IDENTIFICATION AND CATEGORIZATION OF DEFECTS IN CONSTRUCTION SPECIFICATIONS BY UTILIZING NLP

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖZGÜR MADENLİ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
CIVIL ENGINEERING

JANUARY 2025

Approval of the thesis:

IDENTIFICATION AND CATEGORIZATION OF DEFECTS IN CONSTRUCTION SPECIFICATIONS BY UTILIZING NLP

submitted by **ÖZGÜR MADENLİ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** i**n Civil Engineering, Middle East Technical University** by,

Prof. Dr. Naci Emre Altun
Dean, **Graduate School of Natural and Applied Sciences** _____

Prof. Dr. Erdem Canbay
Head of the Department, **Civil Engineering, METU** _____

Assoc. Prof. Dr. Güzide Atasoy Özcan
Supervisor, **Civil Engineering, METU** _____

Prof. Dr. İrem Dikmen Toker
Co-Supervisor, **Construction Man. & Eng., Univ. of Reading** _____

Examining Committee Members:

Prof. Dr. Rıfat Sönmez
Civil Engineering, METU _____

Assoc. Prof. Dr. Güzide Atasoy Özcan
Civil Engineering, METU _____

Assist. Prof. Dr. Aslı Akçamete Güngör
Civil Engineering, METU _____

Assist. Prof. Dr. Gözde Bilgin
Civil Engineering, Başkent University _____

Assist. Prof. Dr. Saman Aminbakhsh
Civil Engineering, Atılım University _____

Date: 08.01.2025

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Last name : Özgür Madenli

Signature :

ABSTRACT

IDENTIFICATION AND CATEGORIZATION OF DEFECTS IN CONSTRUCTION SPECIFICATIONS BY UTILIZING NLP

Madenli, Özgür
Doctor of Philosophy, Civil Engineering
Supervisor: Assoc. Prof. Dr. Güzide Atasoy Özcan
Co-Supervisor: Prof. Dr. İrem Dikmen Toker

January 2025, 175 pages

Construction specifications are crucial parts of design and contract documents. Defective specification statements can cause not only a faulty outcome but also disputes among project stakeholders, claims for project budget and time, project disruptions and even litigation. Identification of defects in technical sections of construction specifications is challenging due to the extensive document volume, limited resources, and reliance on technical staff's experience. Natural Language Processing (NLP) can facilitate analyzing language, uncovering patterns, providing insights, and overcoming the limitations of manual methods. This research aims to develop a structured framework and implement supervised NLP methods for identifying and categorizing defects in specifications. Dataset includes 175 specifications related to 21 architectural works collected from 16 construction projects. A total of 15569 statements were extracted and manually labeled in four defect categories and 8 Machine Learning models, ranging from shallow to transformer-based models, were trained and tested with combinations of different text representation techniques. Subsequently, a study with ChatGPT was conducted.

The research concluded that the pre-trained RoBERTa model outperformed the recognition of defects in construction specifications with a macro F1 score of 91.2% and 98% accuracy. Whereas the performance of the ChatGPT was evaluated as considerably low compared to domain-specific trained ML models. This research offers a data-driven and automated methodology providing construction stakeholders with practical tools to enhance the quality of specifications and decrease disputes by reducing the deficiencies during design, bidding and pre-construction.

Keywords: Construction Specification, Machine Learning, Natural Language Processing, Text Classification, ChatGPT

ÖZ

İNŞAAT TEKNİK ŞARTNAMELERİNDEKİ KUSURLARIN NLP KULLANARAK TESPİTİ VE SINIFLANDIRILMASI

Madenli, Özgür
Doktora, İnşaat Mühendisliği
Tez Yöneticisi: Doç. Dr. Güzide Atasoy Özcan
Ortak Tez Yöneticisi: Prof. Dr. İrem Dikmen Toker

Ocak 2025, 175 sayfa

İnşaat teknik şartnameleri tasarım ve sözleşme belgelerinin önemli bir parçasıdır. Kusurlu şartname maddeleri sadece hatalı bir inşaata yol açmakla kalmaz, proje paydaşları arasında maliyet ve süreyi arttıracak ihtilaflara, inşaatın kesilmesine ve bazen de davalara neden olur. Şartnamelerdeki sayfa sayısının yüksek, insan kaynağının ve zamanın yetersiz olması ve teknik personelin deneyimine bağımlılık gibi nedenler bu kusurların tespit edilmesini oldukça güçleştirmektedir. Doğal Dil İşleme'nin (NLP) insan dilini yorumlayıp belgelerdeki kalıpları ve bağlamları analiz edebilme yeteneği manuel yöntemlerde karşılaşılan zorlukları aşarak başarılı çözümler sunabilir. Bu çalışmada inşaat teknik şartname maddelerindeki kusurların tespit ve sınıflandırılması için yapısal bir çerçeve ve bunun denetimli öğrenme NLP yöntemleri ile uygulanması hedeflenmiştir. Uluslararası 16 inşaat projesinden 21 farklı mimari işe ait toplam 175 şartname veri seti olarak kullanılmıştır. 15569 şartname maddesi içindeki kusurlar dört farklı kusur sınıfında el ile işaretlenmiş, farklı metin vektörü dönüştürme yöntemleri ile 8 farklı makine öğrenimi modeli eğitilmiş ve test edilmiştir. Akabinde ChatGPT ile benzer bir çalışma yapılmıştır.

Araştırma, eğitilmiş RoBERTa modelinin %91,2 oranında makro F1 ve %98 oranında doğrulukla teknik şartnameler içindeki kusurların tespit ve sınıflandırılmasında başarı elde ettiğini göstermiştir. Buna karşın ChatGPT alana özel veri ile eğitilmiş makine öğrenimi modellerine göre oldukça düşük bir performans sergilemiştir. Bu çalışma, inşaat sektörü paydaşlarına şartnamelerin kalitesini geliştirecek ve tasarım, ihale ve inşaat öncesi hazırlık aşamalarında şartnamelerdeki kusurları eksilterek olası ihtilafları azaltacak veri odaklı ve otomatikleştirilmiş bir yöntem önermektedir.

Anahtar Kelimeler: Teknik Şartname, Makine Öğrenimi, Doğal Dil İşleme, Metin Sınıflandırması, ChatGPT



To me

ACKNOWLEDGMENTS

Above all, I would like to express my profound gratitude to my supervisors, Assoc. Prof. Dr. Güzide Atasoy Özcan and Prof. Dr. İrem Dikmen Toker, for granting me the opportunity to conduct this research with their invaluable guidance and support.

I am sincerely grateful to Vicdan Erdoğan for her assistance in the progress of this study. I also deeply appreciate my associates Elvan Odabaşı and Alper Ünalan for their valuable comments.

I owe genuine thankfulness to my eternal friend Servet Ceylan, whose unwavering mental support I have always felt and will continue to cherish.

I want to express my deepest appreciation to my parents, Melahat and Telat, and my beloved sister, Sinem, for their trust, respect, and significant spirit throughout my life.

Finally, my heartfelt gratitude goes to my wife, Evrim, for her persuasive encouragement to start and complete this thesis, her patience through every challenge, and her unyielding belief in me.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xv

LIST OF FIGURES xvii

LIST OF ABBREVIATIONS xix

CHAPTERS

1 INTRODUCTION 1

1.1 Research Background 1

1.2 Research Gap and Objectives 4

1.3 Overview of the Research Method 6

1.4 Thesis Structure 7

2 LITERATURE 9

2.1 Construction Specifications 9

2.1.1 What is a Specification? 10

2.1.2 Construction Specification Types 12

2.1.3 Significance of Construction Specifications 14

2.2 AI, ML & NLP 19

2.2.1 Origins of NLP 19

2.2.2 NLP in Common Practice 25

2.2.3 NLP in the Construction Domain 27

3 RESEARCH METHODOLOGY 35

3.1 Research Design 35
3.2 Research Stages 37
3.2.1 Data Acquisition 37
3.2.2 Text Cleaning 37
3.2.3 Text Labelling 38
3.2.4 Data Set Validation 38
3.2.5 Preprocessing 39
3.2.6 Text Representation 43
3.2.7 ML Modelling 50
3.2.8 Performance Evaluation 57
3.3 Research Tools 60
4 DATA ACQUISITION AND PREPARATION 63
4.1 Data Collection and Selection 63
4.2 Specification Texts Cleaning and Initial Preprocessing 66
4.3 Sentence Labeling 70
4.3.1 Labeling Framework 71
4.3.2 Labeling Statistics 78
4.4 Labeling Validation 83
4.4.1 Validation Test 83
4.4.2 Validation Stage 84
4.4.3 Validation Assessment 86
4.4.4 Validation Outcomes 89
5 DEVELOPMENT AND IMPLEMENTATION OF ML MODELS 91
5.1 Text Preprocessing 91

5.1.1 Text Preprocessing Steps 92
5.1.2 Outcome of Text Preprocessing 95
5.2 Feature Extraction from Specifications 96
5.3 Tuning and Implementation of ML Models 99
5.3.1 Naïve Bayes Models 101
5.3.2 K-Nearest Neighbor Models 102
5.3.3 Linear Support Vector Machine Models 103
5.3.4 Support Vector Machine with RBF Kernel Models 105
5.3.5 Logistic Regression Models 107
5.3.6 TextCNN Models 108
5.3.7 BERT & RoBERTa Models 113
6 PERFORMANCE RESULTS 115
6.1 Evaluation Strategy 115
6.2 Performances of Implemented ML Classification Models 116
6.2.1 NB Models 116
6.2.2 kNN Models 118
6.2.3 Linear SVM Models 120
6.2.4 SVM-RBF Models 123
6.2.5 LR Models 125
6.2.6 TextCNN Models 127
6.2.7 BERT Models 129
6.2.8 RoBERTa 131
6.3 Comparative Evaluation 133
6.4 Validation of Results via Project-Based Investigation 138

6.5 Test of GPT in Defect Determination 141
7 DISCUSSION 147
7.1 Major Findings and Comparison with Existing Research 147
7.2 Error Analysis 151
8 CONCLUSION 155
8.1 Overview of the Study 155
8.2 Key Outcomes and Contributions 156
8.3 Limitations and Recommendations 157
REFERENCES 161
CURRICULUM VITAE 175

LIST OF TABLES

TABLES

Table 1.1 Organization of Thesis 7
Table 3.1 Sample Corpus 44
Table 3.2 Sample Vocabulary Set 44
Table 3.3 Sample Document-Term Matrix (BoW) 45
Table 3.4 Sample TF Calculation 46
Table 3.5 Sample IDF Calculation 47
Table 3.6 Sample Document-Term Matrix (TF-IDF) 47
Table 3.7 Confusion Matrix Sample 58
Table 4.1 Selected Specifications 65
Table 4.2 Specification Distribution Among Projects 66
Table 4.3 Sentence Distribution in Dataset 70
Table 4.4 Defect Categories 72
Table 4.5 Defect Grouping 73
Table 4.6 Defect Frequencies According to Projects 81
Table 4.7 Defect Frequencies According to Specification Titles 82
Table 4.8 Validation Set 85
Table 4.9 Validation Stage Participants 86
Table 4.10 Percentage Agreement Results 87
Table 4.11 Cohen's Kappa Results 87
Table 4.12 Interpretation of Kappa Value (Landis & Koch, 1977) 88
Table 4.13 Majority Voting Results 89
Table 5.1 Sample Preprocessed Statements 95
Table 5.2 Text Representation Approaches in Research 97
Table 5.3 Macro F1 Scores based on Alpha Parameters of NB 101
Table 5.4 Implemented NB Models 102
Table 5.5 Macro F1 Scores based on k-values of kNN 102

Table 5.6 Implemented kNN Models 103
Table 5.7 Macro F1 Scores based on C-Parameters of Linear SVM 104
Table 5.8 Implemented Linear SVM Models 105
Table 5.9 Implemented SVM with RBF Models 106
Table 5.10 Macro F1 Scores based on C-Parameters of LR 107
Table 5.11 Implemented LR Models 108
Table 5.12 F1 Scores According to Kernel Sets 111
Table 5.13 Implemented TextCNN Models 113
Table 5.14 Implemented BERT & RoBERTa Models 114
Table 6.1 Numbering of Implemented ML Models 115
Table 6.2 Performances of NB Models 117
Table 6.3 Performances of kNN Models 119
Table 6.4 Performances of Linear SVM Models 121
Table 6.5 Performances of SVM-RBF Models 123
Table 6.6 Performances of LR Models 125
Table 6.7 Performances of TextCNN Models 127
Table 6.8 Performances of BERT Models 129
Table 6.9 Performances of RoBERTa Models 131
Table 6.10 Test Sets of Project-Based Classification Investigation 139
Table 6.11 Performance Results of Project-Based Classification Investigation 140
Table 6.12 Results of 1. Test with ChatGPT 144
Table 6.13 Results of 1. Test with ChatGPT as Binary Class 144
Table 6.14 Results of 2. Test with ChatGPT 145
Table 6.15 Results of 2. Test with ChatGPT as Binary Class 145
Table 7.1 Highest Evaluation Metrics Achieved from Implemented Models 148
Table 7.2 F1 Scores in Main Research, Project-based, and GPT 150
Table 7.3 Binary Classification Confusion Matrix of Model-92 151
Table 7.4 Five Class Confusion Matrix of Model-92 152

LIST OF FIGURES

FIGURES

Figure 2.1. ML Methods (Lang, 2022) 20
Figure 2.2. Relations between AI, ML & NLP (Vajjala et al., 2020) 22
Figure 2.3. Basic NLP Pipeline 24
Figure 3.1. Research Flowchart 36
Figure 3.2. Supervised ML Flowchart for Text Classification 51
Figure 3.3. Typical Linear SVM Boundary Condition 54
Figure 3.4. Typical Convolutional Layer 56
Figure 3.5. Transformation of BERT and RoBERTa to a Classifier 57
Figure 4.1. Specification Text Cleaning Flowchart 67
Figure 4.2. Cleaned Raw Specification in Excel File 69
Figure 4.3. Distribution of Labeled Defect Groups in Dataset 79
Figure 4.4. Validation Flowchart 83
Figure 5.1. Sentence Labeling in Excel Medium 91
Figure 5.2. Preprocessing Steps 93
Figure 5.3. Histogram of Token Distribution 96
Figure 5.4. Developed TextCNN Architecture 109
Figure 5.5. Epochs vs F1 Scores for Various Kernels 111
Figure 5.6. F1 Score vs Filter Size 112
Figure 6.1. Frequencies of Defect Groups 116
Figure 6.2. Test Set Ratio vs F1 Scores of NB Models 118
Figure 6.3. Max. F1 Scores of NB Models for Different Text Representations 118
Figure 6.4. Test Set Ratio vs F1 Score of kNN Models 120
Figure 6.5. Max. F1 Scores of kNN Models for Different Text Representations. 120
Figure 6.6. Test Set Ratio vs F1 Score of Linear SVM Models 122
Figure 6.7. Max. F1 Scores of Linear SVM Models for Different Text Rep. 122
Figure 6.8. Test Set Ratio vs F1 Score of SVM-RBF Models 124
Figure 6.9. Max. F1 Scores of SVM-RBF Models for Different Text Rep. 124

