the topic was confirmed by a survey [55] of over 90 participants from the SE industry. Results showed that more than 80% of the participants agreed or strongly agreed that (i) dependency type extraction is difficult in practice, (ii) dependency information has implications on maintenance, and (iii) ignoring dependencies has a significant negative impact on project success.

Several empirical studies have explored diverse computational methods that used Natural Language Processing (NLP) [124] [139], WSL technique [51], hybrid techniques [52] and DL [78] in this context. Recently, Wang et al. [161] explored a semi-automatic ML approach based on traceability to identify requirement dependencies to further identify security vulnerabilities. However, none of the approaches considered ROI to decide among techniques and the depth and breadth of their execution level.

9.3.4 Exploration of Machine Learning Process in Software Engineering

We analyzed 96 papers from IEEE, Scopus, ScienceDirect, and ACM Digital Library which exclusively used ML, and data analytics within software engineering, and software development domains. Precision, Recall, Accuracy, and AUC were by far the most common performance measures used by researchers in these papers. Additionally, the choice of performance measure was generally not justified. Most studies did not present all steps of the ML process,

and most of the papers formally present only 3 steps of the ML process such as data pre-processing, evaluation, and parameter tuning and all these steps are underestimated in terms of effort spent.

This study highlights the merits of simultaneously considering technical and business criteria when evaluating tradeoffs faced within machine learning approaches for requirements dependency extraction (RDE). We extend prior work that focused on comparing various ML techniques based upon technical criteria of accuracy to include broader consideration of the impact i.e. Evaluating value generated by the analysis compared to the costs incurred for the analysis.

9.4 Requirements Dependency Extraction

Similar to requirements elicitation [97], extraction of requirements dependencies is a cognitively difficult problem. These dependencies not only influence the development of software but also impact how requirements operate. In this section, we provide the formal problem definition which serves as an example to demonstrate the value of looking beyond accuracy measure and investing in more general concepts of ROI analysis.

9.4.1 Problem Formulation

While there are different types of dependencies between requirements [174], [36] we provide the definitions just for the ones used in the empirical study . For a set of requirements R and a pair of requirements (r, s) ϵ $R \times R$

1) Two requirements r, s are called ***Independent*** if handling one of them has no logical or practical implication for handling the other one. Otherwise, they are called ***Dependent***.

2) ***Requires*** is a form of *Dependent* relationship. If requirement r requires the requirement s to be implemented, then, r and s are in a *Requires* relationship. Requires is an asymmetric relationship.

3) ***Relates_to*** is another specific form of *Dependent* relationship. Requirement r relates_to requirement s if changing one of them has an impact on the other. *Relates_to* is a symmetric relationship[1].

Problem: Binary Requirements Dependency Extraction (RDE)

For a given set R of requirements and their textual description, the binary Requirements Dependency Extraction problem (RDE) is to decide for a given pair (r,s) ϵ $R \times R$ if (r,s) is in a *Requires* (called problem RDE_1) or in a *Relates_to* (called problem RDE_2) relationship.

9.4.2 Research Questions

In this paper, two research questions (RQs) are addressed:

> **RQ4.2:** How to model the ROI for ML classification? Specifically, how to instantiate the model for the problem of RDE?

[1]There are other types of dependencies such as *DUPLICATES*, *BLOCKS* etc. that also occur in the these datasets, however, we have considered the ones that occur most frequently

Rational: The exclusive consideration of accuracy in the selection of ML classification techniques might be misleading. We consider ROI as an alternative and additional criterion. To study the cost and benefit of the ML classification in a specific context, it is essential to consider the complete process of ML classification and the impact of the results in the original problem space.

RQ4.3: For RDE, how is the preference decision between RDE-BERT and RF impacted by the accuracy criteria F1 that includes ROI?

Rational: We evaluate the impact of the selection criteria through two empirical studies on two open-source software (OSS) datasets: Firefox, a software application from Mozilla family [116] and Typo3 [10], a content management software. Our goal is to evaluate two extraction techniques (RDE-BERT and RF) to demonstrate the impact of the consideration of ROI in addition to accuracy considerations.

9.5 ROI Modeling of ML Classification - RQ4.2

Machine Learning classification is an iterative process comprising a series of steps. Aiming at ROI analysis of ML classification requires a look at the effort consumed for all these steps. In what follows, we describe various ML process steps, we estimate cost and benefit, and project the ROI of ML classification.

Although various ML workflow has been defined in the literature [67] [14] [113], in this section, we present the simplified version of it mainly focusing on the ML process.

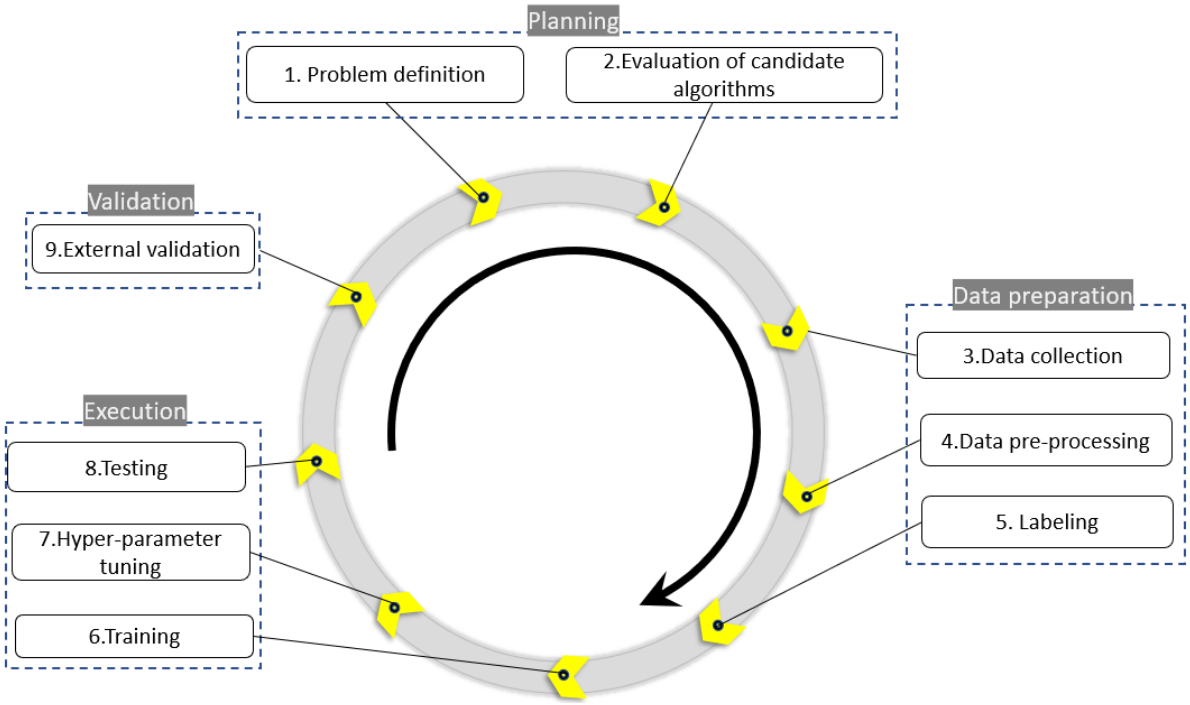


Figure 9.2: Overview of the steps constituting the ML process

9.5.1 Modeling the Process

The process steps are organized into four Phases: A, B, C, and D called Planning, Data Preparation, Execution, and Validation, respectively. Depending on the context, the effort allocated for these steps may vary. However, this approach parallels the process steps and guidelines for pragmatic optimization in software engineering by Ruhe et al. [134].

An overview of the steps is illustrated in Figure 9.2. Here, we did not show all the possible arrows to indicate that loops can, and do, occur between any two steps in the process. The iterative and interactive ML process, involving various phases is summarized as:

Phase A : Planning

Step 1: Scoping and problem formulation Scoping defines the problem context and its boundaries. Problem formulation addresses the key independent and dependent attributes to be considered. As a result of later steps, the problem formulation eventually needs to be adjusted as asking the right question constitutes the largest effort for any application effort.

Step 2: Evaluation of candidate machine learners A variety of ML algorithms exist and new ones are discovered regularly. Commonly used machine learning algorithms include Linear Regression, Logistic Regression, Decision Trees, K-means, Support Vector Machines, Naïve Bayes, Random Forest, and Neural Networks. There is no obvious preference in the sense that "One size fits all". However, there could be recommendations for a particular ML algorithm for a given problem based on its exemplary performance for a similar problem(s). An initial evaluation helps to select the most promising one(s). The selection is influenced by the success criterion of the classification (e.g., accuracy).

Phase B: Data Preparation

Step 3: Data collection Different sources of data might exist for performing ML classification. Data collection looks into what is potentially relevant and checks the type and availability of the data.

Step 4: Data pre-processing Raw data would not be ready for processing through the ML algorithm as it could have duplicates, missing values, and contradictions that need to be tackled first for error-free results. Performing such pre-processing operations, for example, data cleaning, normalization, transformation, feature extraction and selection, etc. are essential for the success of ML classification, but these steps consume a considerable amount of human resources and processing time. The outcome of data pre-processing is the training set which could be processed through ML algorithm further [92].

Step 5: Labeling Labeling is to assign labels to ground truth data [14]. Supervised ML methods need labeled data unlike unsupervised ML methods. Labeling is generally performed by domain experts who identify a set of samples (that are most likely representative of the real-world data) to train the ML models. Depending on the nature of the problem, online crowdsourced platforms could also be used for labeling tasks [121].

Phase C: Execution

Step 6: Training The key idea of ML is to learn from existing data and then apply the resulting model to new data. The quality and quantity of the training data are often as important as the actual machine learning algorithm. To learn from existing data also means that the data set is complete, with known input and output of the observations.

Step 7: Hyper-parameter tuning ML algorithms depend upon several parameters such as named model parameters and named hyper-parameters. Named model parameters can be initialized and updated through the data learning process (e.g., the weights of neurons in neural networks). Named hyper-parameters cannot be directly estimated from data learning and should be set before training an ML model because they define the model architecture. Tuning these parameters means achieving settings that enable good algorithmic performance [94].

Step 8: Testing After training, the model is applied to the selected test set(s) (a small part of labeled data that is held out and excluded from the training process). The larger the number of variables in the real world, the bigger the training and test data should be. From performing testing, classification error counts are captured in the form of a confusion matrix.

Phase D: Validation

Step 9: External validation Success from Step 8 does not automatically imply the success of the results in the context of the application. The validity of the problem formulation and the data might prevent the applicability of the results (i.e., not actionable within the organization resulting in significant wasted effort). Internal validation approaches such as cross-validation can not guarantee the quality of a machine learning model due to potentially biased training data. External validation is critical for evaluating the generalization ability of the machine learning model, where independently derived datasets (external) are leveraged as validation datasets. While such independent validation is also sometimes used to refer to a validation study by other researchers that the researchers who developed the model [85].

9.5.2 Modeling Cost and Benefit

Acknowledging that ML classification is a process of steps with possibly multiple iterations suggests the need to look at the estimated cost for all these steps. Cost estimation is known to be inherently difficult in software engineering [144]. The same is true for value prediction. Despite many factors influencing the costs and benefits of ML classifications, we provide a preliminary model to allow a rough estimate of the ROI.

For cost estimation, we make the assumption that the total cost of performing ML classification with any given ML technique is the sum of cost components of the four phases outlined in the previous section. To simplify the model, we focus on Phase B (Data Preparation) and Phase C (Execution) and ignore the other two phases. Finally, we assume an 80:20 effort (and cost) ratio between Phase B and Phase C, emphasizing the fact that the majority of effort is spent on data preparation.

For modeling the benefit of the classification results, we are looking at classification errors and their cost (penalty) created. A *confusion matrix* CM is a matrix that contains information relating actual with predicted classifications. For n classes, CM will be an $n \times n$ matrix associated with a classifier. Table 9.1 shows the principal entries of CM for binary classification.

Table 9.1: 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)

Table 9.2: Parameters used for ROI computation

		Symbol	**Meaning**	**Unit**
Cost factors[1]	**Phase A**	C_{pl}[2]	Planning phase cost	$
	Phase B	C_{dg}	Data gathering cost	$
		C_{pp}	Pre-processing cost	$
		C_l	Labeling cost	$
	Phase C	C_t[2]	Hyper-parameter tuning cost	$
		$C_{train/test}$	Training and testing cost	$
	Phase D	C_e[2]	External Validation cost	$
Classification Penalty		$Cost_{FP}$	Penalty per FP	$
		$Cost_{FN}$	Penalty per FN	$
Others		N_{HR}	#Human resources	Number
		C_{HR}	Human Resource cost	$/hr
		N_{train}	Size of the training set	Number
		N_{test}	Size of the test set	Number
		N	$N_{train} + N_{test}$	Number
		$Value_{prod}$[3]	Estimated value of the product for a release cycle	$

[1]These are per sample cost factors. All the costs are computed by translating them from minutes to $ by multiplying with resources and cost per hour of the resources
[2]For simplicity few of the cost factors have been assumed to be zero
[3]This value was computed using various cost estimates for a period of one release cycle (= 18 months)

The F1 score is a measure of the model's accuracy based on the training set and defined as the harmonic mean of the model's precision and recall in (9.1).

$$F1 = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{9.1}$$

In the context of dependency classification, 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 independency as a dependency (False Positive)

2) Incorrectly classifying dependency as independent (False Negative). So, using $Cost_{FP}$ and $Cost_{FN}$ as estimated re-work costs due to classification overhead, $Sum(Cost_{FN}, Cost_{FP})$ would be the cumulative expense that a company has to bear.

In a release cycle, if *estimated value* that a product could generate is :$Value_{prod}$ then the $Benefit$ would be the difference of the *estimated value* and the *classification overhead*. Table 9.2 lists the relevant cost components and their corresponding units.