Figure 6.10. Test Set Ratio vs F1 Score of LR Models 126
Figure 6.11. Max. F1 Scores of LR Models for Different Text Representations .. 126
Figure 6.12. Test Set Ratio vs F1 Score of TextCNN Models 128
Figure 6.13. F1 Scores vs Learning Stages of TextCNN Models 128
Figure 6.14. Test Set Ratio vs F1 Score of BERT Models 130
Figure 6.15. F1 Scores vs Learning Stages of BERT Models 130
Figure 6.16. Test Set Ratio vs F1 Score of RoBERTa Models 132
Figure 6.17. F1 Scores vs Learning Stages of RoBERTa Models 132
Figure 6.18. Comparison of Results of Shallow ML Models 133
Figure 6.19. Performances of Shallow ML Models vs Test Sets 134
Figure 6.20. W2V Word Embedding Impact on Models 135
Figure 6.21. Performances of DL Models vs Test Sets 135
Figure 6.22. F1 Scores vs Learning Stages of DL Models 136
Figure 6.23. Performances of Transformers according to Epochs Size 136
Figure 6.24. Highest Macro F1 Scores of Implemented Models 137
Figure 6.25. Comparison of Research and Project-Based Results 141
Figure 7.1. F1 Scores of RoBERTa Model-92 153

LIST OF ABBREVIATIONS

ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
ASTM	American Society for Testing and Materials
BERT	Bidirectional Encoder Representations from Transformers
BIM	Building Information Modelling
BoW	Bag of Words
BS	British Standard
CAD	Computer Aided Design
CBR	Case-Based Reasoning
CNN	Convolutional Neural Networks
CSI	Construction Specification Institute
DL	Deep Learning
EN	European Norm
FIDIC	International Federation of Consulting Engineers
FP	False Positive
GPT	Generative Pre-trained Transformer
ISO	International Organization for Standardization
kNN	K-Nearest Neighbor
LLM	Large Language Model

LR	Logistic Regression
ML	Machine Learning
MLM	Masked Language Model
NB	Naïve Bayes
NER	Named Entity Recognition
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
NSP	Next Sentence Prediction
OOV	Out of Vocabulary
POS	Part of Speech
Qty	Quantity
RBF	Radial Basis Function
RoBERTa	Robustly Optimized BERT Pretraining Approach
Spec	Specification
SVM	Support Vector Machine
TextCNN	Text Convolutional Neural Networks
TF-IDF	Term Frequency-Inverse Document Frequency
TP	True Positive
VSM	Vector Space Model
W2V	Word2Vec

CHAPTER 1

INTRODUCTION

1.1 Research Background

Construction is one of humanity's oldest endeavors, originating from the need for functional and controlled environments. As of 2013, more than 100 million people were employed in the construction industry, making it one of the largest value-adding sectors globally (Renz & Solas, 2016). Regardless of a country's level of development, construction processes are present in every corner of the world, whether in the form of new construction, renovation, rehabilitation, or maintenance. Having begun thousands of years ago, construction continues today, and it will persist in the future, involving numerous stakeholders—owners, customers, contractors, designers, consultants, suppliers, and service providers—at every stage of various construction projects.

Massive amounts of data in textual (e.g., contracts, manuals, specifications), numerical (e.g., calculations, sensor data), and graphical (e.g., CAD, BIM) forms are generated by all stakeholders before, during, and after a construction project (Bilal et al., 2016). Unstructured texts constitute most of this data due to the diversity of stakeholders, the multiple phases of a project, regulatory compliance requirements, and the extensive scope of information needed at each project phase (Ayodele & Kajimo-Shakantu, 2021; Friel, 2022).

Construction specifications are critical divisions of contract documentation, forming a framework that defines the technical properties of materials, execution methods, delivery, storage, quality control, testing, and reference standards for use in construction works (Kalin et al., 2011). Each project should have unique construction

specifications tailored to its specific properties, including construction type, method, location, time, budget, and contract delivery (C. Institute, 2013).

Such construction documents, including vast amount of data, are not free of problems and issues. For instance, in a study among court decisions, Babaeian Jelodar et al. (2015) determined that 24% of the cases involved problems with the contract documentation and breach of contract. Jagannathan & Delhi (2019) highlighted that 40% of claims related to change orders, cost or time overruns, delays, and conflicts caused by the misunderstanding of specifications, contracts and process. “Code or Specification Compliance of Work” and “Ambiguity in Contractual Interpretation” were identified as the predominant dispute reasons within the litigation related to airport projects in United States (Khalef & El-Adaway, 2024). According to a survey among construction contractors conducted by Zaneldin (2020), specification and drawing inconsistencies was titled as the main reason for construction claims.

Copy-pasting specifications, the lack of expertise of the specification writer or designer, misinterpretation of the scope of work or design, inefficiencies in language, and occasionally the intentional disregard of defective statements all contribute to the defects in specifications (Ibrahim et al., 2023). The preparation of construction specifications is a time-intensive task requiring subject-matter expertise and close coordination with drawings and design reports, that often took the primary focus of design teams, owners, and, in some cases, contractors in design-build project deliveries. Consequently, allocating sufficient time, human resources, and budget for developing construction specifications is frequently neglected. This leads to the use of specification templates or the practice of copying specifications from other projects to expedite the process. However, these shortcuts often result in conflicts with other documents, such as drawings; inadequate or contradictory requirements; and duplicated or outdated standards. The defective statements, involving ambiguity, vagueness, conflicts, and lack of required information, result in disputes among project stakeholders, increasing claims and risks in the project budget, quality, and schedule (Jackson, 1990; Lam et al., 2004).

Ambiguity arises when a statement has multiple meanings, or its interpretation varies depending on the reader. Vagueness, on the other hand, refers to a lack of clarity in a statement, making it unclear to everyone. Both ambiguity and vagueness are the key causes of misinterpretation of the requirements in construction specifications, potentially resulting in technical failures or disputes between the contractor and the owner or their representative. Conflicting requirements, whether within the specifications themselves or between specifications and other contractual documents, are another major source of project disruptions and claims. Missing necessary information in specifications can result in project interruptions or force the contractor to take initiative, potentially using materials and workmanship that fail to meet standards or project requirements.

The detection of the deficiencies in construction specifications poses a significant challenge for the construction industry. One reason for this is that, despite being the creators of specifications, consultants involved in the design phase often have less awareness of specification-related problems compared to contractors. Additionally, amendments and corrections implemented at the construction site, often in the form of change orders, are not typically communicated back to the design team, resulting in the same errors being carried over into future projects. The extensive number of pages in specifications, coupled with limited human resources and time allocated for their preparation, further hinders thorough documentation checks for deficiencies and proper organization. The quality of the specifications heavily depends on the expertise of the technical staff responsible for their preparation and review. This expertise requires not only a solid technical background but also an understanding of contractual contexts to ensure the text is free of ambiguous or unclear wording. Therefore, novice staff can also struggle to draft and review specifications effectively, increasing the likelihood of deficiencies, which can lead to technical failures, project delays, or disputes among stakeholders.

Natural Language Processing (NLP), which is a subfield of Artificial Intelligence (AI), has played an increasingly significant role in managing and generating documents in the construction industry over the past decade (Wu et al., 2022). NLP

techniques enable the processing and analysis of texts, offering opportunities to retrieve and extract valuable information buried within documents, as well as to classify semantic content for better organization and understanding of the documents generated by the construction industry. Safety reports, building codes and regulations, web data, contracts, construction documents, and bidding documents are among the most frequently utilized sources in implementing NLP (Baek et al., 2021).

1.2 Research Gap and Objectives

Existing studies based on questionnaires and interviews related to construction specifications primarily approach the subject regarding their usage, associated problems, the consequences of the problems, and practices for improvement. The specification problems stated by construction stakeholders are often presented in a generic manner, lacking a fundamental framework for deficiencies, classification, and a systematic approach to identify and address them.

Meanwhile, construction domain-specific NLP implementations are predominantly focused on areas such as safety management, document and knowledge management, contract agreements and risk management, and compliance with codes and regulations. The few studies that explore the application of NLP to construction specifications are limited to information retrieval and extraction for automatic compliance checking. None of these studies address the identification or analysis of deficiencies within technical specification documents.

This gap underscores the need for a more organized and data-driven frame to analyze and categorize specification defects comprehensively. Developing such an automated methodology could provide construction professionals with practical tools to enhance the quality and reliability of specifications by reducing the deficiencies, consequently the likelihood of project delays and disputes.

This research aims to create a structured, analytics-driven context for recognizing deficiencies within the technical sections of construction specifications and to

evaluate the effectiveness of supervised Machine Learning (ML) NLP models in accurately detecting and categorizing these defects. The ultimate goal is to develop an automated NLP-based method to assist design professionals in document quality control during the design stage and enabling owners and contractors to detect defects in bidding stage to anticipate project risks and mitigate potential future disputes effectively. The subobjectives of this study are to:

1. Develop a structured framework for specification deficiencies: Create a systematic, analytics-driven framework to analyze and categorize deficiencies in the construction specifications, addressing the lack of a fundamental structure in existing studies.
2. Integrate NLP into defect identification and categorization: Develop an NLP pipeline including various supervised ML models having different computational architectures and NLP techniques.
3. Evaluate the effectiveness of NLP models in deficiency recognition: Assess the performance of supervised ML models within NLP pipeline in accurately identifying and categorizing defects within construction specifications with evaluation metrics F1 score, precision and recall.
4. Examine the influence of dataset size and diversity on model accuracy: Investigate how variations in the size and diversity of the training dataset affect the performance of ML models in predicting defects.
5. Investigate the impact of project-originated defects on ML model performance: Evaluate the performance of supervised ML models when trained on datasets including project-specific defects and tested on completely new projects to understand the generalizability of the models.
6. Compare the performance of large language models in defect detection: Evaluate the performance of ChatGPT-4o in identifying and categorizing defects in construction specifications and compare its efficiency with domain specific trained models to understand the capabilities and limitations of Generative Pre-trained Transformers in domain-specific applications.

1.3 Overview of the Research Method

This section presents a brief narrative of the research on developing supervised NLP methods separating technical defect recognition from the experience of the staff and providing time savings for other critical subjects of project management. Detailed research methodology and steps are provided in Chapter 3.

Based on this study's objectives, the research problem was constructed on an NLP text classification pipeline, individually evaluating and categorizing each statement in a specification based on its inherent deficiencies. The research primarily focused on the recognition of deficiencies related to interpretation and usefulness of the specification text from a project management perspective. For instance, technical applicability of stated execution and installation methods or the conformance of the denoted technical properties of materials to specified standards were not evaluated. The data source of this research is comprised of 175 construction specifications belonging to 21 different architectural works from 16 construction projects. A total of 15,569 statements extracted from the dataset were reviewed one by one and labeled manually in four defect categories if existed. This labeled dataset was used for training and testing of 8 different ML algorithms such as Naïve Bayes, k-Nearest Neighbor, Logistic Regression, Linear Support Vector Machine, Support Vector Machine with Radial Basis Function, Text Convolutional Neural Network, BERT and RoBERTa. 6 various text representation methods, Bag-of-Words, TF-IDF, Google W2V, Expanded Google W2v, BERT embedding and RoBERTa Embedding, were utilized for vectorization of text data before implementing in ML models. The corpus, textual dataset, contains a total of 252,588 tokens (words, numbers or alphanumeric), of which 5271 are discrete tokens in 15,569 statements.

96 ML model combinations were developed and implemented on three different train-test set ratios. The results of models were assessed and compared according to macro F1 scores. Subsequently, the approach was validated using selecting the train-test dataset according to construction projects with the developed methodology.

Finally, a study was performed with ChatGPT-4o in defect recognition, comparing with results of main research models and project-based research models.

1.4 Thesis Structure

The thesis is structured in 8 chapters, including various sections as summarized in Table 1.1.

Table 1.1 Organization of Thesis

Chapter-1	• Introduction • Problem statement, Research Gap & Objectives
Chapter-2	• Literature on construction specifications and their significance • Literature on NLP, everyday applications, and implementation in the construction industry
Chapter-3	• Research design and stages • Definition of terms and information on techniques in NLP
Chapter-4	• Data collection and selection approach of the research • Labeling strategy and statistics • Labeling validation
Chapter-5	• Implemented text preprocessing and text representation techniques • Development of implemented ML models
Chapter-6	• Performance results of each ML model and their comparison • Validation of research methods with project-based data • Test of ChatGPT in defect recognition
Chapter-7	• Major findings and discussion of results • Error analysis of best ML model
Chapter-8	• Conclusion of the research • Limitations and recommendations

CHAPTER 2

LITERATURE

This study aims to merge two independent specialized subjects: construction specifications from the construction industry and natural language processing (NLP) from artificial intelligence (AI). The distinct characteristics of the two fields necessitate further identification of their knowledge backgrounds to establish a solid foundation for the research. This chapter briefly addresses the definitions of standard terms from both regions and emphasizes their roles in each domain. In the first half, the construction specifications, duties, and necessities are described. Subsequently, their roles in construction project management and the consequences and reasons for the specification deficiencies are brought forward. The second half of the chapter conveys the development and interaction of NLP in the AI world, its communal daily practices, and the studies that focus on its applications in the construction domain.

2.1 Construction Specifications

Documentation, either textual or graphical, plays a vital role in the construction industry even though the final physical product comes into prominence with its impressive appearance, like a skyscraper or a dam. It provides the chronicle of the work performed, the materials used, or the rules, the requirements, and the agreements reached between each role player in the field (Jackson, 1990). Many types of documents are produced in various phases of a construction project, including bidding, contract signing, design, construction, and handover.

Invitation to bidding, bidding instructions, main contract, general conditions, special conditions, drawings, specifications, and amendments are some document types developed in the initial stage of a construction project. Depending on the project size

and delivery, the amount and content of those documents may vary. For instance, the general and special conditions may be incorporated in the main contract for the construction of a small size house, or the development of technical specifications may be the responsibility of the construction contractor in a design-build project, where it is mainly in the designer's scope working for the client in design-bid-build projects.

2.1.1 What is a Specification?

Specifications, as a general term, are administrative and technical obligations defining the way of required works for a construction project (C. Institute, 2013). The work covers a broad range of tasks, extending from primary executive activities to the commissioning of a utility. Specifications can also define the non-technical tasks that must be accomplished during a construction process, such as expressing the minimum mobilization facilities for contractors, design restrictions and stages, document archiving, submission path, development of shop drawings, or even scheduling and monitoring.

Construction specifications are the term used in this study for the technical specifications directly related to the execution of construction tasks like reinforced concrete works, painting, electrical power cabling, or domestic water piping. They state and detail all requirements for materials, equipment, workmanship, and services to perform the related construction task. Construction specifications specify the type and properties of products to be supplied, their transportation and storage at the site, the tasks required for the execution of the materials, tolerances, inspection, testing, and commissioning if required. They also set compulsory regulations, codes, or standards for the manufactured products or while producing them at sites, like concrete, error limits, test parameters, and acceptable performance values.

Construction specifications are used to meet the quality expectations and requirements of the client under regulatory compliances from the final construction

(Ibrahim et al., 2023). They are the other technical information sources necessary to carry out the construction activities besides the construction drawings. Construction drawings are a way of graphical communication that projects the overall and detailed appearance, dimensions, positions, and assembly of the elements of the construction project. At the same time, the construction specifications tell the story of those elements and how they will be combined in a textual format. Construction drawings and specifications are cooperative documents; they should not be substituted for one another. For instance, the information and terms on both mediums must be consistent so as not to lead to contradictions; therefore, detailed material or execution definitions should be avoided in the construction drawings (Jackson, 1990).

A survey among 79 participants from owners, consultants, and contractors was conducted by Lam et al. (2004) to evaluate the use of specifications in the construction industry. 74% of the participants considered that the construction sector did not benefit from the ultimate capacity of construction specifications. The primary purposes of utilizing the specifications by the clients were to observe the quality performance of the contractors in the construction process and set up the technical requirements during the bidding stage, while for the contractors, it was to procure proper materials during construction and provide more efficient cost estimation during the bidding stage (Lam et al., 2004).

Ibrahim et al. (2023) performed a semi-structured questionnaire survey among 30 consultants and 30 contractors in Saudi Arabia to observe the practice motivation behind the construction specifications. The top three reasons for consultants were to verify the contractor’s submittals, to convey the coordination among the design-related consultants, and to evaluate and decide on the alternatives proposed by the contractor, respectively. The counter goals for contractors were to comply with testing and commissioning, assess the progress's quality level, and deliver the project's activities and technical requirements follow-up to project stakeholders.

2.1.2 Construction Specification Types

Kalin et al. (2011) describes four different types of construction specifications and a mixture of them based on The Project Resource Manual- CSI Manual of Practice (T. C. S. Institute, 2004). Each type has its own advantages and disadvantages, so the decision behind the selection of specification development has to be made according to the owner's requirements, project size and complexity, expected quality level, and construction budget.

Descriptive specifications, which declare the characteristics of the materials and their execution methods in detail without mentioning any brand name, are more suitable in cases where there is no specific code or standard for reference or if the intention is more stringent control over the construction work. An example statement is:

"Aggregate for normal-weight structural concrete: Maximum size not larger than 1/4 of narrowest dimension between forms, 1/3 depth of slab nor 3/4 of minimum clear spacing between individual reinforcing bars; maximum aggregate size shall be 20mm."

If any information is missing in the descriptive specification, it may lead to ambiguity that causes the technical staff to misinterpret it.

Reference standard specifications give more emphasis on referencing the commonly recognized standards like EN, BS, ISO, or ASTM or codes published by governmental or institutional organizations to define the requirements of the materials, execution, or testing like:

"*Fire Resistant Gypsum Board: EN 520, Type F, 1200mm wide and 12.5mm thick, longitudinal edge square, transverse edge tapered."*

Reference standard specifications are a widely utilized form of specification, especially for defining common materials and tasks. They considerably shorten the size of the document and state clearer requirements if defined precisely. That means

the referred standard should be valid, in charge, and also compatible with specified materials or works.

Proprietary specifications obligate the exact trade name of the product manufacturer, its model, type, and other specified properties. The manufacturer stated in the specification can be single or multiple trade names such as:

"*Epoxy Floor Coating: Sikafloor-264N, two-component colored epoxy floor coating with slip resistance finish. Color will be selected by the Owner.*"

In some cases, manufacturers other than designated may be allowed to be considered with "or equal" phrases. However, the meaning of "or equal" should be clearly defined as to how to evaluate the alternatives technically. Proprietary specifications provide better control over material selection and its execution and reduce the length of the text. However, they require a high level of knowledge of products and sometimes extend the preparation of the specification due to the selection period.

Performance specifications focus on the result rather than the way to reach there. The main difference between the descriptive specifications is that while descriptive specifications characterize the material properties and installations, performance specifications identify the features of the outcome, like:

"*Concrete flexural strength at 28 days shall be 4.5 MPa for rigid pavements.*"

Although performance specifications allow more innovative solutions to contractors, they are rarely utilized for a whole specification due to raised challenges in the control of all processes from procurement to commissioning (Thomas et al., 1995; Guo et al., 2005).

Mixed-type specifications do not limit themselves to a certain boundary of the above-mentioned types; rather, they incorporate and adapt accordingly to the document in view of the owner's needs and project, schedule, budget, and experience.

2.1.3 Significance of Construction Specifications

Construction specifications are a part of the contract documents binding lawfully the owner and the construction contractor (Jackson, 1990). Project delivery choice for the construction project also impacts this relation by shifting the development of specifications in project phases. For example, in design-bid-build project delivery, the development of the design, construction drawings, and specifications are the project owner's responsibility. Usually, the owner hires a design company to carry out design work, cost estimation and sometimes preparation of bidding documents. So that the construction specifications are literally a part of the contract, the contractors are aware of the situation and requirements in detail, and they prepare the price proposal according to the conditions in the design. However, the design development is transferred to the construction contractor in design-build project delivery. The owners either prepare a conceptual design including preliminary drawings and draft specifications or only the preliminary drawings and technical descriptions of the works without detailed specifications. Afterward, they hand over the design development scope, including final specifications, to the contractor under the control of a third-party consultant of the owner for monitoring and reviewing the design progress. In the latter case, although the final specifications are not a part of the contract, it is usually bonded with a contract in the scope of the design development. The contractor does not foresee the technical requirements in specifications in detail. Although this allows flexibility to the contractor in developing the specifications, there is also some blurriness that may result in a greater cost than estimated or may create technical or even procurement difficulties.

Therefore, it is expected that construction specifications should be systematic, comprehensible, and concise and should not cause misunderstandings and disputes among the project stakeholders (Erdis & Ozdemir, 2013). While enforcing conformance to technical requirements and the owner's expectations, construction specifications must avoid bringing unnecessary burdens on the construction project. Among the primary contributors to increased costs, delays, and project interruption

in the constructions are disputes arising from construction specifications (G. Lee et al., 2022). Badly drafted specifications do not only lead to deviating from requisites and faulty or unexpected final outcomes but also result in construction claims and litigations (Bennett & Blocker, 2016).

Babaeian Jelodar et al. (2015) conducted research on 101 court decisions in New Zealand and found that 24% of the litigations' reasons were related to the contract documentation problem and breach of contract. Another study states that 40% of claims which include change orders, cost or time overruns, delays, and conflicts due to misunderstanding of specifications and contracts, have contractual and process roots (Jagannathan & Delhi, 2020). Khalef & El-Adaway (2024) investigated 23 dispute cases in litigation regarding airport projects in the United States and revealed that "Code or Specification Compliance of Work" and "Ambiguity in Contractual Interpretation" are the two most common causes of disputes. It was detected that the specification and drawing inconsistencies are the main cause of construction claims, according to the contractors, based on a survey conducted among 43 construction stakeholders, 19 of which were participants from construction contractors in the United Arap Emirates (Zaneldin, 2020). A similar consequence was highlighted by Senouci et al. (2017) that design errors, including specifications, were among the highest causes of change orders resulting in cost overruns.

Major disagreements over specification requirements between owners and contractors ultimately damage both sides. Such disputes lead to delayed construction duration, add unforeseen costs from employing legal services for resolution, and injure the relationships between the parties, which negatively impact the entire project (Jackson, 1990).

Erdis & Ozdemir (2013) studied the root causes of specification-related disputes among 32 court cases and a questionnaire-based survey with 58 medium to large-size construction contractors. Disputes arising from construction specifications often derive from several joined issues. Technical specifications are frequently prepared quickly using a copy-and-paste approach, neglecting to incorporate current industrial

advancements or assess their relevance to market conditions. Specifications often include materials, technologies, and equipment without evaluating their suitability for the specific project or considering their physical, mechanical, and chemical properties under various conditions. Furthermore, ambiguous language in contracts and specifications often leads to misunderstanding, while certain clauses disproportionately favor either the employer or the contractor.

Specifications are the critical control mechanism in project execution by defining the minimum performance standards and construction methods. While carefully developed specifications enhance the construction efficiency and foster collaborative relationships among stakeholders at the site, the defects in specifications harm the construction quality and often result in disputes (Jackson, 1990).

Abrosimova et al. (2022) define the defects that occur in construction projects by examining various local and international legal legislations in three categories: pragmatic, formal, and consequential. The pragmatic definition focuses on the functional aspect of the defect, whether it decreases the value or fits the purpose. The formal definition is related to the conformity level to the stated requirements or standards. The last definition, consequential, emphasizes whether the defect causes real damage or harm to a product or person.

The survey conducted by Lam et al. (2004) also investigated the problems with defective specifications. The authors reviewed 14 possible lists of deficiencies in the literature and requested participants to evaluate the relative occurrence frequencies. Although clients, consultants, and contractors ranked slightly differently, the root causes of defects in construction specifications, incompatible statements to the project, complex and overcrowded sentences, ambiguity in clauses, and lack of coordination with other documents clustered at the top of the list. Additionally, contractors pointed out that the consequences of defective specifications result in delays in getting clarification and material approvals in another question of the same survey, which was also confirmed with the responses from the clients.

Ibrahim et al. (2023) repeated the same examination in their survey and encountered similar conclusions with Lam et al. (2004) regarding the causes of defective specifications. Contractors perceived the main defect as irrelevant clauses to the scope of the project due to copying the same specification developed for a preceding project. The non-locality of the constraints was another deficiency stated, which hindered the finding of the equivalence of stated materials in the local market and, consequently, the approval. Complex grammar of the statements and conflicts in conditions were also among the most influential reasons of defects determined by the contractors.

Numerous studies on litigation cases and the experience and reasoning of industry practitioners have revealed the reasons behind the deficiencies in construction specifications and their solid consequences. Ambiguous wording or vague language creates a misunderstanding of the statements by project participants. Discrepancies between different sections or drawings or other contractual documents and missing information put the contractors in contradiction. Irrelevant requisites or improper standards, codes, and regulations prevent approvals and procuring the most suitable materials. Such defects result in execution errors, reworks, unsatisfactory final outcomes, cost overruns, delays through disputes, and sometimes litigation between the owners and the contractors.

Sector experts and researchers placed plentiful recommendations and techniques to prevent the probable deficiencies raised in construction specifications and direct the specification preparation in a more systematic way.

Erdis & Ozdemir (2013) suggested that specifications developed by the owners should follow the physical and local market conditions, especially to improve flexibility in investment projects. It is extremely critical to allocate sufficient time to prepare specifications to prevent errors and conflicts with other contract documents. Ibrahim et al. (2023) advised utilizing a digital framework for specification reviews during the design stage involving contractors, owners, and consultants. The requirements in the specification statements, particularly the referred standards,

should be updated regularly and defined clearly to confirm with the design requirements. Construction specifications should state the requirements definitely and completely (C. Institute, 2013). The language of the specification should be simple, uniform, and free from excessive wording and ambiguity. The interpretation of the site staff should be prevented by clear descriptions and precise references. Material properties, procurement circumstances, installation methods, testing standards, and tolerances in specifications should be specific to each construction task of the unique project (Kululanga & Price, 2005).

Although the forms of observed defects in the specifications were determined competently, and a significant amount of them can be avoided by placing greater focus on the developing stage, it is challenging why the industry still suffers from those shortcomings.

Some of the challenges that make it difficult to resolve the problems were highlighted by C. Institute (2013):

- Time constraints: Creating comprehensive and accurate contract documents is time-intensive, leading to shortcuts in development and review. Although copying a whole or part of a specification from another project speeds up specification preparation, this alternative often leads to conflicts with other documents like drawings, creates improper requirements and conflicts in statements, and results in duplication or outdated standards.
- Reactive Corrections: Modifications are implemented on-site regularly in the form of change orders. Corrections made during construction phase do not feedback the design team which are not able to correct the errors for future projects.
- Document Organization: Disorganized contract documents related to specifications without a proper table of contents or with unrelated sections complicating the reaching of the correct file, increasing effort and time for staff.

- Owner's Contribution: A lack of understanding of the owner's expectations or a lack of information obtained during the final stages of design development can cause the requirements not to be incorporated totally and accurately.
- Specification Budget: Drawings are the main focal point of the design stage, either by the designer or the owner, who can comprehend the final product visually. Lack of budgeting and sharing enough time for specification development also leads to copy-paste solutions and improper documentation.

2.2 AI, ML & NLP

2.2.1 Origins of NLP

Artificial intelligence (AI) and machine learning (ML) are the two terms frequently used as synonyms in computer science which trigger the misunderstanding of the expressions (Keserer, 2024). Artificial intelligence is a broad term that covers machine learning. It can be defined as the simulation or representation of human intelligence and cognitive behaviors by a machine or a system for learning and solving complicated problems (IBM Data and AI Team, 2023a) such as recognition of photos, interpretation of speech tone and emotion, decision-making, solving complex probabilistic problems, or understanding natural language.

Machine learning (ML) is one of the subcategories of AI, that empowers a system to learn from data, discover patterns and relationships, improve its performance over time, and predict the targeted output values (Vajjala et al., 2020).

Moreover, neural networks are a subset of machine learning and are made up of layers, an input layer, one or multiple hidden layers, and an output layer. There are nodes in each layer that are simulated as artificial neurons and connected to each

other like in the brain. Nodes have weights and threshold values that are activated over the value and transmit the data to the nodes in the next layer.

The hidden layer in a neural network can be one or multiple in series. If the system has only one or two hidden layers, it is called a shallow neural network. Shallow neural networks are simple to construct and efficient to run on certain types of data and applications. When the data is getting dense and more understanding of complex and hierarchical patterns is required to extract, another approach enters the game; Deep Learning (DL). DL having more than two hidden layers is a division of neural networks that is capable of handling more complicated and high-dimensional data.

ML methods are primarily grouped into four main categories according to their data interpretation and problem-solving approach, as presented in Figure 2.1 (Lang, 2022).

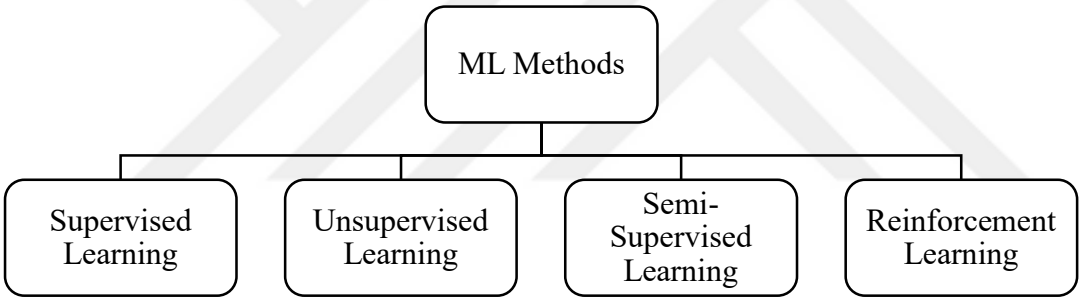


Figure 2.1. ML Methods (Lang, 2022)

The supervised learning method detects the output based on training the input labeled data. In supervised learning, the developed model tries to learn the relationship between the data and the label, which will also be the target to predict. In this context, the label can be any characteristic that defines the parameters and has a relationship with the data.

The unsupervised learning method works on unlabeled data and estimates the output without supervision. This method does not use labeled data as input. In contrast, the objective is to recognize the patterns, structures, or rules inside the data set for the intended purpose.

The semi-supervised learning method falls between supervised and unsupervised learning methods. Its aim is to generalize the rules and relationships on huge amounts of unlabeled data by training on a very small amount of data.

The reinforcement learning method involves learning the patterns and structures through a feedback-based process, receiving rewards for good, and getting punished for bad. There is no requirement for labeled data as supervised or semi-supervised learning methods to initiate the model. The maximization of cumulative reward is the driving mechanism of the technique.

Various algorithms with different complexities, according to their computational power requirement and parameter settings in model development, are employed in ML methods. Some are generally utilized only in one of the methods described above, while some can be used in more than one approach.

Support Vector Machines (SVM), Naïve Bayes (NB), Linear/Logistic Regression (LR), Decision Trees, Random Forests, and k-Nearest Neighbor (kNN) are examples of algorithms that are mostly used in developing supervised learning models.

k-Means, Principal Component Analysis, Latent Dirichlet Allocation, and Hierarchical Clustering are a couple of algorithms for developing unsupervised learning models.

Although the semi-supervised method usually employs other method's algorithms, there are some developed specifically such as Semi-Supervised Support Vector Machine, Transductive Support Vector Machines.

Neural network-rooted algorithms have broader applications and can be used in all of the described approaches above such as Recurrent Neural Networks (RNN), Long Short-Term Memory Networks (LSTM), Convolutional Neural Networks (CNN), Transformers in supervised learning, Word2Vec, GloVe, FastText, Autoencoders, Transformers like Generative Pre-training, BERT, RoBERTa in unsupervised learning, Ladder Networks, Semi-Supervised Generative Adversarial Networks in

semi-supervised learning, Deep Q-Networks, Deep Deterministic Policy Gradient in reinforcement learning.