9.5.3 Modeling ROI

During every classification, $Cost$ and $Benefit$ were computed using the parameters explained in Table 9.2. *Cost factors* are data processing costs (Phase B and Phase C) for all the train (N_{train}) and test (N_{test}) samples (n) in every iteration. This is further translated into dollar-cost by multiplying with hourly charges (C_{HR}) of N_{HR} human resources.

$$Cost = n \times \sum_{all\ applicable} Cost\ factors \times N_{HR} \times C_{HR} \tag{9.2}$$

Return computations for RDA, assumes reward ($Cost_{FP}$) for misidentifying the independent requirements (FP) and heavily penalizing ($Cost_{FN}$) instances that were falsely identified as independent (FN).

$$TotalPenalty = FP \times Cost_{FP} + FN \times Cost_{FN} \tag{9.3}$$

$$Benefit = Value_{prod} - TotalPenalty \tag{9.4}$$

Return and investment are context-specific terms, and studying the ROI of Machine Learning classification needs tailoring to the context of the study. To determine the ROI, we follow the simplest form of its calculation relating to the difference between $Benefit$ and $Cost$ to the amount of $Cost$ as shown in (9.5). Both $Benefit$ and $Cost$ are measured as human effort in person-hours.

$$ROI = (Benefit - Cost)/Cost \tag{9.5}$$

The core investigative focus of our study is to evaluate various conditions under which RDE-BERT (fine-tuned BERT using data specific to requirements dependency extraction) is preferable to the baseline ML method: Random Forrest (RF).

In this empirical analysis, beginning with a small train set, classifiers were created, and then the train set was incremented slowly by a fixed factor to generate new classifiers in every iteration until all the data available for training was exhausted. In every iteration, the classifiers were tested for a small fixed data set to capture the results.

9.6 Data and Experiment Setup

Online bug tracking systems such as Bugzilla [2] and Redmine [5] are widely used in open-source software development. Feature requests, tasks, bugs, epics, stories, features, enhancements, and new requirements are logged into these systems in the form issue reports [145] [26] which help software developers to track them for effective implementation [146], testing and release planning [136].

We mined data from Bugzilla and Redmine related to features for the two OSS projects namely, Firefox - a Mozilla web browser application and Typo3 - a content management system.

Table 9.3: Dependency pair samples from the two datasets

Dependency type	ID	Description	ID	Description
Requires	1432952	add ability to associate saved billing address with payment card in add/edit card form	1429180	option to use new billing address when adding new payment card
	1394451	update illustration for error connection failure	1358293	ux error connection failure copy design and illustration update
	1524948	introduce session group to allow to manage multiple session at same time	1298912	multiple snapshot perform periodic session backup and let user restore particular backup
Relates_to	92822	ignore button for link targets	92297	make it possible to mark specific links to not get checked by linkvalidator
	92576	page tree filter: make it possible to explicitly filter by uid	36075	advanced filtering for the page-tree
	91496	differentiate between password reset "by user" and "by admin"	89513	provide password recovery for backend users

9.6.1 Firefox

In Bugzilla, feature requests are specific types of issues that are typically tagged as "enhancement" [116]. We retrieved these feature requests for the Firefox project using the search engine in the Bugzilla issue tracking system and exported all the related fields such as Title, Type, Priority, Product, Depends_on, and Blocks. Each issue report contains dependency relationships with other issue reports as references metadata [90]. Using this information, 3,773 depends_on (also interpreted as *Requires* dependency type) requirements pairs were retrieved. To generate negative samples, requirements that had no relationship were paired and 21,358 samples were generated.

9.6.2 Typo3

Redmine [5] is a free and open-source web-based management and issue tracking tool website. It allows users to manage multiple projects and associated projects. Various issues across a range of projects are updated each day

which helps software developers to track them for effective implementation. In Redmine, features are a specific type of issue that is extracted in this paper for further data analysis. Typo3 Content Management System (CMS) is an Open Source Enterprise Content Management System [10] with a large global community of approximately 900 members of the TYPO3 Association. We collected information such as issue_links, description, the version found, the version released, issue_id etc. for 5,017 features using Redmine's REST API through a Python script for this study.

All feature descriptions that had fewer than three words in them were filtered out, resulting in 1,324 feature pairs with dependency type *Relates_to*. Using the rest of the features that were not in any type of dependency with others, 9,270 pairs were generated as a negative sample set.

Table 9.3 mentions sample pairs of requirements dependencies. For example, to be able to associate the address with payment card *Requires* ability to use a new billing address when adding a new payment card. For both data sets, to perform binary classification, both positive and negative samples are needed for training. Since we only had dependent (positive) samples in the data, we generated negative samples by pairing the requirements which were not related in the given snapshot of the dataset.

9.6.3 Effort and Value Estimation

Typo3, currently at released version 11, is a complex content management system that is developed as a hybrid OSS software product. It has a core team of 12 members with varying skills and expertise. They have a major release cycle of 18 months and they plan two or more releases ahead of time. Developers are encouraged to track the dependencies in Jira, however, a few of the team members utilize post-its to work and track them. Typo3 does not explicitly consider Requirements Engineering as a development phase, but they term the efforts towards identifying features and extracting dependencies as conceptual work or scoping. Over 15% of the release, the cycle is identified as scoping effort and about 25% of scoping in a release cycle is identified as dependency extraction and identification. Nine team members and the CTO are involved, mostly in identifying the dependencies.

The CEO confirmed that about 80 % of the features are in some form of dependency with each other and missing the dependencies is more problematic than misidentifying them. As he puts this in words, "if you miss dependencies then it starts to ramp up quickly and this is when things go wrong, and breaks deadlines. we wanted to release in April (4 weeks ago) now deadline is mid October".

Typo3 identifies and manages seven different types of dependencies and their inversions such as precedes, blocks, clones, caused_by etc. Most of the dependency issues are identified rigorously through testing and the estimated re-work is about 12%. They have minimal manual testing as they have test suits of over 75,000 test cases. The CEO estimated that the overwork caused by missing dependencies is about 10% of the efforts. The average salary of the nine people involved in re-work is $70 (CAD). A summary of all estimates is provided in Table 9.5.

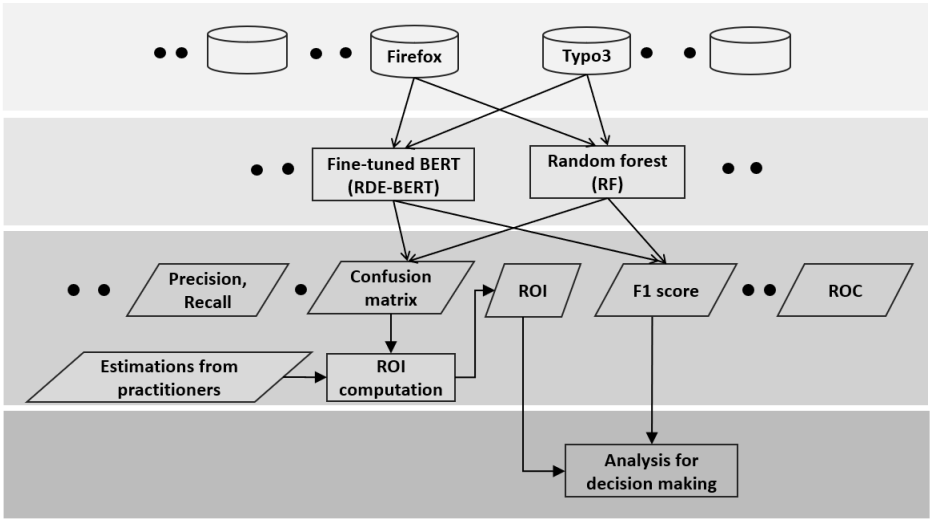


Figure 9.3: Overview of the experiment setup shows workflow and interactions between various components.

9.6.4 Experiment Setup

Figure 9.3 depicts the overview of our experiment setup. The complete approach is multi layered as highlighted in the shades. Each one of these could be further expanded to include additional elements for solution space evaluation further.

In this study, to generate the results, RF, Naive Bayes and SVM ML algorithms were compared against RDE-BERT for the two datasets: Firefox and Typo3. Overall eight experiments were conducted. Since RF performed better among all the conventional ML algorithms [56], we report the results of RF and RDE-BERT (i.e. totally four experiments).

For each experiment, we computed ROI using False Negative and False Positive values (from Confusion matrix). In Section 9.7 we present the insights to aid decision-making in algorithm selection based on these eight outcomes. For additional clarity, we list the names of the analysis of the results and their description in Table 9.4.

Table 9.4: Overview of the various analyses done in Section 9.7

		Description	
Fig 9.4	F1_Firefox	Firefox: Compare F1 of RF and RDE-BERT	RQ4.3 - (a)
Fig 9.5	F1_Typo3	Typo3: Compare F1 of RF and RDE-BERT	RQ4.3 - (a)
Fig 9.6	ROI_Firefox	Firefox: Compare ROI of RF and RDE-BERT	RQ4.3 - (a)
Fig 9.7	ROI_Typo3	Typo3: Compare ROI of RF and RDE-BERT	RQ4.3 - (a)
Fig 9.8	F1_ROI_RDE-BERT_Firefox	Firefox: F1 vs ROI of RDE-BERT	RQ4.3 - (b)
Fig 9.9	F1_ROI_RF_Firefox	Firefox: F1 vs ROI of RF	RQ4.3 - (b)
Fig 9.10	F1_ROI_RDE-BERT_Typo3	Typo3: F1 vs ROI of RDE-BERT	RQ4.3 - (b)
Fig 9.11	F1_ROI_RF_Typo3	Typo3: F1 vs ROI of RF	RQ4.3 - (b)

Requirements pairs were pre-processed to eliminate noise such as spatial characters and numbers. The generated output is fed to RDE-BERT and RF for training. Care was taken to process the same data snapshot through RF

and RDE-BERT models. Further, the fine-tuned BERT model (RDE-BERT) is then used for classification. The data was split (80:20) into train and test sets, and balanced between both classes.

In this empirical analysis, we conducted classification by utilizing a fraction of the whole dataset for training and testing for a small fixed data set. This was repeated by slowly increasing the training set and results were captured.

Random Forest: For RF, we use TF-IDF to generate word vectors before training. Also, hyper-parameter tuning was performed and the results for 10-fold cross-validation were computed, followed by testing.

RDE-BERT: For fine-tuning BERT, a pre-trained BERT model is used in combination with our RDE specific dataset. The result is a fine-tuning BERT model called RDE-BERT. To fine-tune the BERT model, we used *NextSentencePrediction*[2], a sentence pair classification pre-trained BERT model, and further fine-tuned it for the RDA specific dataset on Tesla K80 GPU on Google Colab[3].

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

BERT eliminates the need for feature extraction since it is a language model based on deep learning. BERT, pre-trained on a large text corpus, can be fine-tuned on specific tasks by providing only a small amount of domain-specific data.

9.7 Empirical Analysis - RQ4.3

In this section, we report the results of our empirical analysis and answer RQ4.3. We structure results by the type of decisions to be made: (a) When comparing two techniques: Which one is preferable under conditions selected?, and (b) When looking at one technique, when to stop the analysis? For both decisions, we present the results of the analysis for the two data sets introduced above and the two techniques under investigation using estimates from Table 9.5.

9.7.1 Comparison between RDE-BERT and RF - (a)

The traditional approach for comparing techniques is to look at just accuracy for some fixed training set. Figures 9.4 (F1_Firefox) and 9.5 (F1_Typo3) show the comparison of the F1-scores for varying training set sizes for the two datasets. Results show that RF achieves a higher accuracy more quickly for even small-sized train sets respectively. However, with a training set greater than 40% of the dataset for Firefox and 30% for Typo3, RDE-BERT achieves better results overall.

[2]https://huggingface.co/transformers/model_doc/bert.html#bertfornextsentenceprediction
[3]https://colab.research.google.com/

Table 9.5: Parameter settings for the two empirical analysis scenarios

Parameters	Values
Phase B: $(C_{dg}+C_{pp}+C_l)$[1]	1.5 min/sample
Phase C: $C_{train/test}$	0.30 min/sample
C_{HR}	\$70/hr
N_{HR}	10
N	Firefox:7,546 Typo3: 2,648
$Cost_{FN}$	\$25,000
$Cost_{FP}$	\$10,000
$Value_{prod}$	\$4,000,000

[1] C_{dg}, C_{pp} and C_l are weighed equally (= 0.5min/sample) each. Also ratio of Phase B:Phase C = 80:20 has been considered

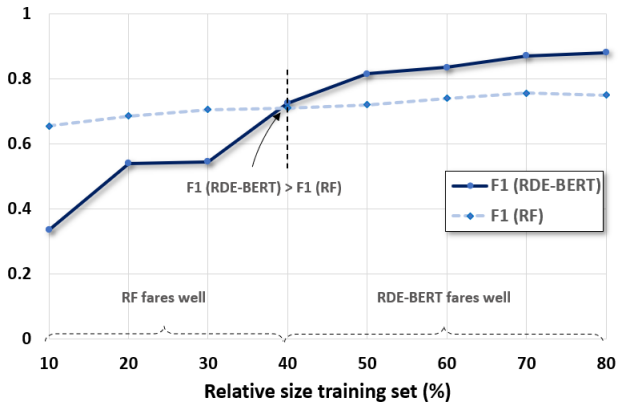


Figure 9.4: F1 of RDE-BERT vs RF for Firefox dataset shows RF achieves higher F1-score for smallest train set unlike RDC-BERT, however, with increasing train set, RDC-BERT outperforms RF

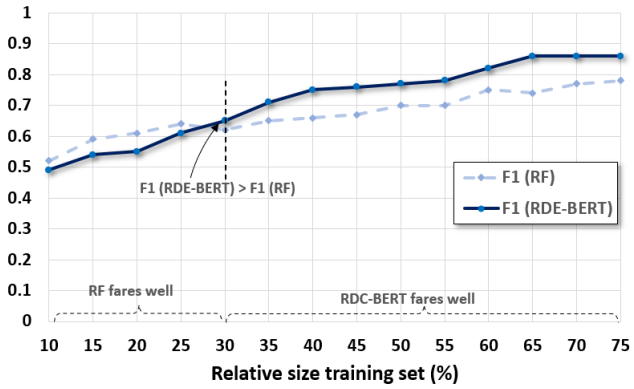


Figure 9.5: F1 of RDE-BERT vs RF for Typo3 dataset shows RF performs well with smaller train set. However, eventually, RDC-BERT excels with increasing train set gradually

Comparison of ROI for the two datasets and two methods (RDE-BERT and RF) is shown in Figures 9.6 (ROI_Firefox) and 9.7 (ROI_Typo3) respectively. For Firefox, with a smaller-sized train set, RF once again performs better comparatively, even though the ROI is negative. Similar results are evident for Typo3. RF performs marginally better ROI-wise for the smaller training set. ROI of RDE-BERT picks up pace only beyond 40% and 30% train set for Firefox and Typo3, respectively.