Natural Language Processing (NLP) is another subcategory of AI that deals with analyzing, modeling, understanding, and interpreting human language (Vajjala et al., 2020) as demonstrated in Figure 2.2. The field of NLP is not limited to comprehension of the language; it also includes manipulation and the generation of human language.

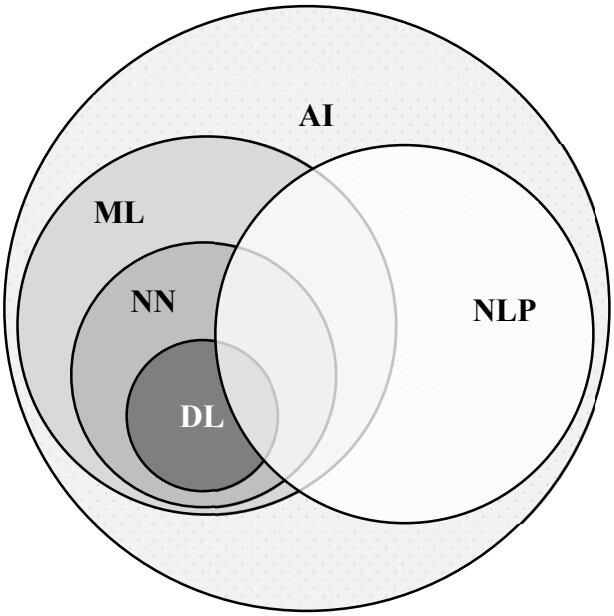


Figure 2.2. Relations between AI, ML & NLP (Vajjala et al., 2020)

NLP approaches can be classified into two broad categories: Rule-based and ML-based (Nailman, 2024).

The rule-based approach incorporates extracting domain-specific patterns and relationships in data, and the development of systematic rules manually and application in the specific field of NLP. Although the rule-based approach performs with high precision when the rules are well-designed, it demands a substantial manual workforce and time (J. Lee et al., 2019). The developed rules are interpretable and under the developer's control, but they lack flexibility, adaptation, and generalization. Also, it requires a high knowledge of domain and linguistics.

Meanwhile, NLP substantially benefits from the advantages of ML methods and the algorithms described above in various stages of its pipeline. The model development spreads from the utilization of statistical ML algorithms to DL algorithms based on the purpose of NLP implementation.

Various NLP applications repeatedly show a common set of key properties. Below is a list of frequent conceptions behind and its widespread NLP applications.

Language modeling is the task of predicting a character in a word or the word in a sentence or a whole sentence based on the previous data, such as auto-correction and auto-completion.

Text summarization is the shortening of larger documents by keeping the overall meaning of the text.

Information extraction is the task of mining the required or valuable information from the text such as keyword, relationship, or calendar event extraction.

Information retrieval is the effort of finding out the query from a large data set like search engines and systems, question and answering.

Text classification is the task of categorization of a word or sentence into predefined groups like sentiment analysis, spam filtering, language detection, and document classification. Topic modeling is a version of text classification in which the model is unsupervised. It employs unsupervised algorithms to extract the topic of the document. Named Entity Recognition (NER) has a similar idea to text classification to label the words or phrases in a sequential strategy.

It is difficult to categorize all NLP applications according to their fundamental tasks, as the development of NLP applications leads to combining different tasks, ML methods, and a mixture of various algorithms in different levels for smart chatbots and voice assistants, machine translation, speech recognition, and predictive text generation.

Whatever the NLP application is, there are similar fundamental steps in an NLP pipeline as shown in Figure 2.3. The progress chart shows the basic stages prerequisite for a simple NLP application. There may be other intra-steps that are joining at any stage of the pipeline. The real-life scenario is not a straight line, as shown; there are cycles inside or among stages.

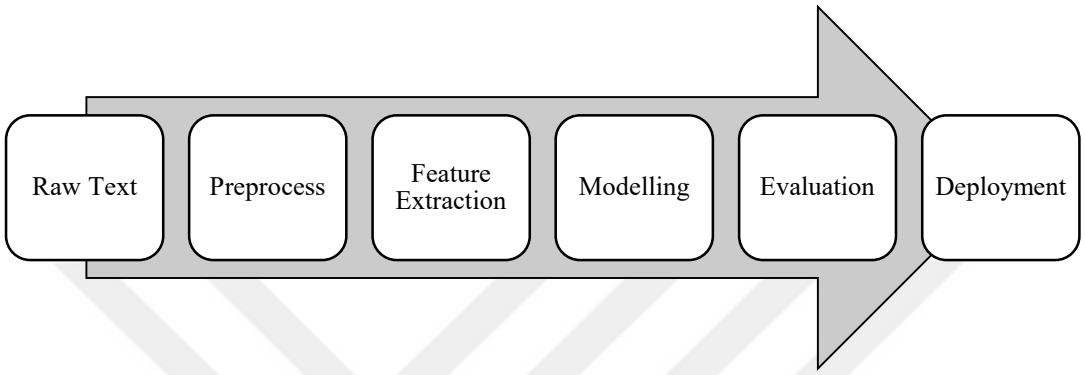


Figure 2.3. Basic NLP Pipeline

The initial step is obtaining raw text from digital documents, web pages, e-mails, customer reviews, social media posts, chat logs, forums, and even hard copy publishing.

The next step is cleaning the text data, removing noises and preprocessing with techniques like lowercasing, POS tagging, punctuation/number removal, or lemmatization to prepare for feature extraction.

Feature extraction is the transformation of preprocessed data into a representation in which applied algorithms recognize the inputs to develop a model. Still, in a rule-based approach, a preprocessing stage is required to simplify the text and convert it into a workable format.

Modeling is the stage of developing and implementing NLP models for the purpose.

The performance of the model is monitored and the decision to move forward or return to the previous stages for modification and update is decided in the evaluation stage.

Deployment is the final stage, servicing the developed NLP application in real life.

2.2.2 NLP in Common Practice

The development of computer science has greatly advanced NLP by turning it into a key part of daily life. ML, specifically neural networks, enables models to learn language patterns and handle large amounts of text data by introducing new NLP features (Khurana et al., 2023). Launch of semantic word embedding of Word2Vec by Mikolov et al. (2013) accelerated the use of DL algorithms by starting a powerful method for representing words as dense vectors in contrast to sparse, capturing semantic relationships, and enabling more effective models for tasks like language translation, sentiment analysis, information extraction, and text classification. Improvements in DL models remarkably the transformers, such as BERT and GPT, thrusted NLP implementations further, making tasks like machine translation, text summarization, and chatbots more accurate (Koroteev, 2021). With better computing power and smarter algorithms, NLP has become more effective, shaping how humans interact with technology and communicate with each other.

Search engines have become an indispensable part of modern life. They not only utilize NLP-based information retrieval techniques to search the entire internet for a single keyword but also aim to predict the semantic intent behind (Ghazizadeh & Zhu, 2021). Search engines often benefit from the autocomplete feature of NLP by recommending the whole word and predicting the entire sentence that the user intends to type, thereby enhancing search efficiency and improving user experience (Verma et al., 2023). Autocorrect is another NLP method to detect misspelled words and advise the best possible form of the word. This feature, along with autocomplete, is widely utilized not only in search engines but also in messaging software, online forms, language learning tools, word processors, and coding platforms (Balush et al., 2021).

Chatbots and smart assistants like Alexa by Amazon, Google Assistant, Siri by Apple, or Bixby by Samsung are other popular examples of NLP that interact with humans in everyday life. These applications understand the user's commands or questions, interpret the intents, and respond back accordingly in a conversational

interface (Weber & Ludwig, 2020). Chatbots are widely used in customer service of commercial trades by receiving inquiries from consumers, resolving issues, providing feedback information, or recording complaints without the support of any human intervention (Shafi et al., 2020). Smart assistants go further by easing with tasks like setting alarms, controlling smart devices, using search engines, opening a web page, music and video playing applications, and giving real-time updates like weather forecasts (Lahiri et al., 2023). Both cases, particularly smart assistants, highly utilize the speech recognition approach of NLP by transcribing the human dialogue into digital textual representation before recognizing the human's command (Haeb-Umbach et al., 2019).

The expansion of digital communication, social media, and e-commerce has transformed traditional face-to-face product surveys into sentiment analysis in a digital environment. The user's opinion on a specific product or service or even a political decision on reviews, forums, web pages, or social media can be recognized and assessed the attitudes by sentiment analysis methods of NLP (Wankhade et al., 2022). The digital transformation provided advantages in both optimizing the cost and time resources for data collection and offering a more detailed understanding of customer preferences, emotions, and trends in a more dynamic and reactive environment that rapidly adapts the strategies and offerings to meet customer demands (Pascucci et al., 2023).

Machine translation is an NLP application that is designed to translate text or conversation by integration of speech recognition from one language to another without human assistance (Jiang & Lu, 2020). NLP breaks down language barriers in the global world by making language translation easier. Modern translation models are better equipped to understand the semantics, context, and syntax of text, enabling more accurate translations (Ranathunga et al., 2023).

2.2.3 NLP in the Construction Domain

The construction industry has benefited from the integration of digital technology and AI, which has transformed construction processes to become more reliable, automated, cost-effective, and efficient, particularly in managing complex, uncertain, and data-intensive problems, while providing notable solutions compared to traditional methods. Pan & Zhang (2021) grouped the study of AI in the construction industry under 6 headings by reviewing 4473 journal papers related to construction engineering and management published between 1997 and 2020, such as process mining, intelligent optimization, knowledge reasoning, information fusion, computer vision, and natural language processing.

Wu et al. (2022) investigated that the topic of NLP began to attract the attention of researchers in the construction industry in the early 2000s, with a slight increase in published journal articles up until 2015. Starting in 2016, the implementation of NLP techniques in the construction field has grown rapidly.

The clustering of associated topics within the construction domain in research has been observed in document and knowledge management, safety management, automated compliance checking, building information modeling, contract risk management, and others, including social media analysis (Ding et al., 2022). Commonly implemented NLP techniques across these studies include classification, information retrieval, information extraction, topic modeling, and Named Entity Recognition (NER).

The majority of textual data in construction domain studies originate from safety documents (accident & injury reports, etc.), codes (international building code, energy conversation code, etc.), web data (former Twitter, news articles, etc.), contracts (contract documents, FIDIC), construction documents (complaints, work descriptions, etc.) and engineering documents (CAD, BIM, etc.), bidding (request for information, request for proposal, etc.) and claim documentations (change orders, litigation cases, etc.) (Baek et al., 2021).

A vast number of textual documents are being created and stored in construction management systems, necessitating handling and organization without human intervention. One of the earliest document classification applications was conducted by Caldas & Soibelman (2003) who developed various shallow ML models to classify 3030 arbitrarily selected product documents within the hierarchical organization of CSI MasterFormat. Similarly, Al Qady & Kandil (2015) studied the classification of 77 claim files belonging to the construction of an airport, categorized into eight claim subjects by using basic text representation of TF-IDF with shallow ML classifier algorithms.

As digitally recorded reports continue to expand within the construction industry, the development of more efficient and reliable methodologies for document management becomes crucial. Baker et al. (2020) categorized over 90,000 safety incident reports generated between the early 2000s and 2018 with the help of both shallow and deep learning ML algorithms. Four different classification combinations, each reflecting distinct characteristics of an incident, were supervised, such as the incident type describing the nature of the incident, injury type, body part denoting the impacted part of the body, and severity of the injury.

Reaching valuable information, specifically inside the unstructured data, is at the heart of knowledge management. Information extraction approaches of NLP are advanced to dig deeper and uncover the hidden knowledge, transforming raw texts into beneficial understandings and fostering more efficient decision-making. Once the data for training an ML model is not enough for achieving accurate predictions or generalization, semi-supervised ML models come into play, leveraging both labeled and unlabeled data to improve performance and overcome data scarcity challenges. K. Liu & El-Gohary (2017) built such a methodology in order to automatically extract information from bridge inspection reports. The Named Entity Recognition (NER) approach in NLP was employed to identify 11 entities from the reports. A small set of labeled data was used to infer the entity classes of unlabeled data through semantic similarity. The combined dataset of inferred and human-

labeled data was then used to train an ML model to predict the entities in the unlabeled data set. Labeling 10% of the test data achieved considerable accuracy.

The use of NLP in safety management is another trendy topic, aimed at better identifying the root causes of incidents and accidents and developing more effective preventive measures in the construction industry. Tixier et al. (2016) developed an NLP method to reduce the need for manual exploration of unstructured injury reports, enhance organizational safety knowledge, and advance safety management practices. A rule-based approach was developed to extract information specific to 101 safety-related variables, 80 attributes, 9 energy sources, 7 injury types, and 5 body parts from 140 randomly selected incident reports out of a total of 2,201 which were also used for developing rules and keyword dictionaries.

Case-Based Reasoning (CBR) is a tool used to detect and reduce accident risks in future cases by analyzing previous similar instances. Zou et al. (2017) built a model that integrates the information retrieval approach of NLP into Case-Based Reasoning (CBR) to retrieve similar risk cases from a dataset of 590 previous records based on an input query. A previously developed set of keywords related to accident risks was enhanced by identifying the most similar words on Wikipedia, with the aim of expanding the corpus dictionary. When a query is input into the model, it first checks its own library. If the query is not found, the model then searches WordNet for synonyms. The top ten most similar cases were retrieved from the dataset by measuring the similarity between the expanded queries and the documents.

T. Kim & Chi (2019) employed information retrieval and information extraction techniques to establish a knowledge management system for construction accident cases. The system incorporates a semantic retrieval model that processes input queries to search and rank 4,263 accident case reports based on their semantic similarity, utilizing a specially developed accident corpus. Hidden information was extracted based on query ranking and categorized into four groups: hazard object (e.g., formwork, scaffolding), hazard position (e.g., high place, temporary facility), work progress (e.g., excavation, transport), and accident result (e.g., collapse, fall).

This extraction was achieved using a combination of rule-based methods and ML algorithm.

A decision support system integrating NLP and BIM was developed by Khan et al. (2024) to forecast the risk of accidents according to various construction tasks. Data from 11,962 accident cases were grouped into 12 construction activities, such as roofing, foundation work, scaffolding, and plastering, and four accident types: falls, struck-by incidents, electric shocks, and confined spaces. Using an unsupervised NLP algorithm, K-means, the interrelation between construction activities and accident types was established through keyword-matching techniques. This information was then transferred to a BIM-4D environment, which visually highlights the frequency of each accident type associated with specific construction activities based on historical data.

The implementation of NLP enhances the early identification of hidden risks in contracts, providing valuable support for contract management during the bidding stage of a construction project. J. Lee et al. (2019) assumed modified clauses in FIDIC standard conditions as "poisonous clauses" that could pose risks to contractors. To address this, a rule-based system was developed to parse clauses into subject-verb-object format and check for risk-related terms using a lexicon enriched with WordNet. The model evaluates each sentence to determine whether it semantically aligns with predefined risk extraction rules to support the existing contract-management tasks.

Risk identification by utilizing NLP not only finds an opportunity in contract documents but also through other issued publications to provide a knowledge basis for a starting line of risk management. Gao et al. (2024) generated an NLP model to detect risky sentences in news articles related to construction projects and industry. A risk term dictionary was created using news articles with a literature review and brainstorming sessions and was further expanded by adding synonyms with the assistance of WordNet. 1,122 out of 2,445 news articles published across various sources, were identified as containing risky paragraphs based on the previously

developed risk thesaurus. After manually assessing each sentence within the paragraphs as either risky or not, the dataset was trained and tested using various ML algorithms, ranging from shallow models to transformer-based approaches.

Hamdy et al. (2024) conducted a study on contract management that incorporated both risk identification and risk assessment. A dataset of 486 clauses from five construction contracts was used to implement a deep learning-based NLP model. Each clause was evaluated according to five risk severity levels, ranging from very low to very high, and categorized into four possible impact types: time, cost, both, or none. The labeled dataset was then trained and tested using a developed Artificial Neural Network (ANN) model, which evaluates each clause and predicts its risk category and expected impact.

Another noteworthy study was observed on the categorization of litigation judgments related to construction disputes into corresponding law articles in China, aiming to assist in fostering dispute strategies for contract management. 10.000 court judgment descriptions having multilabel law articles were involved in the development of classification models by Zhong et al. (2024). Various ML algorithms were trained on judgment texts linked to relevant law articles and tested to predict the applicable law articles based on input case descriptions.

The use of NLP in contract management is not limited to risk or dispute identification and management; it is also applied to requirement extraction from contract documents, a time-consuming and challenging task in understanding clauses not to advance to disputes. Hassan & Le (2020) conducted research on recognizing requirements in contract documents using two different approaches: rule-based methods and machine learning classification. 1787 statements from seven different contracts were labeled as requirement or nonrequirement: Requirements encompass essential obligations and compulsory directives, whereas nonrequirements cover supportive instructions and contextual elements, such as definitions and headings. The dataset was treated as a classification problem and trained and tested using various ML algorithms. Additionally, a list of keywords was extracted to identify

requirements, and a rule-based technique was applied to separate statements reflecting requirement characteristics.

Codes, regulations, and specifications are valuable technical sources in the construction industry that define minimum acceptable standards for safety, quality, and performance, legally binding directives that enforce compliance with the law, and detailed descriptions of requirements for design, material, execution, and testing. Comprehension of the necessary information within these documents is essential not only for fulfilling contractual obligations but also for ensuring compliance with technical requirements that guarantee the satisfactory performance of the final product. Gathering the required data from the crowded documents is a challenging and time-consuming task that entails experience in documenting and processing the text.

J. Zhang & El-Gohary (2016) formulated a rule-based NLP model to extract essential information from construction codes, providing foundational data for automated compliance checking. Two chapters of the International Building Code were used to extract rules and patterns, which were then tested on a separate chapter of the same code. Statements were processed through an NLP pipeline that included POS tagging and phrase structure analysis. Information extraction rules were applied by comparing the tagged phrases with a predefined gazetteer. From this process, nine instances were extracted from the statements: subject, subject restriction, compliance checking attribute, deontic operator indicator, quantitative relation, comparative relation, quantity value, quantity unit, and quantity restrictions.

Another study to evaluate the machine interpretability of regulations or building codes for automatic compliance checking was conducted by Zheng et al. (2024). Seven clause categories of direct, indirect, method, reference, general, term, and others in three interpretability levels of easy, medium, and hard were categorized. An initial set of 1,350 clauses labeled in seven categories was expanded to 1,450 labeled clauses through the implementation of data augmentation techniques. The pre-trained and fine-tuned BERT model was trained, validated, and tested using the

developed dataset. Finally, the trained model was applied to over 150 building codes to evaluate document interpretability levels, by assigning different scores to clauses representing the interpretability levels to calculate the overall score of the document.

A DL-based NLP model extracting construction sequence constraints from regulatory documents was proposed by Zhong et al. (2020). The model was structured in two stages: first, recognizing knowledge entities, and second, classifying the relationships between those entities. The dataset consisted of 40,000 sentences from 14 types of national standards within the Code for Acceptance of Construction Quality in China. Among these, 400 sentences were classified into one of five target categories, such as "*When construction procedure X is finished, procedure Y should be conducted immediately.*" The model first recognized five entities in each sentence, including two procedures, two objects, and one interval of time. Subsequently, the relationships between these entities were classified into one of the five predefined categories.

Utilization of the Named Entity Recognition (NER) approach of NLP to extract required data from technical documents has also been observed in other studies, similar to previous research. Moon et al. (2021) developed a model to identify entities within 56 construction specifications related to road construction projects. The words in 4,659 sentences were tagged as six different entities based on information types, such as organization (e.g., contractor), action (e.g., shall test), element (e.g., design value), standard reference (e.g., ASTM), and none (e.g., for). The developed DL model was trained on 70% of the dataset and tested on the remaining 30% to predict the entities within the sentences.

Moon et al. (2022) also conducted a study on identifying risk categories within the same dataset. Seven contractual risk categories were compiled based on a literature review: payment, temporal, procedure, safety, roles and responsibilities, definitions, and references. A total of 2,807 specification statements were labeled with all applicable risk categories. The dataset was then trained and tested using various ML models, including BERT.

Studies on construction specifications, typically based on questionnaires and interviews, focus on their usage, associated challenges, consequences, and suggested improvements. However, the issues highlighted by construction stakeholders are often generalized, lacking a structured framework for identifying, classifying, and systematically addressing deficiencies. Besides, while limited studies have applied NLP to construction specifications, these efforts are restricted to tasks like information retrieval and automatic compliance checking, without addressing the identification or analysis of specification deficiencies.

This gap demonstrates the need for a structured, data-driven methodology to analyze and categorize specification defects systematically. An automated NLP-based approach could offer construction professionals practical tools to reduce defects and minimize project delays, related cost overruns and disputes. The research aims to develop an analytics-driven framework utilizing supervised ML NLP models to identify and classify deficiencies in construction specifications, addressing the methodological gap outlined in the next chapter.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Research Design

This chapter presents an overview of research stages, focusing on the definitions and descriptions of terms and methods related to the NLP process. Following this brief introduction, the research-specific application of each stage is discussed in Chapter 4: Data Acquisition and Preparation and Chapter 5: Development and Implementation of ML Models.

The primary objective of this study is to investigate the capability of NLP in identifying and classifying defects within construction specification documents. The research approaches this by combining two separate routes: development of a defect framework based on existing literature and development of a text classification NLP pipeline, where each statement in a specification was reviewed and classified individually based on its deficiency. A detailed research flowchart, including an adapted NLP pipeline from generic version (Figure 2.3) is presented in Figure 3.1.

The study initiated with a literature review. Then, it was decided to observe and comprehend the textual data that was the basis of the study. Therefore, selection and text cleaning of the specifications were carried out before the development of defect framework. Following this, the defect framework was created and accordingly the statements were reviewed and labeled manually, and a validation step confirmed the accuracy of the labeling process. Subsequent steps involve common NLP tasks: text preprocessing, feature extraction, model development, and evaluation. The study focuses exclusively on supervised machine learning approaches using distinct computational engines. A comparative analysis was conducted to identify the overall best-performing ML model.

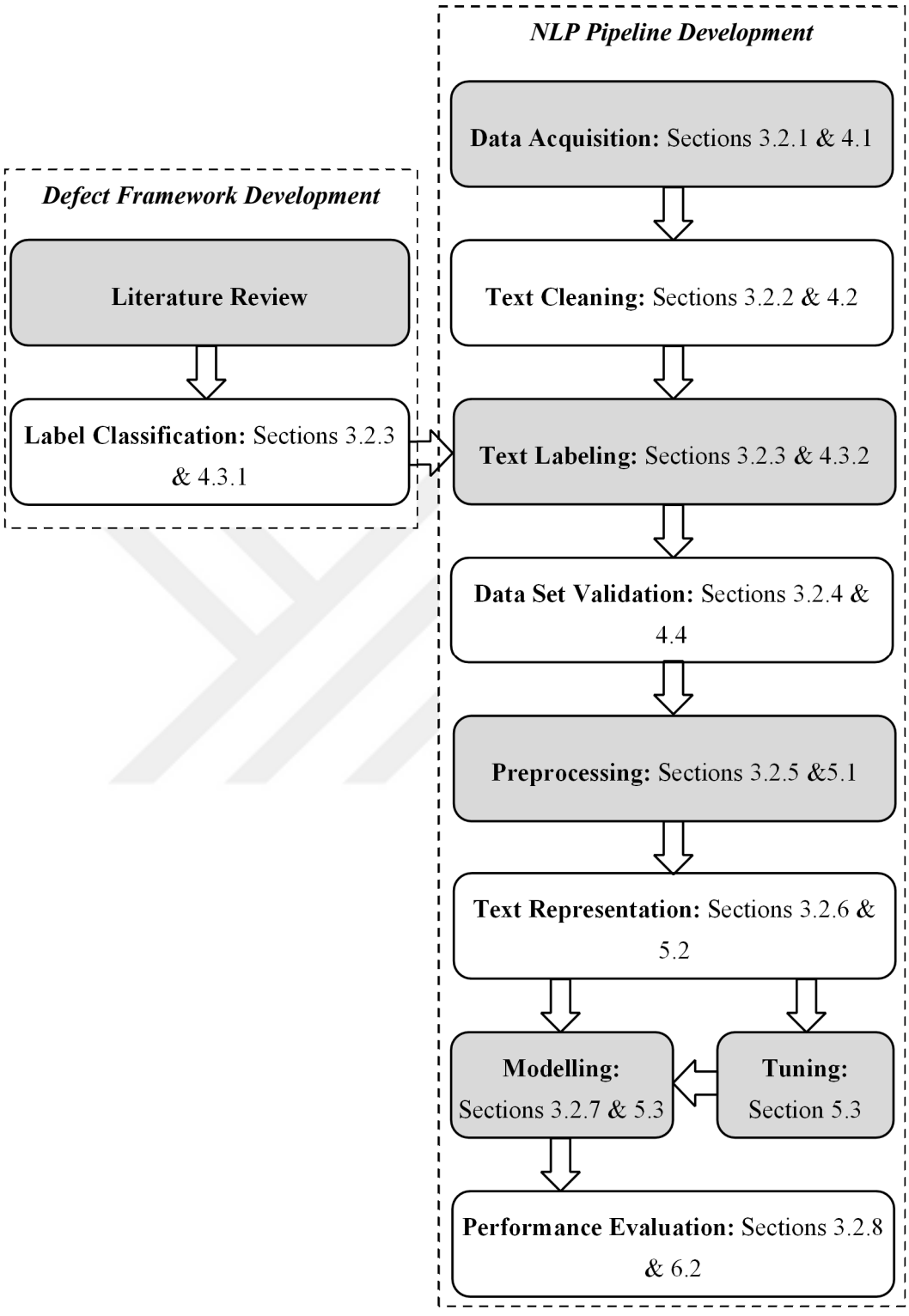


Figure 3.1. Research Flowchart

Next, a project-based study was performed using selected models to verify the practical execution of the research. Finally, an investigation was carried out to evaluate the performance of the ChatGPT without prior training on domain-specific data.

3.2 Research Stages

3.2.1 Data Acquisition

Data acquisition, the collection of construction specifications and filtering, is the first research stage for developing an NLP pipeline after maturing the problem statement and research objectives. Since the target of this study is to detect and classify the defects among the statements of a construction specification by implementation of ML algorithms, it is highly critical to construct the foundations of the models with relevant and enough data.

The relevancy and size of the data are interconnected with each other to provide successful domain-specific ML models. Although ML benefits from transfer learning, which is practicing a model with common training in another completely distinct domain, it still needs to fine-tune the pretrained model with some domain-specific data. Details of data acquisition, the strategy, reasons, and raw data statistics are given in Chapter 4.1 Data Collection and Selection.

3.2.2 Text Cleaning

Mostly, texts are not directly ready for use as data sources in ML Models. They must be cleaned up before the preprocessing stage. Raw text that is essential for ML models must be separated from the noises, which are parts that do not contribute to the results and could have adverse effects on the performance of the ML models.

Therefore, a text cleaning process for selected construction specifications must be implemented before the start of the labeling stage, both for aiding to label the statements under control and to prepare the data for preprocessing.

3.2.3 Text Labelling

Text labeling is literately the development of the train-test set of the ML models. Simply, it is the process of labeling the whole data, in other words, the statements of the specifications according to some ascertained defect classes.

Since the basis of this research is mainly an identification and a classification problem, it is the primary step to decide the classes explicitly defects before the start of labelling. The literature review, conducted to identify the defects observed in construction specifications, is used for defect classes, which form the basis of the labeling of each statement. More information on internal steps is given in Chapter 4.3, Sentence Labeling.

3.2.4 Data Set Validation

Validation in this research is to assess the accuracy of the manually labeled data by the researcher before utilizing in the machine learning models as train and test data. Although the deficiency categorization, definitions, and principles are grounded over previously published research and paper works, the confirmation of labeling is still required to bind the study to the objective truth.

Briefly, a group of professionals experienced in the construction industry other than the researcher review some of the sampled specifications and label them according to predetermined defect groups based on their experience and technical judgment. Details of the validation step are presented in Chapter 4.4 Labeling Validation.

3.2.5 Preprocessing

The preprocessing stage of NLP is a vital internal phase that directly affects the performance of the NLP pipeline. The tasks that are performed under the text cleaning heading are more broad efforts converting the document ready for preprocessing, mostly composed of modification and removal of page and paragraph formats. Whereas the preprocessing stage has analytical exercises at sentence and word level. It includes several procedures in various complexities, from the simplest lowercasing to an advanced level like POS tagging.

Preprocessing provides ML models and algorithms to extract significant patterns more precisely by reducing the noise that obstructs performance. This formulation also enhances the comparability of numerous texts, which is crucial while developing models that are required to recognize and predict categorization.

Another advantageous assistance of preprocessing is reducing the dimensionality of the models. Text data is a collection of words that is accounted as a dimension in some text representation methods. The more different words in the text represent the higher dimension of the representation which creates complexities in handling data and slowing down the running of models. Many of the different forms of the same word do not have any contribution to the performance of the ML models especially in NLP classification problems. For instance, some commonly used words like “the, a, of, in” etc. or the tense and plural form of suffixes like “-ing, -ed, ‘s, -es” etc. may not be remarkably beneficial to the functioning of the NLP pipeline.

The preprocessing stage facilitates the removal of irrelevant information and noise that may poorly influence the training and testing of NLP models. Nonetheless, it is crucial to carefully consider the selection and implementation of preprocessing techniques. In the pursuit of enhancing the effectiveness of machine learning models, adverse impacts must be avoided.

Below is the list and brief description of some common preprocessing techniques implemented in NLP.

Sentence Segmentation: Sentence segmentation is the determination of sentences from paragraphs and separation into individual sentences. Those segmented sentences may be built up from one word like headings or contain multiple words as a regular sentence. Sentence segmentation does not depend on the voice of the sentence, such as active or passive, or functional type of the sentence like declarative or imperative. Dividing considerably long compound and complex sentences into simple ones may also be advantageous on occasion.

As an example, after the sentence segmentation task, the paragraph "*Mix materials in mechanical mixers except finish coats containing lime may be hand mixed. The mechanical mixers must be an approved type that accurately and uniformly controls the quantity of water. When mixing by hand, mix dry plaster aggregate to a uniform color in the mixing box, add water, and hoe the plaster immediately into the water and mix thoroughly to a proper consistency.*" is separated into three tokenized sentences like:

1. *Mix materials in mechanical mixers except finish coats containing lime may be hand mixed.*
2. *The mechanical mixers must be an approved type that accurately and uniformly controls the quantity of water.*
3. *When mixing by hand, mix dry plaster aggregate to a uniform color in the mixing box, add water, and hoe the plaster immediately into the water and mix thoroughly to a proper consistency.*"

Word Tokenization: Similar to sentence segmentation, word tokenization is the splitting of a sentence into isolated words and phrases namely tokens. It is a required step to perform other preprocessing techniques such as stop word removal, stemming or lemmatization.