9.7.2 Bi-criterion Analysis of RDE-BERT and RF- (b)

In the second part of the analysis for RQ4.3, we look at one technique at a time from the perspective of both F1-score and ROI. This will support decision-making towards the question of when does increase accuracy no longer pays off?

As illustrated in figures 9.4 and 9.5, increased training set does not yield better F1-score beyond 65%. The

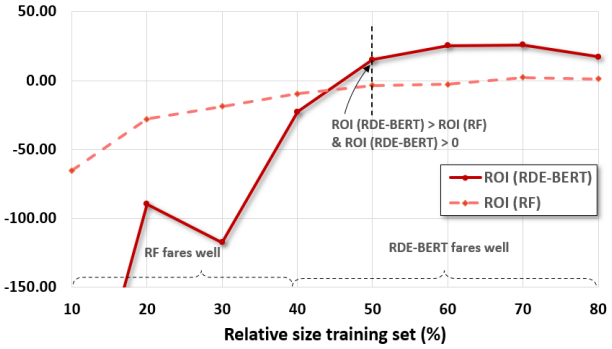


Figure 9.6: ROI of RDE-BERT vs RF for Firefox dataset shows that RDC-BERT generated positive ROI when 50% more train set is made available

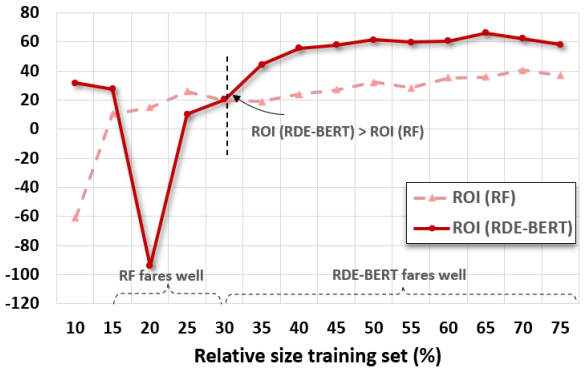


Figure 9.7: ROI of RDE-BERT vs RF for Typo3 dataset shows similar results as RF performs better than RDC-BERT when train set is smaller

F1-score hits a plateau and even starts to degrade for both of the methods and datasets.

However, if we look at the trade-off between the F1 and ROI for both datasets, the results become interesting. Figures 9.8: F1_ROI_RDE-BERT_Firefox show that for RDE-BERT, F1-score increases linearly, however, max ROI is achieved when the train set is 70% of the dataset. Whereas, for RF, in Figure 9.9 : F1_ROI_RF_Firefox shows that F1 and ROI for the train set lower than 40% is better than that of RDE-BERT. Chasing for a higher F1 score does not payoff and one needs to take a closer look at the benefits vs investment in more training data, eventually.

For Typo3, in Figure 9.10: F1_ROI_RDE-BERT_Typo3 shows that F1-score and ROI grow steeply for RDE-BERT with the increasing train set. However, similar to Firefox, ROI and F1 of RF are stable and better than RDE-BERT for the train set smaller than 30%. These findings once again emphasize the need to relook at how F1 and ROI together could aid in deciding on the ML selection.

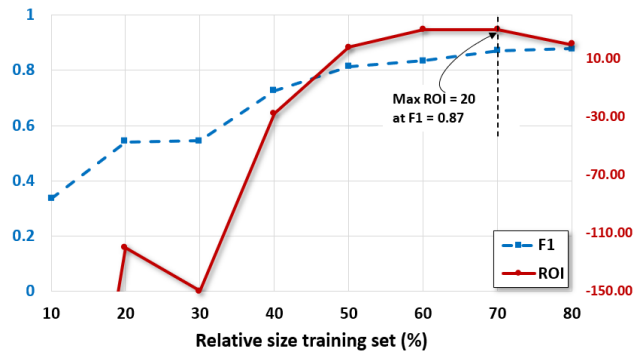


Figure 9.8: F1 vs ROI of RDE-BERT for Firefox dataset, utilizing values from Table 9.5

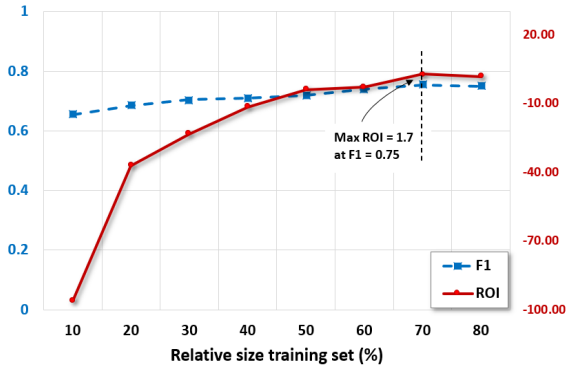


Figure 9.9: F1 vs ROI of RF for Firefox dataset, utilizing values from Table 9.5

In both datasets studies, it is evident that RDE-BERT models require large amounts of data (at least 30% or more) to stabilize and show value (steady positive ROI). When comparing RDE-BERT with RF using ROI criteria (Fig 8 and 9) across the two data sets, RF outperforms RDE-BERT for the lower train set (incurring lower negative

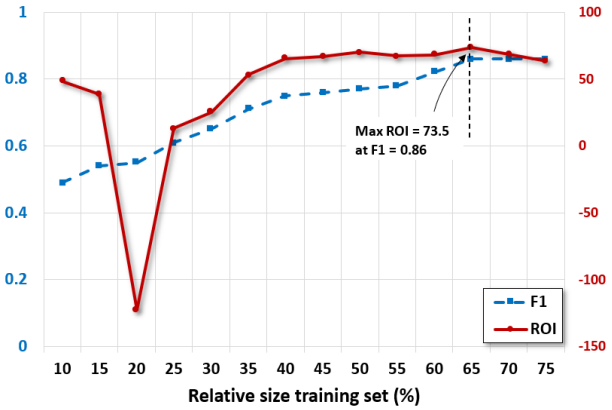


Figure 9.10: F1 vs ROI of RDE-BERT for Typo3 dataset, utilizing values from Table 9.5

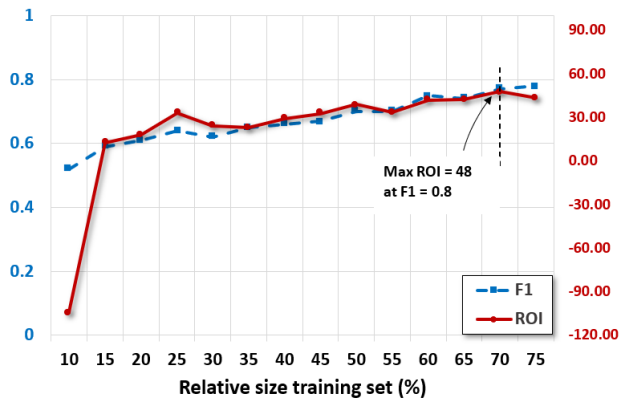


Figure 9.11: F1 vs ROI of RF for Typo3 dataset, utilizing values from Table 9.5

returns). However, positive ROIs are observed only at the larger train set at which RDE-BERT is consistently better than RF. Based upon the Firefox findings (Fig 10 and 11), RDE-BERT approaches the 80% benchmark accuracy with approximately 50% of the training data while RF requires 70% training data to attain the same level of accuracy. However, both techniques can achieve positive ROI with as little 50% training data but RBC-BERT achieves maximum ROI (30) with an accuracy of 0.87 with approximately 70% training data, while RF achieves maximum ROI (2.2) with an accuracy of 0.75 with approximately 70% training data.

Based upon the Typo3 findings (Fig 12 and 13), RDE-BERT approaches the 80% benchmark accuracy with approximately 55% of the training data while RF requires 70% training data to reach the same level of accuracy. However, both RBC-BERT and RF can achieve positive ROI with as little 15% training data, but RBC-BERT achieves maximum ROI (73.5) with an accuracy of 0.86 with approximately 65% training data, while RF achieves maximum ROI (48) with an accuracy of 0.80 with approximately 70% training data. Thus, RBC-BERT can deliver much higher ROI and similar levels of accuracy than RF given approximately the same amount of training data.

Finally, the parameter settings that seeded the initial model (Table 5) were based upon industry estimates, which were possible were verified by senior management in the respective firms. However, some of the findings may be sensitive to these initial conditions. Thus, these would need to be set for the specific context upon which the data sets are based. This is also the basis upon which scenario analysis could be conducted to evaluate the worst case, best case and most likely initial conditions to evaluate the impacts on subsequent decisions.

9.8 Discussion

9.8.1 Implications

ML is not simply a cost of doing business, rather it is a foundational activity that can provide value for the money invested. Our proposed approach aligns this notion with the strategic direction of the organization. While return on investment (ROI) is a common approach used for business planning and decision making, it is not applied as widely

within software engineering or specifically within applied ML.

In our study, we demonstrate how to instantiate ROI in the context of RDE. Our approach provides a pragmatic link between the business and technical aspects of the organization by providing a common language that incorporates both the technical aspects inherent in the evaluation of accuracy, with the business considerations of costs and benefits. We argue that this is an extremely powerful approach that provides evidence that is compelling and consistent for both technical and business decision-making.

In addition, we think that the ROI approach could sensitize the ML team to the entire process of ML classification and how that process fits into organizational processes. The ROI approach is essential for evaluating the possible tradeoffs between accuracy and the benefits. Mainly because without consideration of the key dependencies within the process, benefits in one part of the process (e.g., improved accuracy) can easily be undermined by excessive costs in another part of the process that would not typically be considered if focused exclusively on accuracy. Alternatively, lower levels of accuracy in the ML process might be acceptable if other benefits are accruing at reasonable costs. Thus, valuable ML investments are potentially being avoided based upon not meeting accuracy expectations, when those ML solutions could be sufficient to realize high payoffs for the organization.

Our approach increases the transparency of the decision-making process by adding diversity to the evaluation criteria that foreground the various tradeoffs being made. The development of AI tools that businesses and consumers can trust is essential for their continued adoption, especially as there is increasing regulatory scrutiny of the biases that arise in the ML algorithms or inherent in the data used for training.

While ML algorithms are generally trusted for relatively mechanical well-defined problems, this trust plummets when the decisions are subjective, and likely to vary by contextual variables that are not well understood. This in turn increases the pressure to adjust ML algorithms for variations in specific markets further driving up development costs. Such pressure directs the focus on customizing products and services based upon ML algorithms for specific markets while increasing costs further and undermines the benefits for certain markets or customers [74]. The proposed approach considers technical and business aspects simultaneously and provides a more traceable set of interconnected processes. This approach includes business and technical considerations to enable management to evaluate the risks of some undesirable decisions and the tradeoffs needed to realize the likely benefits.

9.8.2 Limitations

We have explored RF and RDE-BERT in the context of the RDE problem and presented our results. Since there is no single method which could work for any given problem, comparison of multiple approaches and their results remains out of the scope of this study.

Another threat to validity is the related to the conclusions made. Although we have taken care to randomize the data by shuffling and used stratified split to take care of balanced data in both training and validation, multiple runs with varying first iteration data sample are needed to be more confident on the conclusions made. However, we argue that the key observations made are valid from the restricted empirical validation performed.

9.9 Discussion

ML classification is widely used in many disciplines of Science and Engineering. In this study, we demonstrate that just looking at performance measures such as accuracy could be misleading when, for example, deciding between two ML techniques evaluated for solving the same problem. Conversely, ignoring the cost and benefit of such a classification could cause the risk of unprecedented emphasis on improving accuracy that might not generate any value for the additional efforts spent. Additionally, in this research, we also provide a high-level ML process for classification (supervised machine learning). However, with minuscule changes, this process can be adapted to unsupervised ML methods easily.

We proposed to complement Data Analytics of empirical studies with ROI analysis to avoid over analyzing data in this work. To validate the need, we performed an analysis of accepted papers of ESEM conferences between 2015 and 2019 and found that 51 out of 190 papers (27%) were addressing some form of DA. Among them, 39% included some consideration of cost, value, or benefit. However, none of them directly explored or discussed ROI or used cost-benefit analysis to decide the degree of DA needed. From a decision-making perspective, selecting one out of many techniques, and for a selected technique, deciding the termination of analysis amount to enlarge the scope from one to two criteria.

Beyond accuracy, reflecting the benefit, it is essential to look into the investment as well. Exclusively looking into the different aspects of accuracy is cardinal, but it does not provide a full picture as the effort consumption and impact are ignored. Effort estimation is well studied, however; prediction of value [28] has not been explored as much. Even rough estimates may be helpful to decide how much further investment into DA is reasonable. To make this agenda successful, economical, business, and social concepts need to be taken into account, apart from just the technical aspects.

Acknowledgement We would like to thank graduate students Saipreetham Chakka and Aris Aristorenas for their assistance in generating results and conducting literature review for this study.

Summary

As shown in Figure 9.12, results from previous evaluations: Active Learning, Transfer Learning and Fine-tuned BERT pave way to the ROI analysis aspect for RDE due to contrasting needs for training data and their eventual performance variations. Thus, we proposed two contextual frameworks: ML process model and ROI modeling to derive ROI values for different empirical studies. Results showed that Accuracy is not enough to weight the performance of ML method and additional criteria such as ROI needs to be considered for effective decision making.

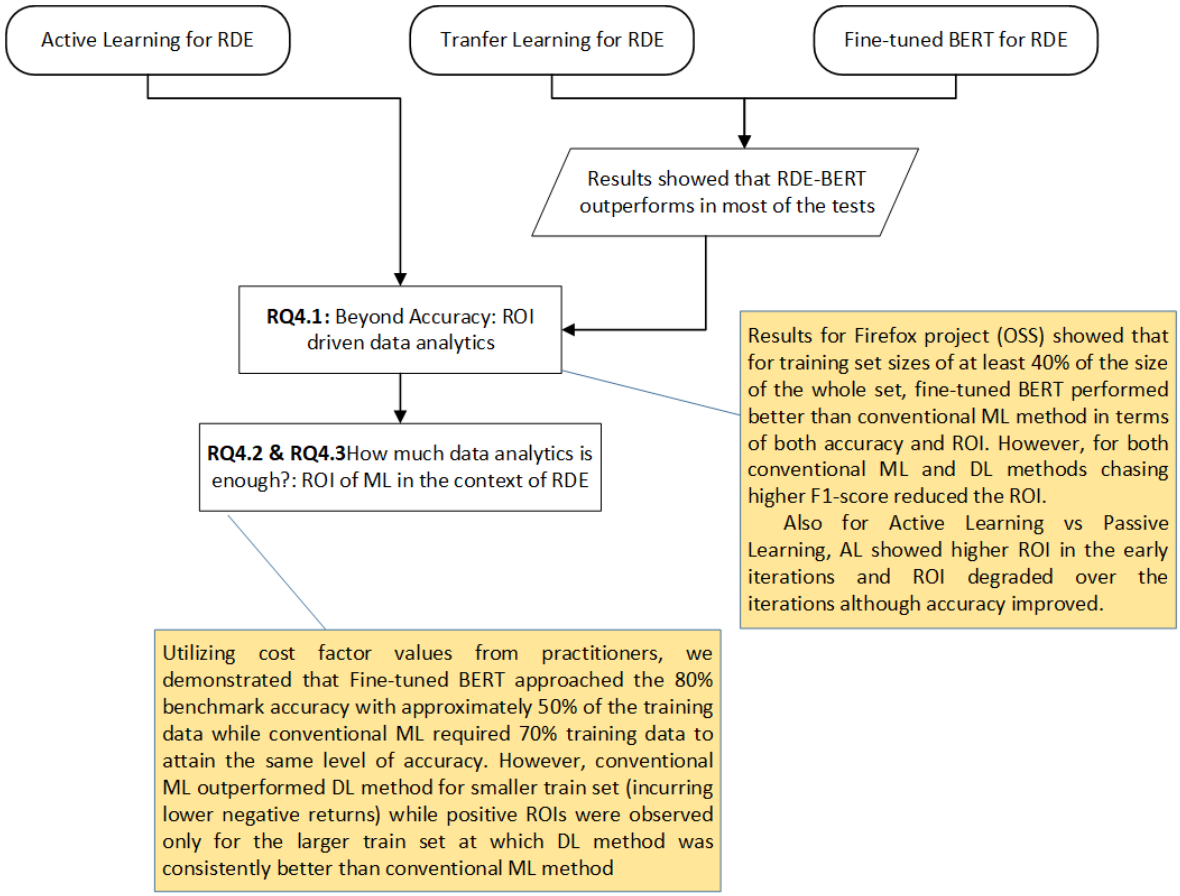


Figure 9.12: Bird eye view of the research questions, their logical connection and the briefly explained results

Part V

SReYantra: An RDE Toolbox

Chapter 10

Toolbox: SReYantra, an RDE and ROI Analysis System

10.1 Introduction

SReYantra is a comprehensive tool that provides conventional and advanced ML approaches based solutions for RDE and their ROI analysis. It infers underlying dependency information based on NLP. It utilizes advanced ML techniques such as Active Learning, Transfer Learning and DL language model: BERT (Bidirectional Encoder Representations from Transformers) as its various components for automating dependency extraction. It also provides a mechanism to compute the ROI of ML algorithms to present a clear picture of trade-offs between cost and benefits.

The term *Yantra* in `SReYantra`[1], means a machine or a systematic broadly applicable "system, method, instrument, technique or practice". Since we propose one such method for research related to **S**oftware **Re**quirements, hence the name `SReYantra`. *SReYantra* has four main RDE specific modules as follows. Of these four modules, proof of concept (POC) of the first three has already been developed. For now these POCs provide a command-line-based interface only (for now), they can be transformed into graphical interface utilities with additional effort.

- Weakly Supervised Learning and Active Learning for effective data acquisition.
- DL and conventional ML-based Transfer Learning utilize annotated data of a project for training and then use it to predict dependencies for another project.
- DL method to overcome feature extraction challenge.
- ROI analysis to find the trade-offs between various approaches.

This work is in progress, and we envision consolidating the already implemented components into one holistic tool and further developing the ROI analysis utility to complete the tool implementation. In this chapter, we describe

[1] https://en.wikipedia.org/wiki/Yantra

architectural components, workflow and user interface of *SReYantra* in detail.

Architecture: Figure 10.1 shows the architectural components of *SReYantra*.

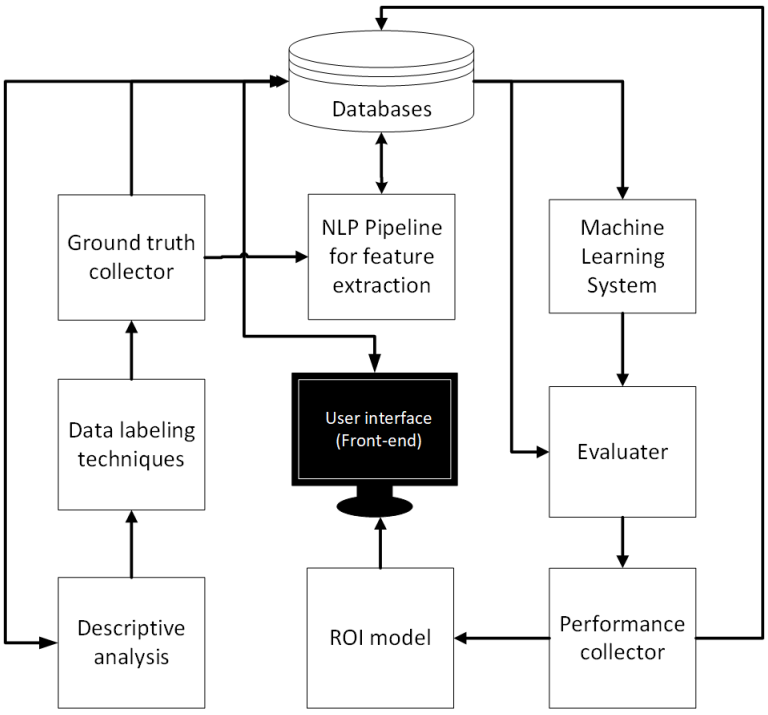


Figure 10.1: Architectural components of *SReYantra* and their interactions

Descriptive analysis: As the first step to any ML process, the first step is to analyze the data and gather important statistics such as the number of samples, their distribution and sentence length. These are essential for considering the data used in ML training in the later stages.

Ground truth collector: In order to use ML for automation, there is a need for a training dataset which is essentially annotated data generated by observation by domain experts. In the absence of such data, we utilize two methods (developed and explained in Chapter 2 & 3).

Data labeling techniques: *SReYantra* offers two data labeling techniques: Weakly Supervised Learning (WSL): used in the absence of domain expert and Active Learning: a human in the loop annotation gathering technique to query the most uncertain labels with the help of the domain expert, as two options for annotated data generation. While Active Learning needs an oracle to annotate the data, WSL uses the joint agreement on the label by two or machine learners to achieve such annotations. This component can be easily extended to other co-testing strategies.

Databases: Since datasets undergo transformations in different stages, we use stand-alone databases such as MongoDB or MySQL to store it. All the components interact with the database to read from, modify (if needed) and write back its contents.

NLP pipeline: Most of the data is collected in raw form and needs to be pre-processed before modeling. We

have utilized special character numbers, symbols, and punctuation removal followed by stop words removal and lemmatization before using the Bag-of-words feature extraction method paired with TF-IDF for our empirical studies. However, this can be easily extended to other techniques such as word2vec and word embedding in the future.

Machine Learning System: Since we are utilizing supervised and WSL and AL techniques in our research, we have implemented classification and prediction. This component includes training based on the selection of the ML method. We currently offer conventional ML methods such as NB, RF, SVM, and the ensemble of these three ML algorithms and fine-tuned BERT methods for RDE. This component can be easily extended for additional methods employing development and implementation.

Evaluation: Evaluation of ML methods is performed using various measures explained in Section 1.3 (Evaluation measures). We conduct 10×10 cross-validation and take standard deviation into account. Also, for testing (on held-out/unseen dataset), we repeat the testing 10 times to finally capture the averaged results at the end.

Performance collector: This component performs housekeeping operations such as acquiring and tracking the details for various ML methods and writing the evaluation data back to the database. This component also passes the data to the ROI modeling component further.

ROI modeling Utilizing various cost factors derived from decision makers' experience, we model ROI using the approach explained in Section 9.5. This component is the crux of this complete application as it provides a mechanism to change multiple parameter values such as cost of FP, FN and value of the product to generate separate reports for users/decision-makers to aid algorithm selection.

User interface This is the front-end of this tool which provides a graphical interface for end-users to interact with the system. Datasets such as raw train and test uploading, various inputs at different stages of tool usage are gathered through it. This user interface also provides a dashboard-like utility to view various charts and measures generated by ROI modeling and Performance collector components.

In the next section, workflow of the *SReYantra* tool is elaborated.

10.2 Workflow

Based on the dataset size, *SReYantra* lets the user choose appropriate settings at every instance to extract dependencies. Firstly, train set size is used to decide the next approach. If the train set size is below some threshold (that can be set apriori), the user can choose data acquisitions methods first. Once again, the user can choose ML methods to proceed further. *SReYantra* also allows choosing multiple ML approaches as their performance can be compared for decision making after evaluation. The AL-based method needs additional user inputs (elaborated in Section 3.3.2) such as choice of uncertainty sampling technique, confidence level threshold, termination criteria, number of annotations per iteration etc. This workflow considers the scenario of cross-project dependency prediction and deep learning

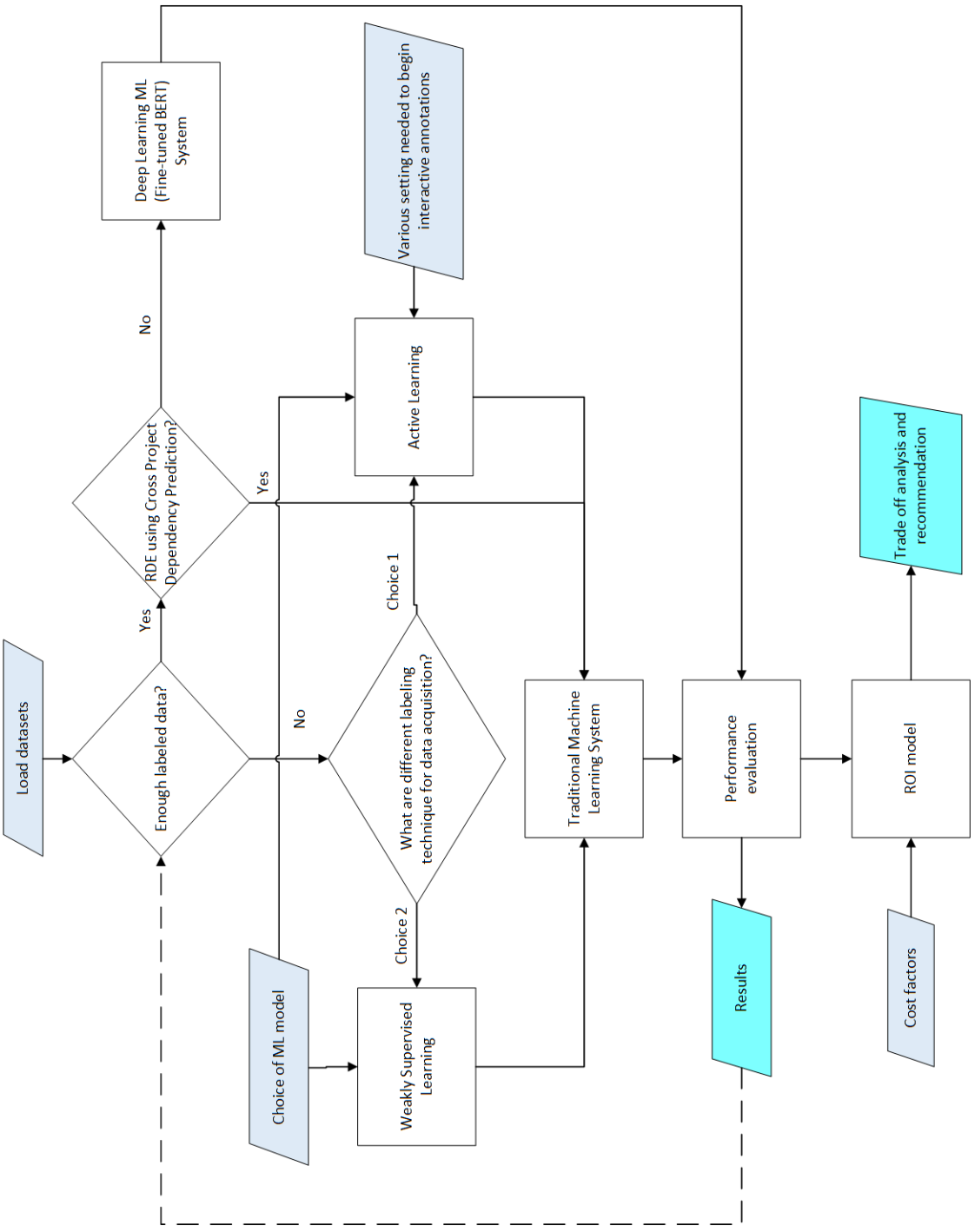


Figure 10.2: *SReYantra*'s workflow diagram. all the components thus far developed will be integrated to create a logical flow

ML methods once data is acquired, thus providing an interactive ML evaluation platform for effective analysis and decision making. Once the ML performance evaluations are captured, *SReYantra* performs ROI modeling (explained in Section 9.5). Based on the various values input for various cost factors, different results are generated, which are then displayed in the form of charts in the dashboard like a user interface for the end-user. *SReYantra* is also provided with an interactive interface for changing inputs and results visualization, which will be described in the next sections with wire-frames of *SReYantra*

10.3 User Interface

The user interface will have multiple pages to interact with *SReYantra* and perform RDE. Wire-frames of this tool is shown in a series of images as follows. Figure 10.3 shows the first page of user interaction which facilitates raw