The separated sentence "*Mix materials in mechanical mixers except finish coats containing lime may be hand mixed."* is transformed into a list of tokens as "*Mix*", "*materials*", "*in*", "*mechanical*", "*mixers*", "*except*", "*finish*", "*coats*", "*containing*", "*lime*", "*may*", "*be*", "*hand*", "*mixed*","."."

Word tokenization also includes the separation of the numbers, symbols, abbreviations, and punctuations within the sentence. One critical point is the segmentation of phrases like in the sentence "*Provide asbestos-free materials.*". Tokenization of "*asbestos-free*" could be potentially different than "*asbestos*" and "*free*". Word tokenization is a domain-specific task that could be dependent on the subject that the NLP pipeline is built on (Vajjala et al., 2020) .

Lowercasing: Lowercasing is converting all words in a text to lowercase format to normalize the text data and decrease the vocabulary size. "*Mix"* and "*mix"* are accounted for two distinct tokens that don't have semantic differences other than generating extra burden on ML algorithms in NLP applications, especially in classification problems. Hence, lowercasing simplifies the data handling and improves model generalization.

Punctation and/or Number Removal: Punctuation marks like periods, question marks, commas, parentheses, and numbers do not have remarkably significant contributions to the performance of the ML models. Some NLP applications like sentiment analysis or machine translation could benefit from punctuation marks, but text classification may not. Since both punctuation marks and numbers introduce additional tokens that increase the size of vocabulary, a thorough assessment should be conducted to determine if the numerical data in the text is essential for the NLP model.

Stop Word Removal: The most frequent words used in English such as "the, in, of, a, and, or, to" etc. are termed stop words (Vajjala et al., 2020). Stop words do not have independent semantic content that contributes to the overall representation of a sentence. There isn't a dedicated list of what stop words are. They must be compiled according to the type of NLP application in which a classification problem may have a different set of stop words than a sentiment analysis. Therefore, the set of stop words should be carefully selected so as not to lose valuable aspects in a sentence.

Stemming: Stemming is the task of removing the suffixes of a word and converting it to a root form. For instance, the sample sentence of "*mix materials in mechanical*

mixers except finish coats containing lime may be hand mixed" is transformed into "*mix materi in mechan mix except finish coat contain lime may be hand mix*" after the stemming process. All words "*mix*", "*mixers*" and "*mixed*" are stemmed to "*mix*" word which has a sense and also the originals are semantically connected, whereas the words "*materials*" and "*mechanical*" are rooted as "*materi*" and "*mechan*" which are in unusual forms. The obtained stem or so-called root is not always a meaningful word. The aim of stemming is to improve the model performance by grouping similar words and standardizing them regardless of their articulation. However, the stemming process may lead to overstemming where there is a loss of distinction between two words as shown in the above example. Both "*mixers*" and "*mixed*" are stemmed to the word "*mix*" but one of them defines machinery the other is the action to be completed. Therefore, the interpretation of the stem may give rise to confusion.

Lemmatization: Lemmatization is another process to convert words into their root or lemma while keeping their meaning in contrast to stemming. Lemmatization does not cover only the removal of suffixes but also includes the conversion of the word into its basic form like from "*best*" to "*good*".

The sentence given in stemming as an example is transformed to "*mix material in mechanical mixer except finish coat contain lime may be hand mix*" after lemmatization. The actions "*mix*" and "*mixed*" keep the root verb form whereas "*mixers*" is converted to singular form only that is keeping the machinery definition. Moreover, the words "*mechanical*" and "*material*" are in their usual, interpretable forms.

POS Tagging: Part of speech tagging is the assignment of categorization to the words in a sentence based on their part of speech, also accepted as lexical categories like noun, verb, adjective or adverb etc. (Bird et al., 2009). POS tagging is useful for detecting the sentence's grammatical structure, which helps to clarify the ambiguities in terms of the various meanings of a word.

POS tagging of example sentence is "*Mix (VB) materials (NNS) in (IN) mechanical (JJ) mixers (NNS) except (IN) finish (JJ) coats (NNS) containing (VBG) lime (NN)*

may (MD) be (VB) hand (NN) mixed (VBN).". The tags inside the parenthesis represent the lexical categories; for instance, “VB” describes a verb, “VBG” as a verb in gerund, “IN” as a preposition, or “NN” as a singular noun.

POS tagging is a basic and essential stage in NLP applications like machine translation, NER and information extraction or can be used as an intra-step before lemmatization.

Fundamental preprocessing techniques are described briefly above. Throughout the study, some of them might be eliminated or modified from the generic application of the technique or may be incorporated in other stages. The decision on the application of basic preprocessing techniques is considered during the development of the research. Moreover, custom techniques may be created based on the requirements of the ML models. Specific information on the preprocessing stage of the study are given in Chapter 5.1 Text Preprocessing.

3.2.6 Text Representation

ML algorithms consist of rules or procedures to execute assignments, primarily to recognize the meaning of data and patterns and forecasting values from specified input variables (IBM Data and AI Team, 2023b). The essential question arises in definition: how the algorithms read and comprehend the data if it is in the form of text, in other words, strings in computer science. All the previous stages, starting from the data collection to the end of the preprocessing were working on the text forms. However, ML algorithms require numbers to study, learn, and forecast. So, it is a stringent prerequisite to transform the text data into a mathematical representation form which is called text representation (Vajjala et al., 2020).

Text representation is the route of converting text into a consistent structure to convey the data to ML algorithms in the NLP pipeline. Vector Space Model (VSM) is a way of representing a token (word, phrase, character, document, etc.) in an algebraic vector, which is composed of numbers only. VSM aims to convert text data

in a consistent way so that ML algorithms can process the input and give a reliable output.

This specific representation is also called Text Vectorization. There are different methods to convert text to vectors, from simple statistical-based to DL-based semantics approaches.

Bag of Words (BoW): Bag of Words is a simple statistical base text vectorization method that omits the context and grammar (Otten, 2024). Vectorization is formed by collecting the vocabulary in text data as a bag. The capacity of the bag is limited by the distinct number of tokens in the corpus, namely documents.

The representation of text is in the form of a document term matrix where each row signifies a sentence, and each column is the frequency of the token in the related sentence (Patil et al., 2023). An example of a corpus composed of 3 sentences is shown below:

Table 3.1 Sample Corpus

Sentence-1	add gypsum plaster
Sentence-2	apply plaster by hand
Sentence-3	not to apply gypsum to surface contain frost

The vocabulary set of the above corpus contains 11 tokens which means the size of the columns is also 11.

Table 3.2 Sample Vocabulary Set

w1	w2	w3	w4	w5	w6	w7	w8	w9	w10	w11
add	gypsum	plaster	apply	by	hand	not	to	surface	contain	frost

3 sentences with 11 tokens create a matrix of 3x11 in which each number shows the frequency of each token in the related sentence, as demonstrated below. The sequence of the columns must follow the same index shown above for all rows. The

vocabulary index doesn’t affect the algorithms' outcome so that they may be created differently.

Table 3.3 Sample Document-Term Matrix (BoW)

	w1	w2	w3	w4	w5	w6	w7	w8	w9	w10	w11
S-1	1	1	1	0	0	0	0	0	0	0	0
S-2	0	0	1	1	1	1	0	0	0	0	0
S-3	0	1	0	1	0	0	1	2	1	1	1

The above example simply presents how BoW transforms a corpus into a vector representation ready to input ML algorithms. BoW is simple to create and recognize. The vector sizes of sentences that are fixed with the dimension of vocabulary are compatible with each other.

However, this feature could also result in a high dimensionality problem and a sparse matrix in which the numbers are mostly zero. Considering a corpus containing 1000 sentences and 10000 tokens in a vocabulary set produces a matrix of 1000x10000 in which a sentence comprised of 4 tokens is represented with a vector containing 4 positive integer numbers, the remaining 9996 numbers are all zero.

BoW doesn’t consider the semantics of words. If lemmatization or stemming is not performed previously, the words “apply” and “applied” are considered separately, which unnecessarily increases the vector dimension. Also, the grammatical structure of the sentence does not have any influence on setting up the corpus matrix, which leads to the loss of text semantics.

Another disadvantage of BoW is the inability to handle Out of Vocabulary (OOV) words. Since the vocabulary set covers the words from a certain corpus, any word not present in the corpus do not have a representation in the vector.

Although there are numerous disadvantages of BoW, it is highly utilized in text vectorization of classification problems in NLP (Vajjala et al., 2020) due to its straightforwardness and practicability.

TF-IDF: Term Frequency-Inverse Document Frequency (TF-IDF) is another statistical text vectorization technique that dignifies some of the words in a document relative to others in contrast to BoW, where every word is equally important. The logic behind TF-IDF is that the most frequent words in every document, like stop words, do not contribute significant meaning to the text, so they should be penalized. In contrast, rare words are valuable and should be prioritized.

TF-IDF score of each word is a multiplication of two terms: Term Frequency and Inverse Documents Frequency.

TF is truly normalized term frequency, which is the division of the frequency of a word or token in a document to the total number of words in the documents (Patil et al., 2023).

$$\text{TF} = \frac{\text{frequency of token (t) in document (d)}}{\text{total number of tokens in document (d)}} \quad \text{Eq(1)}$$

IDF is the measurement of the importance of a word by penalizing the frequent words and raising the importance of unique words.

$$\text{IDF} = \log \frac{\text{total number of documents}}{\text{number of documents containing token (t)}} \quad \text{Eq(2)}$$

$$\text{TF-IDF score} = \text{TF x IDF} \quad \text{Eq (3)}$$

TF calculation of the example in Bow:

Table 3.4 Sample TF Calculation

TF	w1	w2	w3	w4	w5	w6	w7	w8	w9	w10	w11
S-1	0.33	0.33	0.33	0	0	0	0	0	0	0	0
S-2	0	0	0.25	0.25	0.25	0.25	0	0	0	0	0
S-3	0	0.125	0	0.125	0	0	0.125	0.25	0.125	0.250	0.125

IDF Calculation:

Table 3.5 Sample IDF Calculation

w1	w2	w3	w4	w5	w6	w7	w8	w9	w10	w11
0.477	0.176	0.176	0.176	0.477	0.477	0.477	0.477	0.477	0.477	0.477

The TF-IDF vectorization:

Table 3.6 Sample Document-Term Matrix (TF-IDF)

TF	w1	w2	w3	w4	w5	w6	w7	w8	w9	w10	w11
S-1	0.16	0.059	0.059	0	0	0	0	0	0	0	0
S-2	0	0	0.044	0.044	0.119	0.119	0	0	0	0	0
S-3	0	0.022	0	0.022	0	0	0.060	0.119	0.060	0.060	0.060

Sentences are considered separate documents in the calculation of TF-IDF vectors above. It is observed that common words between sentences like gypsum, plaster, and apply have the lowest numbers in related sentences.

Although TF-IDF captures the importance of significant words and penalizes the common ones to a certain degree, it is still unable to overcome the loss of semantics and grammar structure, high dimensionality and sparsity problems, and inability to deal with OOV words.

Word2Vec (W2V): Word2Vec is a word embedding method developed by Mikolov et al. (2013) to overcome the drawbacks of BoW and TF-IDF. Unlike the discrete representation of BoW and TF-IDF, word embedding is a distributed representation where each word is represented by a multi-dimensional dense vector. Word embeddings that capture the context, meaning, and semantic connection between the words are created by neural network models training on a large corpus.

Word embeddings are dense vectors that have mostly non-zero numbers in contrast to sparse vectors, therefore they carry more information about the word. Semantically close words like cement, concrete, and aggregate are closer to each

other than door or wood in vector space. Even this feature by itself can improve text classification by clustering similar words or documents in vector space.

Mikolov et al. (2013) presented two models for W2V representation: Continuous Bag-of-Words (CBOW) and Skip-Gram model. CBOW model predicts the target word in a sentence by examining the surrounding words, the words before and after the target word. Whereas the Skip-Gram model forecasts the neighboring words by studying the center word.

CBOW Model: predict "gypsum" in the sentence "not to apply …... to surface contain frost" based on neighbor words

Skip-Gram Model: predict "to apply" and "to surface" in the sentence "not …... gypsum …... contain" based on "gypsum"

W2V method applies a shallow neural network model which includes one linear hidden layer to simplify and speed up the generation of embeddings. The size of this single hidden layer also shapes the dimensionality of the word embeddings. If there are 50 neurons in the hidden layer, the embedding vector is in the dimensionality of 50 likewise.

W2V brought efficiency in training large corpora, semantic relationships between words, and the context of the word. However, the generated word embeddings are still static; in other words, they depend on the trained corpus, and the various meanings of the word based on context are not reflected in embeddings. For example, the word "concrete" has the exact embedding vector whether it is defining a construction material or an adjective as in the meaning of actual. The domain of the corpus influences the representation of the word.

The efficiency of word embedding created by W2V is influenced by the size of the corpus used. The larger the corpus, the higher the representation of the words is generated. Therefore, pre-trained word vectors trained on the Google News dataset containing about 100 billion words are planned to be used for text representation in

the research. Also, custom word embeddings from the research data set are generated to expand and tune the pre-trained model.

BERT: Bidirectional Encoder Representations from Transformers (BERT) is an unsupervised language model developed by Devlin et al. (2019) to generate contextualized word embeddings by capturing the meaning of the word in different sentences. BERT employs a transformer architecture which was introduced by Vaswani et al. (2017).

The transformer model operates self-attention mechanisms that capture all the words in one or more sentences simultaneously rather than sequential processing. BERT evaluates the relationships between the words in combined sentences also up to 512 tokens. The bidirectional feature of BERT lets to explore the left and right contexts of a word at the same time in contrast to one-directional movement. Therefore, BERT creates different word vectors for the word “concrete” in sentences “concrete buildings are cheaper than steel ones” and “he showed concrete evidence to support his colleague”.

BERT is implemented in two steps: pretraining and fine-tuning. Pretraining is the unsupervised training of the BERT model on unlabeled data or corpus to catch the possessed patterns and generate embeddings. Two unsupervised tasks are performed while pretraining the model: masked language model (MLM) and next sentence prediction (NSP). Randomly selected 15% of tokens in each sentence are masked from the input data to force the model to guess the correct vocabulary according to its context in the MLM step. In the NSP step, it is demanded from the model to predict the next sentence from the corpus sentence to extract the sentence relationships.

Pretraining of BERT was carried out on BooksCorpus (800 million words) and Wikipedia English (2500 million words). Two model sizes were developed, such as the base model having a dimensionality of 768 for word embeddings and the large model having 1024.

RoBERTa: Robustly Optimized BERT Pretraining Approach (RoBERta) is a modified version of BERT enhanced by Liu et al. (2019). RoBERTa is developed over BERT's masking approach in dynamic form but eliminating the next sentence prediction strategy which is stated by developers as not having a significant contribution to model performance. RoBERTa was pre-trained on a much larger corpus than BERT with longer batch sizes and higher steps that provide to learn longer sequences of tokens than BERT. Consequently, the performance of RoBERTa was determined to outperform the BERT model by Liu et al. (2019).

The pre-trained model of RoBERTa was trained on BookCorpus, Wikipedia English, Common Crawl News, OpenWebText, and Stories including a subset of Common Crawl data. Two versions were developed as large and base models.

3.2.7 ML Modelling

The development and implementation of rule-based NLP models are not considered in this study to be in conformance with the research objective, which is the development of supervised learning models for the identification and classification of deficiencies observed in specification texts.

The process of supervised learning models for text classification in this research is presented in Figure 3.2. X-train data is a collection of the statements selected for training the models; y-train labels are the labels of the corresponding statements in other words defect groups. The parameters for each ML algorithm are determined by cross validating the train data according to previous research results and recommendations.