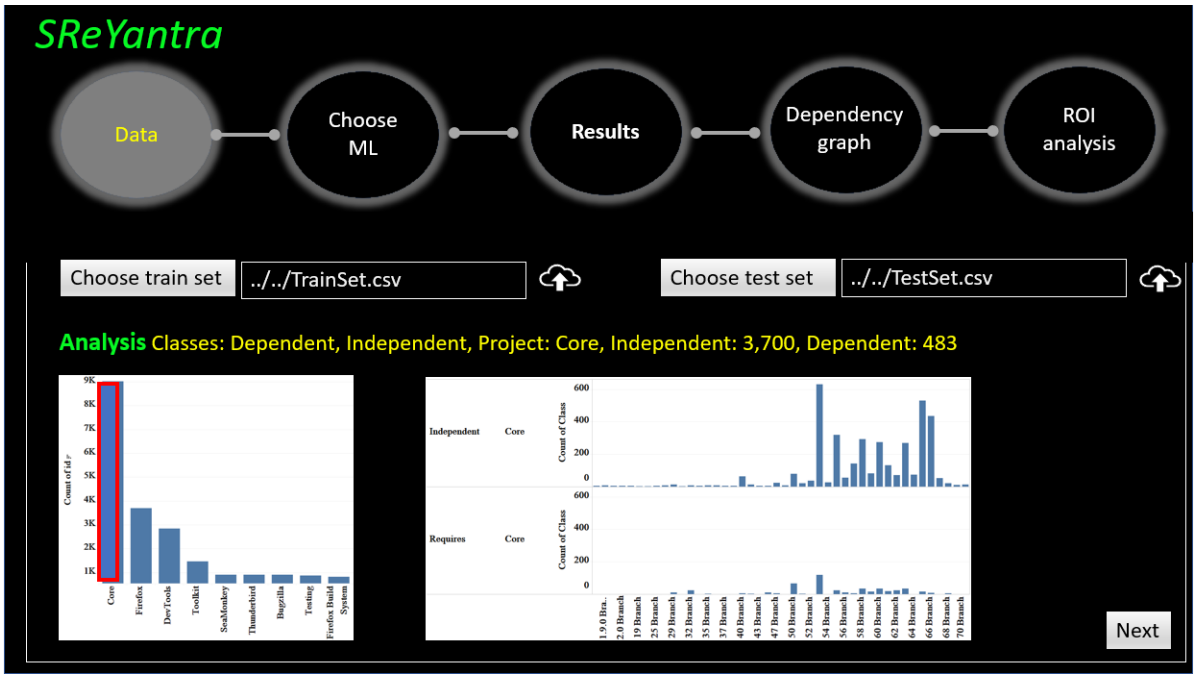


Figure 10.3: Datasets are uploaded through the interface, which is validated at the backend. Data is then stored in the database, and descriptive statistics of the dataset are then displayed through various interactive charts.

data upload and its statistical visual overview. Figure 10.4 an interactive page is shown to the user if the annotated data samples are fewer than 100 by default (provision will be made to let the user alter this value). According to various choices made through this page, respective ML methods are executed, and results are accumulated. Figure 10.5 shows the results page. This interface provides interactive visual representations of the chosen ML methods. F1-score, Precision, Recall charts and box plots are displayed. This interface can be extended to include other measures in the future. At every stage snapshot of the dataset is maintained in the system, which enables re-execution of the experiments with varying settings and combinations. This tool could benefit decision-makers to get the projections

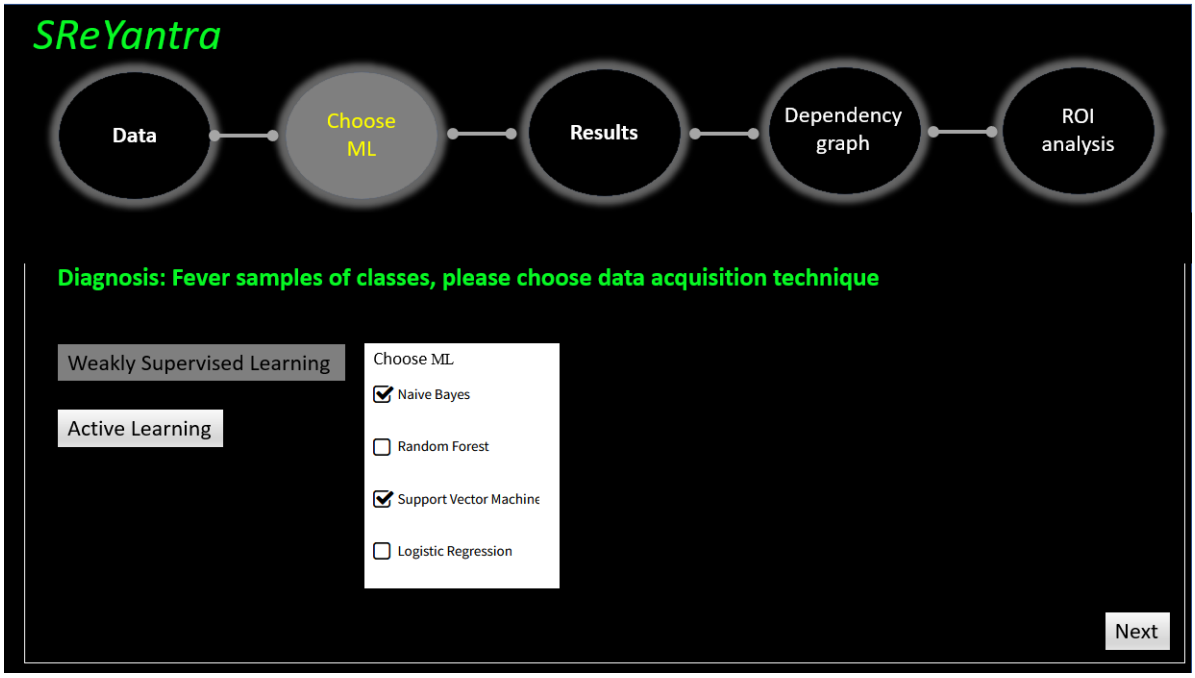


Figure 10.4: Based on the workflow described in Figure 10.2, user is allowed to choose the data acquisition method based and asked with additional information based on the selection made

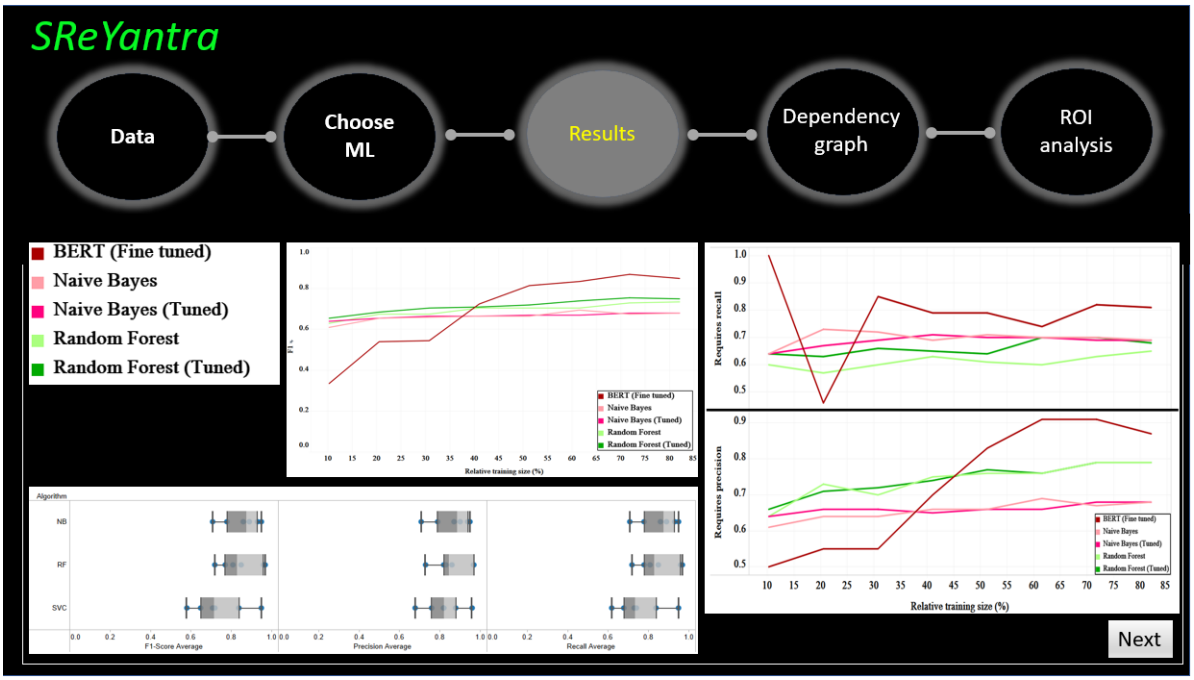


Figure 10.5: Utilizing various settings and the dataset, ML training and testing stages are executed in the backend to generate the results. Various charts and plots summarizing the outcome are displayed on this Results page.

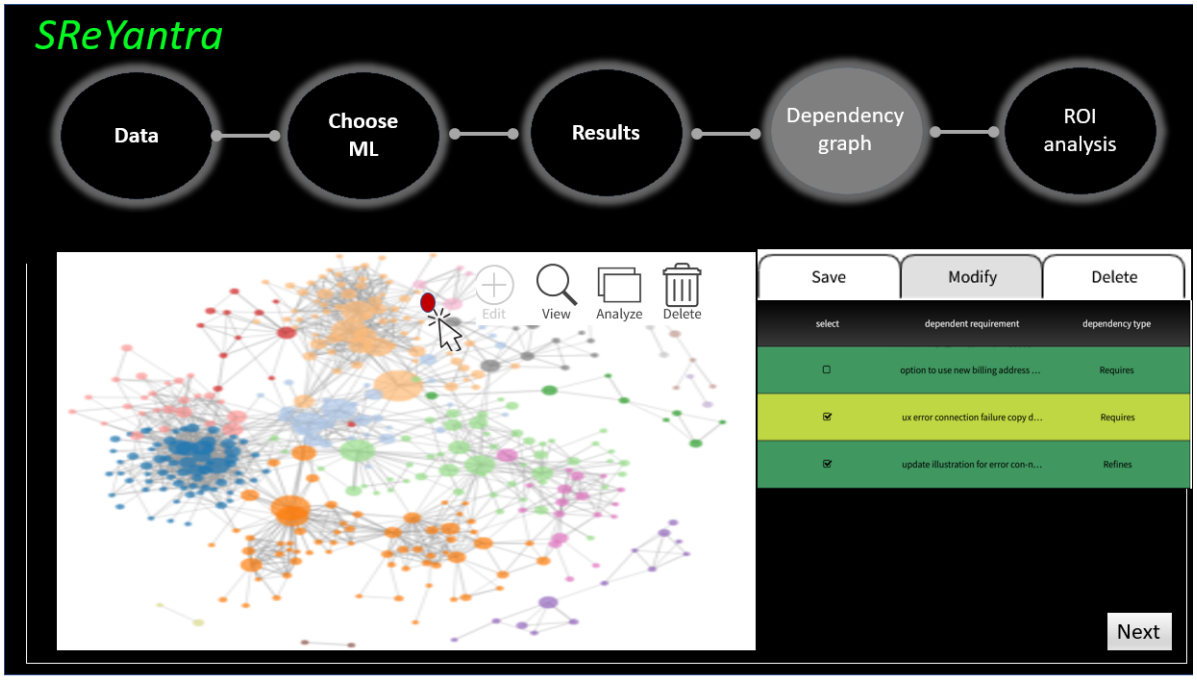


Figure 10.6: Trained model is then used to predict the dependencies for unlabeled data, and the graphical interface is used to show the dependency network. This interface is made interactive, allowing modification to dependencies to correct or amend incorrect dependencies (if any) by the domain experts.

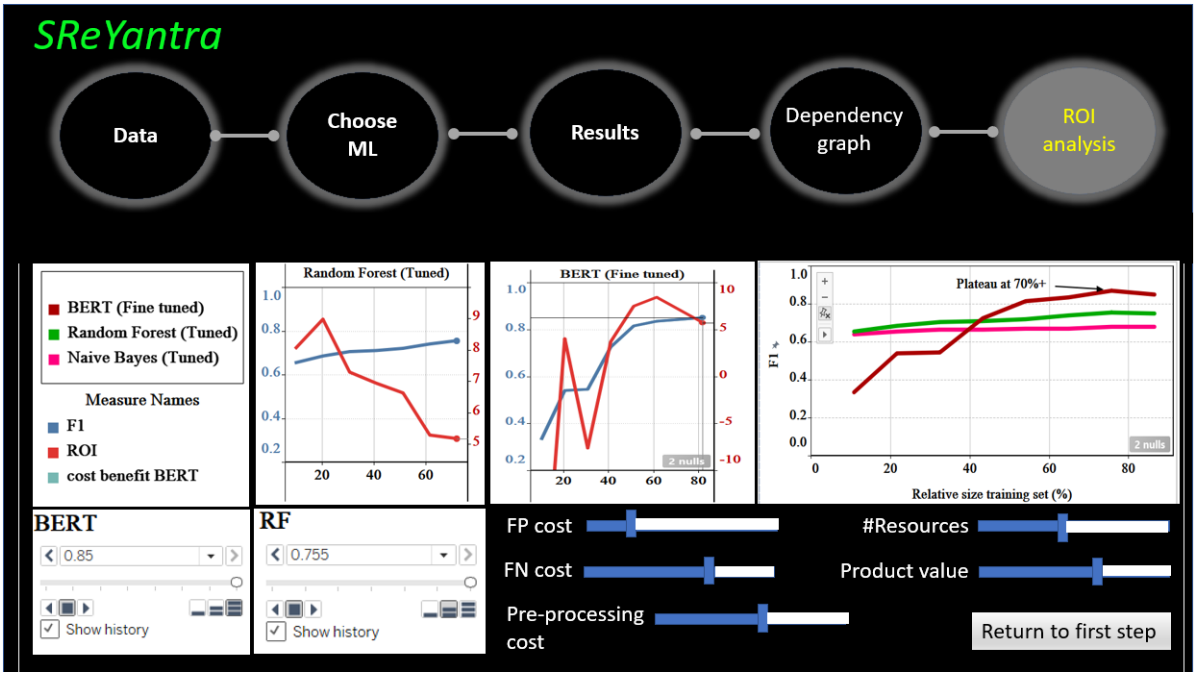


Figure 10.7: ROI Analysis page provides an interactive interface for the end-user to perform trade-off analysis by varying cost factors. This interface provides a provision to the decision-makers to weight accuracy in terms of benefit based on the investments made all along the ML process

of multiple methods in terms of business value and subsequent benefits, which could be compared and analyzed beforehand.

10.4 Conclusion

The *SReYantra* tool provides a holistic interface to address the RDE problem using multiple ML-based approaches and its ROI analysis for effective decision making. Specifically, it will provide various functionalities as follows.

- A ML-based software that can automate RDE while using various inputs from the end-user in every instance.
- Provides a method to compare and evaluate the approaches not only using ML performance measures but also based on ROI.
- Allows end user to make changes to cost factors and view the projections of changing ROI for decision making.
- Provides a dashboard-like interface to interactively vary the cost factor settings to visualize the results quickly
- Provides scope for tool extension as additional algorithms can be developed and added as modules in the future.
- Once RDE is performed, the dependencies network can be shown in a graphical interface for better understanding. This interface can be enhanced to allow end-users to validate and modify the dependencies. In this way, *SReYantra* could also be used as an interface to validate the dependencies and repeat the RDE operations in iterations to steer it in the direction of the end user's vision and interest.

Chapter 11

Main Contributions and Future Work

11.1 Revisiting the Research Questions

In this thesis, I have attempted addressing the challenges of Requirements Dependency Extraction (RDE) automation using various advanced machine learning approaches. These challenges were synthesized from our three-pronged approach to problem identification, namely, 1) Literature analysis, 2) State-of-the-practice survey with the practitioners, and 3) Preliminary study for RDE automation using a public dataset. Literature analysis pointed out that identifying requirements dependency is cognitively difficult, and domain experts' time is crucial in this activity. Moreover, such RDE is still an open question, and the application of ML and NLP explorations are being explored in RE in general. Further state-of-practice survey affirmed the expansiveness of the problem in the real world. It directed us towards our research questions formulations and empirical evaluations using public, OSS and industry datasets. In this section, I summarize my research contributions to the body of knowledge.

RQ1: The first research question aimed to evaluate if Weakly Supervised Learning and Active Learning are effective at addressing data acquisition challenges of RDE automation. Weakly Supervised Learning labels a unlabeled sample if the majority of the classifiers agree on the labeling. Whereas Active Learning overcomes the labeling bottleneck by asking queries in the form of unlabeled instances to be labeled by an oracle (e.g., a human annotator). Empirical evaluation of these two methods on public and open-source software datasets showed encouraging results. Further comparison of Active Learning with Ontology Based-retrieval and a hybrid model showed that the human efforts in RDE could be reduced by 50% for the two industry datasets: Blackline Safety, Canada and Siemens, Austria, evaluation.

RQ2: The second aim of this research was to evaluate the Transfer Learning approach for RDE in the advent of limited training data. Transfer Learning transfers dependency extraction knowledge learned from source projects to the target project. We evaluated the conventional ML and Deep Learning (DL) method (fine-tuned BERT) for Transfer Learning in RDE. Empirical analysis of six Mozilla family projects using conventional

ML showed pessimistic results for multi-class classification. However, fine-tuned BERT (DL approach) showed Transfer Learning could yield 27% to 50% better performance comparatively for the F1-score measure.

RQ3: The analysis of the DL approach as part of the third research question was targeted at the feature extraction challenge of RDE automation. Feature extraction transforms the raw text into suitable internal numerical representations, i.e. feature vector. Recent cutting edge DL-based method: Bidirectional Encoder Representations from Transformers (BERT) is a pre-trained language model that has been successful at various NLP tasks such as Question Answering and Natural Language Inference. BERT overcomes the need for feature extraction as part of text pre-processing and understands the relationship between a pair of sentences. Hence, we tuned BERT further using RDE specific data and evaluated six OSS projects. Results were outstanding as fine-tuned BERT outperformed conventional ML methods by 13% to 27% for the F1-score measure and successfully identified the dependency direction for ordered dependent requirement pairs.

RQ4: Study of the last research question provides deeper insight into ML approach selection for RDE. Essentially, this research question evaluated the benefits of using ROI as an additional criterion for ML classification. Although there is a wide spectrum of ML methods to choose from for RDE, choice of ML is tedious when cost and benefit factors are considered. Complex DL approaches need larger training data; however, in what circumstances they could be preferred over simple algorithms when data size and effort investments are considered are explored in this research. We propose an nine-stage ML process and a novel ROI modeling approach while evaluating this research question. Through the three empirical evaluations on three open-source software: Firefox, Redmine and Typo3 using practitioners inputs for cost factors, I demonstrated that focusing on improving accuracy using advanced ML methods might not generate value for the additional effort spent in data accumulation and pre-processing

Figure 11.1 shows the relationship between the four research questions and their overlapping sections with the others.

11.2 Future Work

In this research, we have used advanced ML approaches such as Weakly Supervised Learning, Active Learning, Transfer Learning, and Deep Learning for RDE. Despite extensive evaluations, there remain various directions for future work, listed as follows.

- **WSL for RDE:** WSL preliminary study showed that the quality of the textual content describing requirements is one of the key performance factors of our approach. Also, the overall use of this approach in different domains needs thorough evaluation, which is part of the future research agenda.
- **AL for RDE:** In the future, we will evaluate the impact of the size and completeness of the ontology on the RDE. We expect a trade-off between the effort invested into ontology creation and the benefit generated; therefore, the final effort invested in the ontology may depend on context factors of the organizations and

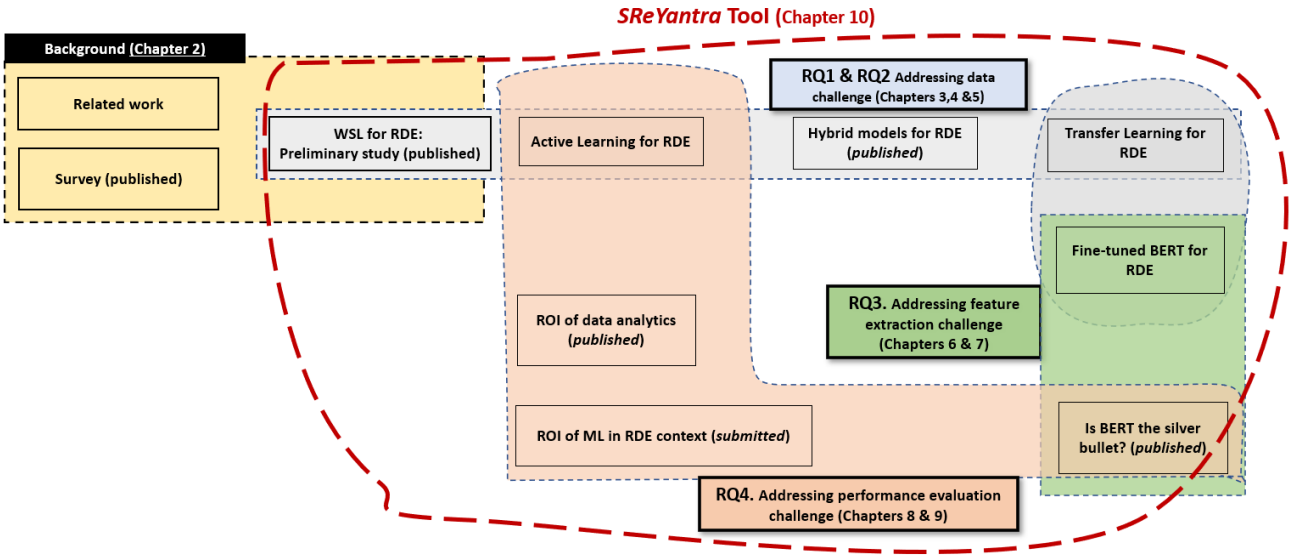


Figure 11.1: Overall organization: Research Questions (RQ), their overlapping connections, and corresponding chapters of my thesis. *SReYantra* is the outcome of this research which attempts to address four challenges of RDE automation.

their projects. Existing approaches for ontology construction in RE [126] could be customized to this cost-effectiveness objective using some estimation model for ontology engineering [148]. In another direction, we plan to invest in tuning AL further to find out the classifier's optimized performance based on the characteristics of the problem.

- **DL for RDE:** With rapidly evolving NLP research, we anticipate the latest solutions such as XLNet [84] to have similar evaluation outcomes. However, we envision conducting these evaluations as part of our future work.

- **ROI analysis:** We use RDE as a sandbox to build a proof of the concept for ROI analysis for ML selection. In the future, we will extend the results in various dimensions. The concepts of this research will be applied and evaluated for problems from other domains. However, the challenge is to project the benefit of achieving better accuracy results and estimating the total effort of data analysis. Also, depending on the problem, we will investigate other ML methods and additional data sets.

- **Alternative hybrid methods:** The ML and NLP research fields are rapidly evolving as cutting-edge research is bringing efficient methods to the fore. Similarly, there are various ways to generate hybrid models out of the most effective ones, such as AL and DL methods which showed stellar results in my research. How we can combine the best of these two methods to generate a hybrid method for RDE remains as part of our future work.

- **Hierarchical and group of requirements:** Requirements are not always described at the fine granular level. However, our solutions can be extended to analyze hierarchical or group of requirement with modifications. This will be explored as part of the future work.

- **Dependency aware test prioritization:** Another direction for RDE research would be to evaluate how requirements dependency aware test prioritization would impact testing in an incremental and iterative software development process.
- **Recommendation is not enough - Explainable AI for RDE:** In the advent of the rise in explainable AI and its applications in Software Engineering [13], it remains open research in the context of RDE. It is important to understand why certain pairs are classified or misclassified as dependent. We envision to do this thorough analysis as part of the future work. Such further systematic analysis can help in identifying reasons for false negative and false positives.
- **Evaluation measures:** In this research, we have considered F1-score, precision and recall as ML performance evaluation measures. However, others such as AUC and ROC need to be looked at for RDE. We envision considering these measures in future work.
- **Toolbox:** Current research work presented in this thesis lays the groundwork for a more extensive toolbox envisioned to address all the RDE specific challenges. The list of items presented in this section is envisioned as part of a toolbox presented in Chapter 10, which provides a high-level design and other details.

Bibliography

[1] Apache OpenNLP Toolkit. Available at: http://opennlp.apache.org. Accessed 2020-02-04. [Online].

[2] Bugzilla: bug-tracking system. https://www.bugzilla.org/. April, 2020.

[3] Bugzilla: product information. https://bugzilla.mozilla.org/describecomponents.cgi?products. Accessed: September 2019.

[4] European rail traffic management system. https://en.wikipedia.org/wiki/European_Rail_Traffic_Management_System. Accessed: 29 Jan 2019.

[5] Redmine: bug-tracking system. https://www.redmine.org/projects/redmine/issues/. April, 2020.

[6] Related issues information. https://www.redmine.org/projects/redmine/wiki/RedmineIssues. Accessed: Aug 2021.

[7] Roi of machine learning. https://towardsdatascience.com/return-on-investment-for-machine-learning-1a0c431509e#:~:text=Just%20like%20with%20any%20investment,cost%20of%20mistakes%20and%20accuracy. Accessed: Dec 2021.

[8] Scikit-learn model evalutation. https://scikit-learn.org/stable/modules/modelevaluation.html#precision-recall-f-measure-metrics. Accessed: 15 Apr 2019.

[9] Scikit-learn supervised learning. https://scikit-learn.org/stable/supervisedlearning.html#supervised-learning. Accessed: 29 Jan 2019.

[10] Typo3 — the professional, flexible content management system. https://typo3.org/. Accessed: May, 2021.

[11] M. Abbas, A. Ferrari, A. Shatnawi, E. P. Enoiu, and M. Saadatmand. Is requirements similarity a good proxy for software similarity? an empirical investigation in industry. In *REFSQ*, pages 3–18, 2021.

[12] M. D. Aias Martakis. Handling requirements dependencies in agile projects: A focus group with agile software development practitioners. In *Proceedings of the 7th International Conference on Research Challenges in Information Science (RCIS)*, pages 1–11. IEEE, 2013.

[13] R. Aleithan. Explainable just-in-time bug prediction: Are we there yet? In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 129–131. IEEE, 2021.

[14] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann. Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 291–300. IEEE, 2019.

[15] A. Araujo, M. Golo, B. Viana, Sanches, 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*, pages 378–389. SBC, 2020.

[16] A. Arellano, E. Zontek-Carney, and M. A. Austin. Frameworks for natural language processing of textual requirements. *International Journal On Advances in Systems and Measurements*, 8:230–240, 2015.

[17] C. Arora, M. Sabetzadeh, S. Nejati, and L. Briand. An active learning approach for improving the accuracy of automated domain model extraction. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 28(1):4, 2019.

[18] N. Assawamekin, T. Sunetnanta, and C. Pluempitiwiriyawej. Ontology-based multiperspective requirements traceability framework. *Knowledge and Information Systems*, 25(3):493–522, 2010.

[19] M. Atas, R. Samer, and A. Felfernig. Automated identification of type-specific dependencies between requirements. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 688–695. IEEE, 2018.

[20] A. Aurum and C. Wohlin, editors. *Engineering and Managing Software Requirements*. Springer-Verlag, 2005.

[21] J. Barnes et al. *The Nicomachean Ethics*. Penguin, 2004.

[22] A. Begel and T. Zimmermann. Analyze this! 145 questions for data scientists in software engineering. In *ICSE*, pages 12–23, 2014.

[23] N. Bencomo, J. L. Guo, R. Harrison, H.-M. Heyn, and T. Menzies. The secret to better ai and better software (is requirements engineering). *IEEE Software*, 39(1):105–110, 2021.

[24] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *J. of ML research*, 13(2), 2012.

[25] D. M. Berry. Empirical evaluation of tools for hairy requirements engineering tasks. *Empirical Software Engineering*, 26(6):1–77, 2021.

[26] T. Bhowmik and S. Reddivari. Resolution trend of just-in-time requirements in open source software development. In *2015 IEEE Workshop on Just-In-Time Requirements Engineering (JITRE)*, pages 17–20. IEEE, 2015.

[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*, pages 20–29. 2021.

[28] S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Grünbacher. *Value-based software engineering*. Springer Science & Biz Media, 2006.

[29] M. Binkhonain and L. Zhao. A review of machine learning algorithms for identification and classification of non-functional requirements. *Expert Systems with Applications: X*, 1:100001, 2019.

[30] C. Bird, T. Menzies, and T. Zimmermann. *The art and science of analyzing software data*. Elsevier, 2015.

[31] G. Bockle, P. Clements, J. D. McGregor, D. Muthig, and K. Schmid. Calculating ROI for software product lines. *IEEE Software*, 21(3):23–31, May 2004. Conference Name: IEEE Software.

[32] B. Boehm, L. Huang, A. Jain, and R. Madachy. The ROI of software dependability: The iDAVE model. *IEEE Software*, 21(3):54–61, May 2004. Conference Name: IEEE Software.

[33] B. Boehm, R. Valerdi, and E. Honour. The roi of systems engineering: Some quantitative results for software-intensive systems. *Systems Engineering*, 11(3):221–234, 2008.