Training of the algorithm with the train data and its matching labels under adjusted parameters yield the ML models in this research. Selected statements for test data (X) proceed through each trained model to obtain the ML model's prediction. A comparison of the predicted labels with the test set labels are performed to evaluate

the performance of the trained model. The evaluation methods are detailed in the next section.

It is decided to share distinctively 10%, 20% and 30% of the whole data set as the test data. This also leads to observing the developed model's performance in various train data sizes. Test data is never included in the training of each trial. For instance, in the 20% test, only 80% of the whole data is used for training the algorithm to create the model.

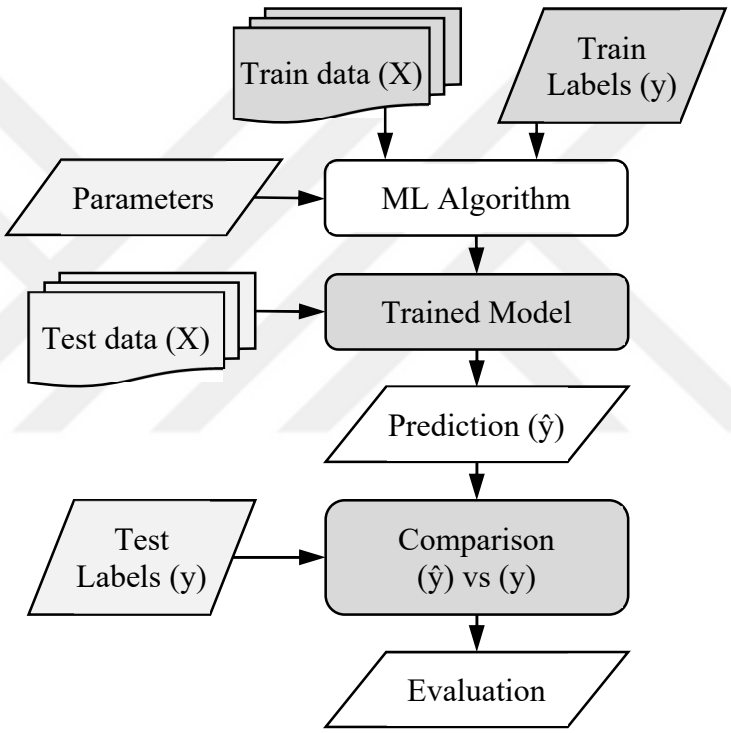


Figure 3.2. Supervised ML Flowchart for Text Classification

The development of supervised learning models for text classification applications involves the functioning of numerous algorithms, each exhibiting unique features. These are sometimes also referred to as classifiers. The classifier employed within the NLP pipeline originates from either traditional shallow machine learning algorithms and neural network algorithms.

Support Vector Machines (SVM), Naïve Bayes (NB), Linear/Logistic Regression (LR), Decision Trees, Random Forests, and k-Nearest Neighbour (kNN) are several common examples of traditional shallow ML algorithms that are used as classifiers in supervised learning.

Recurrent Neural Networks (RNN), Long Short-Term Memory Networks (LSTM), Convolutional Neural Networks (CNN), and Transformers like Generative pre-training, BERT, and RoBERTa are some DL algorithms used as classifiers in text classification problems.

The classifier's performance depends on the dataset's type and size and the classification problem's complexity. The research also questions how various models perform with the same dataset under three train-test splits.

Two criteria are assessed in selecting classifiers: Firstly, there should be algorithms with different complexities and techniques from shallow ML and DL algorithms. Secondly, considerably well-performed models from each category that are observed in the domain from previous research studies in text classification applications have priority (Baek et al., 2021; Wong et al., 2024).

Below algorithms and pre-trained models are considered to deploy in developing the classifier in the NLP pipeline of this research.

Logistic Regression (LR): Logistic regression is a statistical technique that calculates an output value, interpreted as the probability, with the logistic function. LR is considered a binary discriminative classifier where the output values are between 0 and 1.

$$\text{sigmoid function} \quad \hat{y} = \frac{1}{1 + e^{-z}} \qquad \text{Eq}(4)$$

$$z = b + w^{t}x \qquad \text{Eq}(5)$$

where (z) is the linear combination of input values (x), weights (w), and bias term (b).

Logistic regression uses sigmoid function (ŷ) which judges the result of binary classification. When the value (ŷ) is equal to or higher than 0.5, the result is accepted in positive class. If it is less than 0.5, the class is decoded in a negative zone. While training a model, LR applies a loss function called binary cross entropy with regularization to modify the weights to minimize the loss function. Regularization is a way of penalizing the loss function and in sequence the model to prevent overfitting by restricting the weights in the model.

Support Vector Machine (SVM): Support Vector Machine is a supervised learning algorithm that classifies the data by separating the data distribution with a boundary line or hyperplane in multi-dimensional space. The boundary condition is obtained by maximizing the distance between the boundary and the nearest data point from each class which is called also margin, as shown in Figure 3.3 for a typical linear SVM.

The figure is an example of a hard margin which means that the data can be entirely linear separable. In contrary circumstances, when the data distribution is not able to be divided perfectly linearly, a soft-margin approach is introduced to the algorithm that allows some misclassification within the margin. Soft margin targets to minimize a loss function having regularization parameter C. Regularization parameter C controls the loss function's weight where a high value of C brings less regularization, fitting the data as much as possible, and a low value of C leads to high regularization, more generalization, more tolerant to errors.

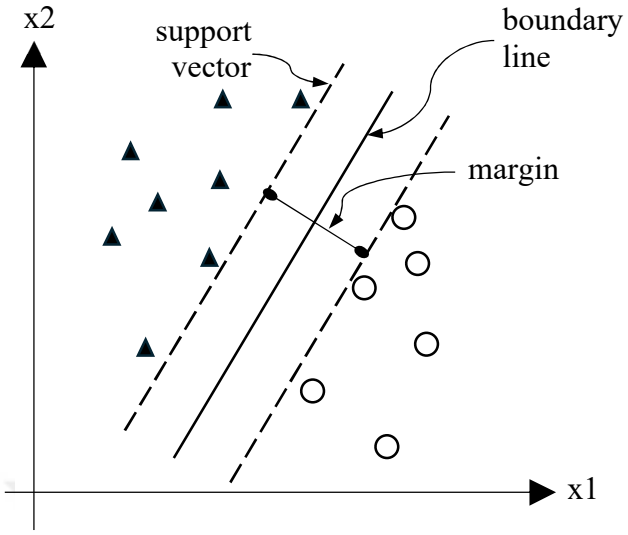


Figure 3.3. Typical Linear SVM Boundary Condition

SVM can train non-linear classification in higher dimensional space as well as linear ones with the help of different kernel functions.

Naïve Bayes (NB): Naïve Bayes is a probabilistic supervised learning algorithm based on Bayes' theorem. The method undertakes a Naïve assumption that given the class label, all features are independent of each other. The objective of the NB classifier is to select the highest probable class according to given data. Since the denominator in the equation is the same for all classes, the equation is transformed into a simpler format.

$$P(y/X) = \frac{P(X/y).P(y)}{P(X)} \longrightarrow P(y/X) = P(X/y).P(y) \quad Eq(6)$$

Where P(y/X) is the probability of class given the features of X, P(X/y) is the conditional probability of X features given the class y, and P(y) is the prior likelihood of class y.

Multinomial NB is an adaptation to estimate categories when the data is discrete like word frequency vectors. The classifier counts the frequency of each feature observed in the class as the conditional probability.

$$P(y/X) = P(y).\prod_{i=1}^{n} P\left(\frac{x_i}{y}\right)^{count(xi)} \quad Eq(7)$$

The conditional probability of features given the class is calculated by smoothing to prevent zero probabilities as follows:

$$P(x_i/y) = \frac{\text{related word frequency in class} + \alpha}{\text{total word frequency in class} + \alpha n} \quad Eq(8)$$

where n represents the size of the vocabulary. $\alpha = 1$ in Laplace smoothing, $\alpha < 1$ for Lidstone smoothing (Pedregosa et al., 2011).

k-Nearest Neighbor (kNN): k-nearest neighbor is an instant-based supervised learning algorithm for classification problems. kNN functions by memorizing the labels in the training set to use the classification of new data sets. The k represents the number of closest neighbors the algorithm uses to make a prediction. The distance between points is calculated mostly by Euclidean distance.

In supervised learning, when a new feature is input, the algorithm retrieves k nearest points to the new data in trained vector space and predicts the class based on majority voting. For instance, assuming k=5, kNN calculates the closest 5 points to new data and checks the class of each neighbor. The predicted class of the new data is the class that has a majority among 5 neighbors.

The parameter k in kNN rules the prediction's performance. When k is low, the algorithm is affected by noises in the train data set, or, conversely, when k is high, there is too much generalization that decreases flexibility.

Convolutional Neural Networks (CNN): Convolutional Neural Networks (CNN) is a kind of DL algorithm that is primarily formed of the convolutional, pooling, and fully connected layers. CNN is highly utilized in image classification and object recognition applications.

The convolutional layer is the first layer of CNN that admits the input data and performs the calculations to a higher extent. In the convolutional layer, a filter

(kernel) which is in smaller size, moves over (stride) the input layer and maps the features to a smaller shape output (A. Zhang et al., 2023). Kernel, which covers the weights, dot products with a part of input data having the same size and strides to the next windows as shown in Figure 3.4.

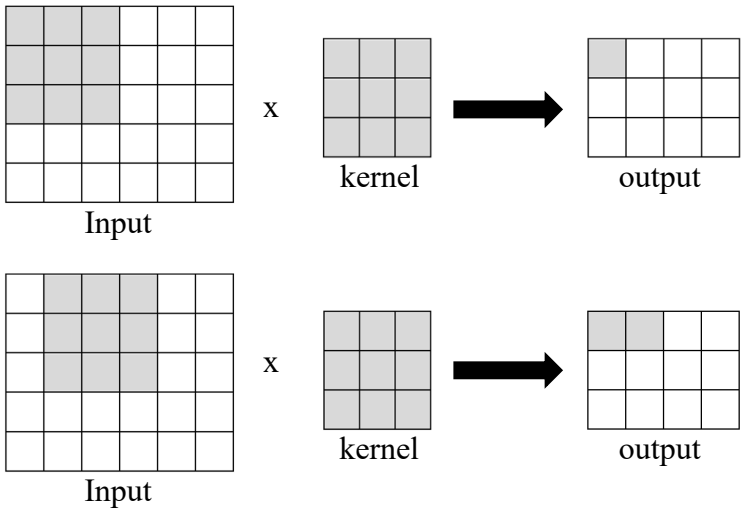


Figure 3.4. Typical Convolutional Layer

Various convolutional layers having different kernel sizes exist in a typical CNN algorithm. The sizes of the outputs of convolutional layers depend on the dimensions of the input and the kernels. So, for the same input, the outputs of the separate convolutional layers have distinct dimensions. The pooling layer in sequence samples down the outputs reduces dimensionality by selecting the max values in the output matrix or averaging the values. Finally, a fully connected layer combines the output from the pooling layer and produces the final output for the target such as a classification problem.

Kim (2014) adapted the CNN algorithm into NLP and demonstrated that a straightforward CNN model achieves substantially accurate results in text classification. In the TextCNN model, a sentence in which each word has multidimensional embeddings passes through one-dimensional convolutional layers with different kernel sizes to create outputs. One fully connected layer is placed at

the end of the model with a softmax function that decides the classification of the sentence. Accordingly, the classification power of the CNN algorithm was applied also in NLP problems.

BERT & RoBERTa: BERT and RoBERTa are the two transformer-type models that have a similar neural network basis. It was stated in the preceding text representation section that both models were developed in two stages: pre-training and fine-tuning. The pre-training stage is the training of both models with an extensive corpus, as described previously. It is a costly effort, time-consuming and requires high computational resources. However fine tuning of the models is a more straightforward step.

The BERT and RoBERTa models are fine-tuned by training the pretrained models with another domain-specific corpus. This task is also considered transfer learning in which the pre-trained model is adapted with a new corpus. When a classification layer is added to the end of the fine-tuned model, the whole model is transformed into a supervised learning classifier as indicated in Figure 3.5. Consequently, the same BERT and RoBERTa models can be used for text classification by adding a layer to the end of the model.

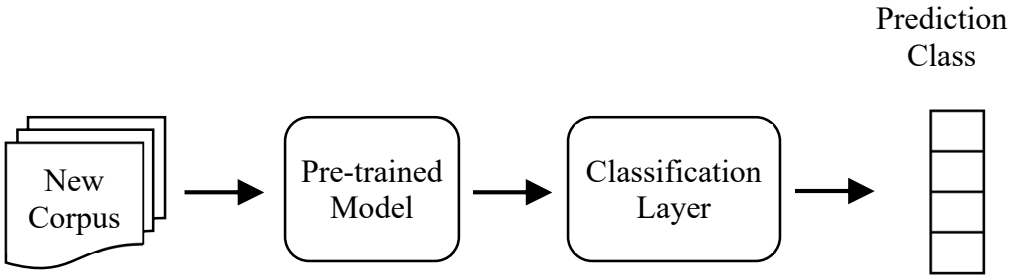


Figure 3.5. Transformation of BERT and RoBERTa to a Classifier

3.2.8 Performance Evaluation

The final step of the NLP pipeline is evaluating the output of each ML model. Two types of performance metrics are used: Confusion Matrix and Evaluation Metrics.

Confusion matrix is a table showing the distribution of True Positives (TP) and False Positives (FP) among the classes for a multi-class model as a sample of 3-class shown in Table 3.7. The total of each column represents the total of predicted outputs, whereas the total of each row is the total of the actual values. The diagonal numbers are considered TP for each class.

For example, the total number of actual values in class 2 is the sum of FP_{12}, $TP_{22,}$ and FP_{32}. Whereas the total number of predicted outputs for class 2 is the sum of FP_{21}, $TP_{22,}$ and $FP_{23.}$ The confusion matrix provides a visual representation of the classification errors for each category.

Table 3.7 Confusion Matrix Sample

		(Predicted Class) 1	2	3
(Actual Class)	1	TP_{11}	FP_{21}	FP_{31}
	2	FP_{12}	TP_{22}	FP_{32}
	3	FP_{13}	FP_{23}	TP_{33}

Evaluation metrics are Accuracy, Precision, Recall, and F1 score.

Accuracy is the measurement of correct predictions among the total predictions. It is the ratio of total TP to total predictions.

$$\text{Accuracy} = \frac{TP_{11} + TP_{22} + TP_{33}}{\text{Total of TPs}} \text{ (for above example)}$$

Accuracy is a common evaluation metric when the frequency values of the classes are considerably uniform. In the opposite case, when one category dominates over others, accuracy misinforms, even resulting in a higher score. The numerator of the above accuracy equation is raised by the governed class.

Precision measures how many of the predictions for a class are true. It is the ratio of TP for a class to the total of the predictions for the same class. Precision is calculated

individually for each category. The overall precision of a model is calculated either by weighted or macro (unweighted) averaging. In weighted average precision calculation, the precision of each class is weighted based on the percentage of each class in the data set, whereas the macro is the arithmetic mean, which does not include any weight.

$$\text{Precision for Class 1} = \frac{\text{TP}_{11}}{\text{TP}_{11} + \text{FP}_{12} + \text{FP}_{13}} \text{ (for above example)}$$

Recall is the share of true predictions to the actual frequency of the same class. It is the ratio of TP for a class to the total of the actual values for the same class. Recall is also calculated separately for each model and averaged to demonstrate the overall performance of the model, such as precision.

$$\text{Recall for Class 1} = \frac{\text{TP}_{11}}{\text{TP}_{11} + \text{FP}_{21} + \text{FP}_{31}} \text{ (for above example)}$$

F1 score is the observation of both precision and recall simultaneously. It is the harmonic mean of precision and recall.

$$\text{F1 Score} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad \text{Eq(9)}$$

Macro F1 score is the arithmetic average of F1 scores of all individual predicted classes.

$$\text{Macro F1 Score} = \frac{\sum \text{f1 scores}}{3} \quad \text{(for above example)}$$

Confusion matrix and evaluation metrics specify how well the developed models can perform text classification, in other words, identify and classify the actual deficiencies in construction specifications. Moreover, these techniques assist in determining the performance variations between combinations of various text representation methods and ML algorithms. Another consequence is comparing the results with other research studies in the domain.