[34] F. P. Brooks and N. S. Bullet-Essence. Accidents of software engineering. *Computer*, 20(4):10–19, 1987.

[35] P. Carlshamre. Release planning in market-driven software product development: Provoking an understanding. *Requirements engineering*, 7(3):139–151, 2002.

[36] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. N. och Dag. An industrial survey of requirements interdependencies in software product release planning. In *Proceedings Fifth IEEE International Symposium on Requirements Engineering*, pages 84–91, Aug 2001.

[37] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. N. och Dag. An industrial survey of requirements interdependencies in software product release planning. In *5th IEEE International Symposium on Requirements Engineering (RE 2001), 27-31 August 2001, Toronto, Canada*, pages 84–93, 2001.

[38] R. Chitchyan and A. Rashid. Tracing requirements interdependency semantics. In *Early Aspects*, 2006.

[39] R. Chitchyan, A. Rashid, P. Rayson, and R. Waters. Semantics-based composition for aspect-oriented requirements engineering. In *Proceedings of the 6th International Conference on Aspect-oriented Software Development*, pages 36–48. ACM, 2007.

[40] J. Cleland-Huang, G. Zemont, and W. Lukasik. A heterogeneous solution for improving the return on investment of requirements traceability. In *Proc. 12th IEEE Requirements Engineering Conference, 2004.*, pages 230–239. IEEE, 2004.

[41] A. Costa, G. Pasta, and G. Bergamaschi. Intraoral hard and soft tissue depths for temporary anchorage devices. In *Seminars in orthodontics*, volume 11, pages 10–15. Elsevier, 2005.

[42] A. G. Dahlstedt and A. Persson. Requirements interdependencies-moulding the state of research into a research agenda. In *Proceedings of Ninth International Workshop on Requirements Engineering: Foundation for Software Quality, Klagenfurt/Velden, Austria*, pages 55–64. Citeseer, 2003.

[43] Å. G. Dahlstedt and A. Persson. Requirements interdependencies: state of the art and future challenges. In *Engineering and managing software requirements*, pages 95–116. Springer, 2005.

[44] A. G. Dahlstedt and A. Persson. Requirements interdependencies: State of the art and future challenges. In *Engineering and Managing Software Requirements*, pages 95–116. Springer, 2005.

[45] F. Dalpiaz, A. Ferrari, X. Franch, and C. Palomares. Natural language processing for requirements engineering: The best is yet to come. *IEEE Software*, 35(5):115–119, 2018.

[46] S. Das, N. Deb, A. Cortesi, and N. Chaki. Sentence embedding models for similarity detection of software requirements. *SN Computer Science*, 2(2):1–11, 2021.

[47] A. F. de Araújo and R. M. Marcacini. Re-bert: automatic extraction of software requirements from app reviews using bert language model. In *Proc. 36th Annual ACM Symposium on Applied Computing*, pages 1321–1327, 2021.

[48] D. Dermeval, J. Vilela, I. I. Bittencourt, J. Castro, S. Isotani, P. H. da S. Brito, and A. Silva. Applications of ontologies in requirements engineering: a systematic review of the literature. *Requirements Engineering Journal*, 21(4):405–437, 2016.

[49] G. Deshpande. Sreyantra: Automated software requirement inter-dependencies elicitation, analysis and learning. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 186–187. IEEE, 2019.

[50] G. Deshpande. Sreyantra: automated software requirement inter-dependencies elicitation, analysis and learning. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 186–187. IEEE, 2019.

[51] G. Deshpande, C. Arora, and G. Ruhe. Data-driven elicitation and optimization of dependencies between requirements. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pages 416–421. IEEE, 2019.

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

[53] G. Deshpande, Q. Motger, C. Palomares, I. Kamra, K. Biesialska, X. Franch, G. Ruhe, and J. Ho. Requirements dependency extraction by integrating active learning with ontology-based retrieval. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 78–89. IEEE, 2020.

[54] G. Deshpande and J. Rokne. User feedback from tweets vs app store reviews: an exploratory study of frequency, timing and content. In *2018 5th International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, pages 15–21. IEEE, 2018.

[55] G. Deshpande and G. Ruhe. Survey: Elicitation and maintenance of requirements dependencies. https://ispma.org/elicitation-and-maintenance-of-requirements-dependencies-a-state-of-the-practice-survey/. Accessed: Dec, 2018.

[56] G. Deshpande and G. Ruhe. Beyond accuracy: Roi-driven data analytics of empirical data. In *Proc. of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2020.

[57] G. Deshpande, G. Ruhe, and C. Saunders. How much data analytics is enough? the roi of machine learning classification and its application to requirements dependency classification. *arXiv preprint arXiv:2109.14097*, 2021.

[58] G. Deshpande, B. Sheikhi, S. Chakka, D. L. Zotegouon, M. N. Masahati, and G. Ruhe. Is bert the new silver bullet?-an empirical investigation of requirements dependency classification. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, pages 136–145. IEEE, 2021.

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

[60] V. T. Dhinakaran, R. Pulle, N. Ajmeri, and P. K. Murukannaiah. App review analysis via active learning: reducing supervision effort without compromising classification accuracy. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 170–181. IEEE, 2018.

[61] T. G. Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[62] P. Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[63] T.-b. Du, G.-h. Shen, Z.-q. Huang, Y.-s. Yu, and D.-x. Wu. Automatic traceability link recovery via active learning. *Frontiers of Information Technology & Electronic Engineering*, 21(8):1217–1225, 2020.

[64] M. Elbadawi, S. Gaisford, and A. W. Basit. Advanced machine-learning techniques in drug discovery. *Drug Discovery Today*, 2020.

[65] H. Erdogmus, J. Favaro, and W. Strigel. Return on investment. *IEEE Software*, 21(3):18–22, 2004.

[66] B. Farbey and A. Finkelstein. Evaluation in Software Engineering: ROI, but more than ROI. page 6.

[67] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, 1996.

[68] D. M. Fernández and S. Wagner. Naming the pain in requirements engineering: A design for a global family of surveys and first results from germany. *Information and Software Technology*, 57:616–643, 2015.

[69] D. M. Fernández, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetrò, T. Conte, M.-T. Christiansson, D. Greer, C. Lassenius, et al. Naming the pain in requirements engineering. *Empirical software engineering*, 22(5):2298–2338, 2017.

[70] A. Ferrari, G. O. Spagnolo, and S. Gnesi. Pure: A dataset of public requirements documents. In *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, pages 502–505. IEEE, 2017.

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

[72] I. Figalist, C. Elsner, J. Bosch, and H. H. Olsson. An end-to-end framework for productive use of machine learning in software analytics and business intelligence solutions. In *International Conference on Product-Focused Software Process Improvement*, pages 217–233. Springer, 2020.

[73] 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.

[74] R. C. d. C. François Candelon et al. Ai regulation is coming: How to prepare for the inevitable. 2021.

[75] A. Goknil, I. Kurtev, K. Van Den Berg, and W. Spijkerman. Change impact analysis for requirements: A metamodeling approach. *Information and Software Technology*, 56(8):950–972, 2014.

[76] A. Goknil, I. Kurtev, K. van den Berg, and J.-W. Veldhuis. Semantics of trace relations in requirements models for consistency checking and inferencing. *Software & Systems Modeling*, 10(1):31–54, 2011.

[77] H. Guan, G. Cai, and C. Zhao. An automatic approach to extracting requirement dependencies based on semantic web. In *2021 8th International Conference on Dependable Systems and Their Applications (DSA)*, pages 414–420. IEEE, 2021.

[78] J. Guo, J. Cheng, and J. Cleland-Huang. Semantically enhanced software traceability using deep learning techniques. In *2017 IEEE/ACM 39th ICSE*, pages 3–14. IEEE, 2017.

[79] J. Guo, M. Gibiec, and J. Cleland-Huang. Tackling the term-mismatch problem in automated trace retrieval. *Empirical Software Engineering*, 22(3):1103–1142, 2017.

[80] J. Guo and J.-C. Huang. Ontology learning and its application in software-intensive projects. In *Proceedings of the 38th International Conference on Software Engineering Companion*, pages 843–846, 2016.

[81] L.-Z. Guo and Y.-F. Li. A general formulation for safely exploiting weakly supervised data. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[82] E. Guzman, M. El-Haliby, and B. Bruegge. Ensemble methods for app review classification: An approach for software evolution (n). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 771–776. IEEE, 2015.

[83] M. Harman, J. Krinke, J. Ren, and S. Yoo. Search based data sensitivity analysis applied to requirement engineering. In *Proc. of the 11th Annual conf. on Genetic and evolutionary computation*, pages 1681–1688, 2009.

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

[85] S. Y. Ho, K. Phua, L. Wong, and W. W. B. Goh. Extensions of the external validation for checking learned model interpretability and generalizability. *Patterns*, 1(8):100129, 2020.

[86] C.-W. Hsu, C.-C. Chang, C.-J. Lin, et al. A practical guide to support vector classification. 2003.

[87] huggingface.co. Bert with the huggingface pytorch library. https://huggingface.co/transformers/training.html. Accessed: May 2021.

[88] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to Statistical Learning*, volume 112. Springer, 2013.

[89] T. M. Khoshgoftaar, E. B. Allen, W. D. Jones, and J. P. Hudepohl. Cost-Benefit Analysis of Software Quality Models. *Software Quality Journal*, 9(1):9–30, Jan. 2001.

[90] B. Kim, S. Kang, and S. Lee. A weighted pagerank-based bug report summarization method using bug report relationships. *Applied Sciences*, 9(24):5427, 2019.

[91] E. Kocaguneli, T. Menzies, et al. Active learning and effort estimation: Finding the essential content of software effort estimation data. *IEEE Transactions on Software Engineering*, 39(8):1040–1053, 2013.

[92] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas. Data preprocessing for supervised leaning. *International journal of computer science*, 1(2):111–117, 2006.

[93] R. Krishna and T. Menzies. Bellwethers: A baseline method for transfer learning. *IEEE Transactions on Software Engineering*, 45(11):1081–1105, 2018.

[94] M. Kuhn, K. Johnson, et al. *Applied predictive modeling*, volume 26. Springer, 2013.

[95] V. Kulshreshtha, J. T. Boardman, and D. Verma. The emergence of requirements networks: the case for requirements inter-dependencies. *IJCAT*, 45(1):28–41, 2012.

[96] Y. Li and J. Cleland-Huang. Ontology-based trace retrieval. In *2013 7th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE)*, pages 30–36. IEEE, 2013.

[97] S. Lim, A. Henriksson, and J. Zdravkovic. Data-driven requirements elicitation: A systematic literature review. *SN Computer Science*, 2(1):1–35, 2021.

[98] J. Lin, Y. Liu, Q. Zeng, M. Jiang, and J. Cleland-Huang. Traceability transformed: Generating more accurate links with pre-trained bert models. In *2021 Proc. ICSE Conference*, pages 324–335. IEEE, 2021.

[99] C. X. Ling, S. Sheng, T. Bruckhaus, and N. H. Madhavji. Predicting software escalations with maximum roi. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4–pp. IEEE, 2005.

[100] D. Lo, L. Jiang, A. Budi, et al. Active refinement of clone anomaly reports. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 397–407. IEEE, 2012.

[101] H. Lu, E. Kocaguneli, and B. Cukic. Defect prediction between software versions with active learning and dimensionality reduction. In *2014 IEEE 25th International Symposium on Software Reliability Engineering*, pages 312–322. IEEE, 2014.

[102] S. Ma, S. Wang, D. Lo, R. H. Deng, and C. Sun. Active semi-supervised approach for checking app behavior against its description. In *2015 IEEE 39th Annual Computer Software and Applications Conference*, volume 2, pages 179–184. IEEE, 2015.

[103] W. Maalej, M. Ellmann, and R. Robbes. Using contexts similarity to predict relationships between tasks. *Journal of Systems and Software*, 128:267–284, 2017.

[104] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe. Toward data-driven requirements engineering. *IEEE Software*, 33(1):48–54, 2016.

[105] R. Malhotra and S. Meena. Empirical validation of cross-version and 10-fold cross-validation for defect prediction. In *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pages 431–438. IEEE, 2021.

[106] S. J. Martínez Fernández, C. P. Ayala Martínez, and J. Franch Gutiérrez. Rearm:a reuse-based economic model for software reference architectures. In *Int. Conf on Software Reuse*, pages 97–112. Springer, 2013.

[107] M. McDaniel and V. C. Storey. Evaluating domain ontologies: Clarification, classification, and challenges. *ACM Comput. Surv.*, 52(4):70:1–70:44, 2019.

[108] D. Meignan, S. Knust, J.-M. Frayret, G. Pesant, and N. Gaud. A review and taxonomy of interactive optimization methods in operations research. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(3):17, 2015.

[109] T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, and T. Zimmermann. Local versus global lessons for defect prediction and effort estimation. *IEEE Transactions on software engineering*, 39(6):822–834, 2012.

[110] T. Menzies and M. Shepperd. Special issue on repeatable results in software engineering prediction. *Empirical Software Engineering*, 17(1-2):1–17, Jan. 2012.

[111] P. Mikalef, I. Pappas, J. Krogstie, and P. A. Pavlou. *Big data and business analytics: A research agenda for realizing business value*. Elsevier, 2020.

[112] A. T. Mısırlı, A. B. Bener, and B. Turhan. An industrial case study of classifier ensembles for locating software defects. *Software Quality Journal*, 19(3):515–536, 2011.

[113] MLworkflow. Machine learning workflow. https://cloud.google.com/mlengine/docs/tensorflow/ml-solutions-overview. Accessed: Aug, 2021.

[114] Q. Motger, R. Borrull, C. Palomares, and J. Marco. OpenReq-DD API documentation. https://api.openreq.eu/dependency-detection/swagger-ui.html. Accessed 2020-02-04. [Online].

[115] Q. Motger, R. Borrull, C. Palomares, and J. Marco. Openreq-dd: A. requirements dependency detection tool. In *Joint Proceedings of REFSQ-2019 Workshops, Doctoral Symposium, Live Studies Track, and Poster Track co-located with the 25th International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2019), Essen, Germany, March 18th, 2019*, 2019.

[116] Mozilla.org. Userguide/bugfields. https://wiki.mozilla.org/BMO/UserGuide/BugFields#bug_type. Accessed:April, 2020.

[117] P. K. Murukannaiah and M. P. Singh. Platys: An active learning framework for place-aware application development and its evaluation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 24(3):1–32, 2015.

[118] I. Muslea, S. Minton, and C. A. Knoblock. Active learning with multiple views. *Journal of Artificial Intelligence Research*, 27:203–233, 2006.

[119] S. Nagrecha and N. V. Chawla. Quantifying decision making for data science: from data acquisition to modeling. *EPJ Data Science*, 5(1):27, 2016.

[120] J. Nam, S. J. Pan, and S. Kim. Transfer defect learning. In *2013 35th international conference on software engineering (ICSE)*, pages 382–391. IEEE, 2013.

[121] M. Nayebi, M. Marbuti, R. Quapp, F. Maurer, and G. Ruhe. Crowdsourced exploration of mobile app features: A case study of the fort mcmurray wildfire. In *Proc. ICSE*. ACM, 2017.

[122] R. Neches, R. E. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. R. Swartout. Enabling technology for knowledge sharing. *AI magazine*, 12(3):36–36, 1991.

[123] A. Ngo-The and M. O. Saliu. Fuzzy structural dependency constraints in software release planning. In *The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ'05.*, pages 442–447. IEEE, 2005.

[124] J. N. och Dag, B. Regnell, P. Carlshamre, M. Andersson, and J. Karlsson. A feasibility study of automated natural language requirements analysis in market-driven development. *RE*, 7(1):20–33, 2002.

[125] F. Olsson. A literature survey of active machine learning in the context of natural language processing. 2009.

[126] I. Omoronyia, G. Sindre, T. Stålhane, S. Biffl, T. Moser, and W. D. Sunindyo. A domain ontology building process for guiding requirements elicitation. In *Procs. of the 16th International Working Conference on Requirements Engineering: Foundation for Software Quality REFSQ*, volume 6182 of *Lecture Notes in Computer Science*, pages 188–202. Springer, 2010.

[127] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[128] K. Pohl. *Process-Centered Requirements Engineering*. Research Studies Pre, 1996.

[129] J. Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. New Jersey, USA, 2003.

[130] S. Rastkar, G. C. Murphy, and G. Murray. Automatic summarization of bug reports. *IEEE Transactions on Software Engineering*, 40(4):366–380, 2014.

[131] J. C. Ribeiro and J. Araújo. Asporas: A requirements agile approach based on scenarios and aspects. In *Proceedings of the 7th International Conference on Research Challenges in Information Science (RCIS)*, pages 313–324, 2008.

[132] H. W. Rittel and M. M. Webber. Wicked problems. *Man-made Futures*, 26(1):272–280, 1974.

[133] G. Ruhe. *Product release planning: methods, tools and applications*. Auerbach Publications, 2010.

[134] G. Ruhe. Optimization in software engineering: A pragmatic approach. In *Contemporary Empirical Methods in Software Engineering*, pages 235–261. Springer, 2020.

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

[136] G. Ruhe and M. O. Saliu. The art and science of software release planning. *IEEE software*, 22(6):47–53, 2005.

[137] I. Rus, M. Lindvall, and S. Sinha. Knowledge management in software engineering. *IEEE software*, 19(3):26–38, 2002.

[138] A. Sainani, P. R. Anish, V. Joshi, and S. Ghaisas. Extracting and classifying requirements from software engineering contracts. In *2020 Proc. RE Conference*, pages 147–157. IEEE, 2020.

[139] R. Samer, M. Stettinger, M. Atas, A. Felfernig, G. Ruhe, and G. Deshpande. New approaches to the identification of dependencies between requirements. In *31st Conference on Tools with Artificial Intelligence*, ICTAI '19. ACM, 2019.

[140] H. Schütze, C. D. Manning, and P. Raghavan. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103, 2010.

[141] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[142] B. Settles. From theories to queries: Active learning in practice. In *Active Learning and Experimental Design workshop colocated with AISTATS 2010*, pages 1–18, 2011.

[143] S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed. Machine learning for software engineering: A systematic mapping. *arXiv preprint arXiv:2005.13299*, 2020.

[144] M. Shepperd. Cost prediction and software project management. *Software project management in a changing world*, pages 51–71, 2014.

[145] L. Shi, C. Chen, Q. Wang, S. Li, and B. Boehm. Understanding feature requests by leveraging fuzzy method and linguistic analysis. In *Proc. of the 32nd Conf. on ASE*, pages 440–450. IEEE Press, 2017.

[146] Y. Shin, J. H. Hayes, and J. Cleland-Huang. Guidelines for benchmarking automated software traceability techniques. In *Proceedings of the 8th Int. Symposium on Software and Systems Traceability*, pages 61–67. IEEE Press, 2015.

[147] F. Shull, J. Singer, and D. I. Sjøberg. *Guide to advanced empirical software engineering*. Springer, 2007.

[148] E. P. B. Simperl, C. Tempich, and Y. Sure. Ontocom: A cost estimation model for ontology engineering. In *International Semantic Web Conference*, pages 625–639. Springer, 2006.

[149] S. Soomro, A. Hafeez, A. Shaikh, and S. H. A. Musavi. Ontology based requirement interdependency representation and visualization. In *International Multi Topic Conference*, pages 259–270. Springer, 2013.

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

[151] K. P. Stan Buehne, Guenter Halmans. Modelling dependencies between variation points in use case diagrams. In *International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ)*, pages 59–70, 2003.

[152] S. Sun. A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7-8):2031–2038, Feb. 2013.

[153] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto. An empirical comparison of model validation techniques for defect prediction models. *IEEE TSE*, 43(1):1–18, 2017.

[154] I. Tenney, D. Das, and E. Pavlick. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

[155] R. Van Solingen. Measuring the roi of software process improvement. *IEEE software*, 21(3):32–38, 2004.

[156] K. Verma and A. Kass. Requirements analysis tool: A tool for automatically analyzing software requirements documents. In *International semantic web conference*, pages 751–763. Springer, 2008.

[157] A. Vogelsang. Feature dependencies in automotive software systems: Extent, awareness, and refactoring. *Journal of Systems and Software*, page 110458, 2019.

[158] A. Vogelsang and S. Fuhrmann. Why feature dependencies challenge the requirements engineering of automotive systems: An empirical study. In *2013 21st IEEE Requirements Engineering Conference (RE)*, pages 267–272. IEEE, 2013.

[159] S. Wagner, D. M. Fernández, M. Felderer, A. Vetrò, M. Kalinowski, R. Wieringa, D. Pfahl, T. Conte, M.-T. Christiansson, D. Greer, et al. Status quo in requirements engineering: A theory and a global family of surveys. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 28(2):1–48, 2019.

[160] J. Wang, S. Wang, Q. Cui, and Q. Wang. Local-based active classification of test report to assist crowdsourced testing. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, pages 190–201, 2016.

[161] W. Wang, F. Dumont, N. Niu, and G. Horton. Detecting software security vulnerabilities via requirements dependency analysis. *IEEE Transactions on Software Engineering*, pages 1–1, 2020.

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

[163] G. M. Weiss and Y. Tian. Maximizing classifier utility when there are data acquisition and modeling costs. *Data Mining and Knowledge Discovery*, 17(2):253–282, 2008.

[164] N. Weston, R. Chitchyan, and A. Rashid. Formal semantic conflict detection in aspect-oriented requirements. *Requirements Engineering*, 14(4):247, 2009.

[165] R. J. Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.

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

[167] www.altexsoft.com. Measuring roi for machine learning and data science. https://www.altexsoft.com/blog/business/how-to-estimate-roi-and-costs-for-machine-learning-and-data-science-projects/. Accessed: Dec 2021.

[168] Z. Xu, J. Liu, X. Luo, and T. Zhang. Cross-version defect prediction via hybrid active learning with kernel principal component analysis. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 209–220. IEEE, 2018.

[169] X.-S. Yang, S. Deb, Y.-X. Zhao, S. Fong, and X. He. Swarm intelligence: past, present and future. *Soft Computing*, 22(18):5923–5933, 2018.

[170] Y. Yang, X. Xia, D. Lo, T. Bi, J. Grundy, and X. Yang. Predictive models in software engineering: Challenges and opportunities. *arXiv preprint arXiv:2008.03656*, 2020.

[171] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196, 1995.

[172] D. S. Yudi Priyadi, Arif Djunaidy. Requirements dependency graph modeling on software requirements specification using text analysis. In *1st International Conference on Cybernetics and Intelligent System (ICORIS)*, pages 221–226. IEEE, 2019.

[173] K. Zamani, D. Zowghi, and C. Arora. Machine learning in requirements engineering: A mapping study. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, pages 116–125. IEEE, 2021.

[174] H. Zhang, J. Li, L. Zhu, D. R. Jeffery, Y. Liu, Q. Wang, and M. Li. Investigating dependencies in software requirements for change propagation analysis. *Information & Software Technology*, 56(1):40–53, 2014.

[175] L. Zhang, J.-H. Tian, J. Jiang, Y.-J. Liu, M.-Y. Pu, and T. Yue. Empirical research in software engineering—a literature survey. *Journal of Computer Science and Technology*, 33(5):876–899, 2018.

[176] W. Zhang, H. Mei, and H. Zhao. A feature-oriented approach to modeling requirements dependencies. In *13th IEEE Conf. Requirements Engineering*, pages 273–282. IEEE, 2005.

[177] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, and R. T. Batista-Navarro. Natural language processing (nlp) for requirements engineering: A systematic mapping study. *arXiv preprint arXiv:2004.01099*, 2020.

[178] Z.-H. Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2017.

[179] T. Zimmermann. One size does not fit all. In *Perspectives on Data Science for Software Engineering*, pages 347–348. Elsevier, 2016.

Appendix A

Publication consent from my collaborators

From: Chahal Arora <[redacted]@ucalgary.ca>
Sent: November 19, 2021 3:19 PM
To: Gouri Deshpande <[redacted]@ucalgary.ca>
Subject: Re: [For my thesis] Need your permission to include our work together on a publication

Hi Gouri,

I am doing fine. How are you?

Please regard this email as an approval to include content from our publication in your thesis. All the best 🙂

Thanks and Regards
Chahal Arora

Ikagarjot Kamra <[redacted]@outlook.com>
Sun 11/21/2021 6:50 PM
To: Gouri Deshpande

[△EXTERNAL]

Hi Gouri,

As discussed, please consider this email as an approval to include the mentioned research paper in your thesis.

Wish you all the very best.

Regards
Ikagarjot Singh

From: Jason Ho <[redacted]@atb.com>
Sent: November 23, 2021 11:06 AM
To: Gouri Deshpande <[redacted]@ucalgary.ca>
Subject: Re: [Ext Sender] [For my thesis] Need your permission to include our work together on a publication

[△EXTERNAL]

Hi Gouri,

Yes, I approve. Please proceed and good luck with your defence

Regards
Jason Ho

Cristina Palomares <[redacted]@gmail.com>
Sat 11/20/2021 8:15 AM
To: Gouri Deshpande
Cc: [redacted]

[△EXTERNAL]

Hi Gouri,

Everything is good from my side, which is quite a lot in the given situation.

Your proposal looks good to me. Just a small change, my name is spelled "Cristina".

Good luck with your thesis.

Best regards,
Cristina

Katarzyna Biesialska <[redacted]@upc.edu>
Sat 11/20/2021 3:11 AM
To: Gouri Deshpande

[△EXTERNAL]

Dear Gouri,

yes, I confirm that the statement of contribution is correct concerning my role in the publication.

Good luck with your thesis and its defense! 🤞

All the best!

Kasia

Quim Motger <[redacted]@essi.upc.edu>
Mon 11/22/2021 12:34 AM
To: [redacted]
Cc: Guenther Ruhe

[△EXTERNAL]

Dear Gouri,

I hope you are doing well.

Looks good to me too. I confirm my approval.

Good luck with your thesis :)

Best regards,

From: Dylan Valentin Lachou Zotegouon <[redacted]@ucalgary.ca>
Sent: Friday, November 19, 2021 3:43 PM
To: Gouri Deshpande <[redacted]@ucalgary.ca>
Subject: RE: [For my thesis] Need your permission to include our work together on a publication

You have my full support Gouri

And Good luck on your Thesis

Xavier Franch <[redacted]@essi.upc.edu>
Mon 11/22/2021 12:42 AM
To: Gouri Deshpande; [redacted]
Cc: Guenther Ruhe

[△EXTERNAL]

Dear Gouri,

I confirm my approval.

Best luck with the preparations!

Regards,

Xavi

From: Behnaz Sheikhi <[redacted]ucalgary.ca>
Sent: November 20, 2021 12:46 PM
To: Gouri Deshpande <[redacted]@ucalgary.ca>
Subject: Re: [For my thesis] Need your permission to include our work together on a publication

Hi, dear Gouri,

I hope you are doing well.

Sure you can include the paper in your thesis.

Good luck.

Sincerely,
Behnaz Sheikhi

From: Dylan Valentin Lachou Zotegouon <[redacted]@ucalgary.ca>
Sent: Friday, November 19, 2021 3:43 PM
To: Gouri Deshpande <[redacted]@ucalgary.ca>
Subject: RE: [For my thesis] Need your permission to include our work together on a publication

You have my full support Gouri

And Good luck on your Thesis

From: Sai Preetham Chakka <[redacted]@ucalgary.ca>
Sent: Friday, November 19, 2021 10:19 PM
To: Gouri Deshpande <[redacted]e@ucalgary.ca>
Subject: Approval for thesis

Hello Gouri,

I hope you are doing good. I approve to include the publication "Is BERT the New Silver Bullet? - An Empirical Investigation c Requirements Dependency Classification", 8th International Workshop on Artificial Intelligence for Requirements Engineering, Canada, 2021 for your thesis.

Thanks, and Regards
Sai Preetham Chakka.

From: Mohammad Navid Masahati <[redacted]@ucalgary.ca>
Sent: Friday, November 19, 2021 5:59 PM
To: Gouri Deshpande <[redacted]@ucalgary.ca>
Subject: Re: [For my thesis] Need your permission to include our work together on a publication

Hello Gouri,

I approve, of course. 🙂
best of luck with your thesis.

Kind regards,
Mohammad Navid

From: Chad Saunders <[redacted]@ucalgary.ca>
Sent: Friday, November 19, 2021 4:23 PM
To: Gouri Deshpande <[redacted]ucalgary.ca>
Subject: Re: [For my thesis] Need your permission to include our work together on a paper

Hi Gouri,

Glad to hear you are keeping on top of the paper review process.

I can confirm my approval of including the paper noted below in your thesis.

Thanks

Chad~