3.3 Research Tools

This section summarizes the main tools utilized throughout the research stages.

Python is the most common programming language used in NLP development and applications due to its extensive and supportive libraries specialized in NLP tasks and data science (Shaikh, 2023). Not only its rich libraries place Python on top of the list, but also its user-friendliness, convenience to learn and adapt, flexibility, and widespread community support have made Python popular in the AI environment. Consequently, it is decided to work with Python 3.11.9 programming language to carry out the stages in the NLP pipeline of this research. Also, some of the statistical calculations in the early stages of research and the data validation are performed with the support of Python coding.

Primary Python libraries intended to make use of in the research are introduced below. Task-specific details of the tools are given in related sections of the thesis.

Pandas: Pandas is an open-source data analysis and handling library developed by McKinney (2010) for Python. It creates and transforms data into a tabular format similar to Excel, which is called Dataframe. Pandas can filter, combine, merge, clean, separate, or transform data or dataframes. It can read and write different file formats including Excel file formats. Pandas is highly used in the research's initial stages, especially in importing and exporting text data to/from the Python environment, combining text data from different specifications, and manipulating the columns.

Spacy: Spacy is a free library created for advanced NLP applications in Python environment (Honnibal et al., 2020). Spacy has trained pipelines for over 25 languages and supports custom pipelines. Some of its main capabilities are sentence and word tokenization, POS tagging, dependency parsing, lemmatization, Named Entity Recognition, and pre-trained word embeddings.

NLTK: Natural Language Toolkit (NLTK) is another NLP-focused library written in Python by Bird et al. (2009). NLTK supports more NLP tools and techniques

mainly designed for research and training purposes than Spacy. Due to their flexibility and higher customization ability, NLTK tools are used mostly in the preprocessing steps of the study's NLP pipeline.

Scikit-learn: Scikit-learn is a widely used powerful machine learning and data science library projected by Pedregosa et al. (2011). It provides huge amounts of data extraction and analysis tools for preprocessing, algorithms and models, data comparison and evaluation metrics, validation models, dimensionality reduction, etc. Common uses of scikit-learn in the NLP pipeline are in text representation, data preparation, cross-validation, supervised learning modeling, and model evaluation stages.

Gensim: Gensim is a Python NLP specialized library developed by Řehuřek & Sojka (2010) to support the application of unsupervised ML algorithms to unstructured text data. It carries out topic modeling, document similarity and extracts word embeddings by Word2Vec, FastText, and Doc2Vec. It also supports running with pre-trained models like Google W2V.

Keras: Keras (Chollet & others, 2015) is an advanced neural network library developed in Python, specializing in building, training, and running deep learning models. Keras is highly used in developing and evaluating neural network models like Recurrent Neural Networks, Convolutional Neural Networks, Long Short-Term Memory, and Transformers. KerasNLP is an adaptation of Keras, which focuses on NLP tasks, including pre-trained models, tokenizers, preprocessing layers, and metrics, facilitating the development of state-of-the-art NLP workflows. KerasNLP provides a more user-friendly interface for developing NLP models on neural networks, especially with pre-trained transformers.

CHAPTER 4

DATA ACQUISITION AND PREPARATION

4.1 Data Collection and Selection

The construction industry is a multidisciplinary environment where various technical knowledge branches meet to create a common outcome. Even the construction of a simple one-story house is still an incorporative product that unites architectural, structural, geotechnical, electrical, and mechanical works mainly. Each of these professions has its unique characteristics and language in addition to its own requirements, standards, calculations, and design philosophies.

It was appraised that the selection of raw research data, in other words, construction specifications would play a critical role in the development of research and should be based on some strategies as below:

- Specifications should be from a single discipline. So, the technical language: terms, descriptions, and referred standards, could be consistent.
- Specification subjects from the selected discipline should be diversified to prevent bias. The more subjects are included in the data, the more it represents the global discipline. However, enough specifications from each title must be involved in the set. Moreover, an upper limit must be set for the quantity of specification titles to avoid skewing the distribution of the data.
- Specifications selected from different projects would expand the integration of various project backgrounds and professionals.

It was decided to collect specifications from architectural works based on the above-listed criteria. Since, structural and civil works have comparatively fewer specification subjects, which would inhibit the variety of data and limit the size of

textual data for the development of supervised ML models. On the other hand, electrical and mechanical works have a higher range of specification subjects; however, the specifications are extremely long, which would practically prevent the diversification of subjects for data labeling.

Upon the completion of miscellaneous assessments, 21 construction specification subjects belonging to architectural works were selected as the data for the identification and classification of defects as shown in Table 4.1. The specifications which were developed and drafted originally in English, were collected from 16 different international construction projects including countries from Europe.

The number of specifications is spread between 4 and 15, with 4 specifications for stucco having the lowest number, whereas 15 specifications for steel doors and frames have the highest number. The distribution of specifications between the projects is presented in Table 4.2. Project-7 has the lowest number of specifications with 4 whereas Project-1 has the highest with 16. The total number of specifications in the whole data set is 175.

Before advancing further process in research, it was decided to estimate the data size and its sufficiency to develop ML models. One sample from each specification subject was selected. As denoted in Table 4.1, the total number of pages of the sample and the number of sentences (statements and titles) in three randomly selected specifications were counted. The average sentence per full page was calculated as 18. The total number of sentences in the data set was estimated as 21,888. It was concluded that the data set could have an adequate number of sentences to construct supervised ML models for classification while comparing other studies in literature.

Table 4.1 Selected Specifications

Spec No	Specification Title	Qty	Pages from Sample	Total Pages
1	Carpentry	6	6	36
2	Bituminous Waterproofing	6	5	30
3	Board Insulation	6	5	30
4	Mineral Fiber Blanket Insulation	9	4	36
5	Metal Roof Panels	7	15	105
6	Flashing and Sheet Metal	8	9	72
7	Firestopping	6	5	30
8	Joint Sealants	5	8	40
9	Steel Doors and Frames	15	6	90
10	Wood Doors	7	3	21
11	Aluminum Windows	9	5	45
12	Door Hardware	11	20	220
13	Glazing	14	6	84
14	Support for Gypsum Board	9	4	36
15	Gypsum Plastering	7	5	35
16	Stucco	4	4	16
17	Gypsum Board	14	8	112
18	Ceramic Tiling	13	7	91
19	Acoustical Ceilings	8	6	48
20	Terrazzo Floor Tile	5	3	15
21	Toilet Accessories	6	4	24
Estimated Total Number of Pages				***1,216***
Estimated Total Number of Sentences				***21,888***

Table 4.2 Specification Distribution Among Projects

	Projects																
Specs	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**Total**
1	x		x	x						x		x			x		***6***
2	x			x	x								x	x	x		***6***
3			x	x						x	x	x		x			***6***
4	x		x		x			x	x	x		x		x	x		***9***
5	x	x		x			x		x	x	x						***7***
6			x				x	x	x		x	x	x	x			***8***
7			x			x		x		x	x	x					***6***
8	x	x					x		x	x							***5***
9	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	***15***
10	x					x		x	x			x	x	x			***7***
11	x	x						x	x		x		x	x	x	x	***9***
12	x		x		x	x		x	x		x	x		x	x	x	***11***
13	x	x	x		x	x		x	x	x	x	x	x	x	x	x	***14***
14			x		x			x	x	x		x		x	x	x	***9***
15	x				x			x			x		x	x	x		***7***
16			x	x					x		x						***4***
17	x	x		x	x	x		x	x	x	x	x	x	x	x	x	***14***
18	x		x	x	x			x	x	x	x	x	x	x	x	x	***13***
19	x							x	x	x	x	x		x		x	***8***
20	x					x	x	x			x						***5***
21	x	x	x		x				x		x						***6***
Total	***16***	***7***	***12***	***8***	***10***	***7***	***4***	***14***	***15***	***12***	***15***	***13***	***9***	***14***	***11***	***8***	***175***

4.2 Specification Texts Cleaning and Initial Preprocessing

The text cleaning step covers the removal of noise in specification text and prepares for an initial preprocessing that includes sentence tokenization and transforming the text into Excel format to review and label the sentences as denoted in Figure 4.1.

The tasks in this stage were accomplished partly manually and partly with the support of Python programming language. Manual efforts were carried out in Word document file format over each specification document individually.

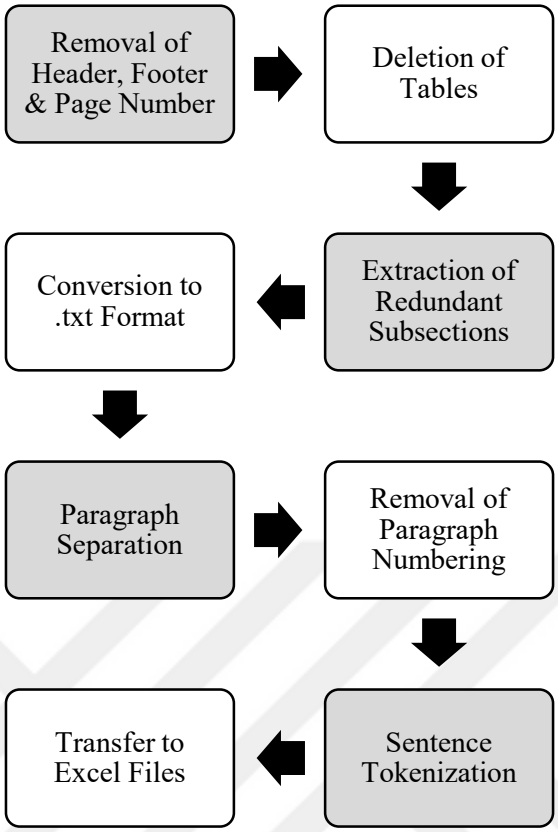


Figure 4.1. Specification Text Cleaning Flowchart

1. **Removal of the header, footer, and page number:** Headers, footers, and page numbers on every page of each specification file were deleted if they existed. In contrast, they do not have any contribution to the performance of ML models in contrast, they have the potential to create solecistic words by positioning inside a word or between words.
2. **Deletion of tables:** Construction specifications are technical documents that may contain remarkable tables. During the text cleaning stage, it was decided to remove completely the complex tables which were determined as the tables having more than two columns. In consideration of the research objective of the study not discovering the technical conformance of the specifications with NLP, the elimination of possible resulting disturbance of the complex tables on the performance of ML models was appraised superior to their existence. Tables with two columns remained while each row would be recognized as a separate sentence.

3. **Extraction of redundant subsections:** References and submittals subsections existing in the specifications were adjudged from documents because these sentences do not embody research-oriented information and would not contribute to the study apart from skewing the data.
4. **Conversion to txt format:** Python programming language is capable of reading files in Word format; however, word format contains additional text metadata like font and paragraph style, font size, page format properties, etc. They are not required in further steps in contrast to the txt file format containing just plain text data. After performing the above tasks, all files in Word format were converted to raw txt file format to be read by Python coding for the following progression.
5. **Paragraph separation:** Although by conversion of documents from Word file format to Txt file format, most of the metadata in files were discarded, there was still some page syntax that needed to be modified, such as empty lines, whitespaces, and line numberings. Each txt file read by Python code was a single Python string object that is required to be checked, cleaned, and separated. Initially, each paragraph within the single string object was split into individual string objects, and empty lines were deleted by using the Python Natural Language Toolkit (NLTK) library (Bird et al., 2009). Whitespaces, which are fundamentally described as spaces and tabs at the beginning and end of each paragraph, were also removed.
6. **Removal of paragraph numbering:** Specification documents have numbering in front of subtitles throughout the text. It has been noticed that those numberings extend up to four digits. The Regex expression module of Python is utilized to remove the numberings at the beginning of titles and paragraphs.
7. **Sentence tokenization:** Until this stage, all text was primarily cleaned from the noises and converted into raw paragraphs that were either titles, single sentences, or a combination of multiple sentences. Since, during the research development, it was required to review each sentence individually and label

it according to defect type, multiple sentence paragraphs had to be additionally separated. Spacy library (Honnibal et al., 2020) was used to perform sentence tokenization. The separated sentences were shaped into tabular form with Pandas Dataframe (McKinney, 2010), where each row was either a title or a single statement sentence.

8. **Transfer to Excel files:** Cleaned and initial preprocessed specifications were reorganized in a list format in the previous step. Reviewing and labelling of the statements would not be practical in a Python environment therefore it was decided to transfer all text in tabular format to Excel files which is a more user-friendly medium for reviewing and labeling the statements as shown in the raw version in Figure 4.2. Since there are 175 specifications from 16 projects under 21 subjects, the organization of specifications was crucial while monitoring the process of labeling the statements. Consequently, the specifications of each project were listed in separate spreadsheets of a project-specific Excel file.

	A	B	C	D	E	F	G	H	I
1	**Gypsum Plastering**								
2	DELIVERY, STORAGE, AND HANDLING.								
3	Deliver manufactured materials in the manufacturers' original unbroken packages or containe								
4	Keep cementitious materials dry and stored off the ground, under cover, and away from swea								
5	ENVIRONMENTAL CONDITIONS.								
6	Gypsum Plaster.								
7	Maintain an ambient temperature of not less than 13 degrees C continuously during plasterin								
8	Maintain this temperature for not less than one week prior to the application of plaster.								
9	Provide regulated ventilation to prevent "sweat outs" or "dry-outs."								
10	When the building is exposed to hot dry winds or day-to-night temperature differentials of 1(								
11	MATERIALS.								
12	Conform to the specifications, standards, and requirements specified herein.								
13	Provide asbestos-free materials.								
14	GYPSUM BASE COAT PLASTER.								
15	Ready mixed type gypsum plaster with perlite shall conform to ASTM C 29, ASTM C 332 standa								
16	GYPSUM FINISH COAT PLASTER.								
17	Shall conform to EN 196-7,EN 459-2, TS 370 standards and manufacturer's recommendations.								

Figure 4.2. Cleaned Raw Specification in Excel File

Following the completion of this stage, Python code was written to find out the distribution of sentences between the specification subjects as presented in Table 4.3.

A total of 21232 sentences which were quite close to the estimated of 21888 (Table 4.1) were calculated in all data sets. The least quantity of sentences was observed in Stucco with 371 whereas the highest one was in Steel Doors and Frames with 2073. The number of sentences shown includes both the specification statements and headings.

Table 4.3 Sentence Distribution in Dataset

Spec No	Specification Title	Spec Qty	# of Sentences
1	Carpentry	6	781
2	Bituminous Waterproofing	6	567
3	Board Insulation	6	647
4	Mineral Fiber Blanket Insulation	9	758
5	Metal Roof Panels	7	1527
6	Flashing and Sheet Metal	8	1940
7	Firestopping	6	453
8	Joint Sealants	5	759
9	Steel Doors and Frames	15	2073
10	Wood Doors	7	533
11	Aluminum Windows	9	987
12	Door Hardware	11	1706
13	Glazing	14	1537
14	Support for Gypsum Board	9	760
15	Gypsum Plastering	7	885
16	Stucco	4	371
17	Gypsum Board	14	1589
18	Ceramic Tiling	13	1527
19	Acoustical Ceilings	8	946
20	Terrazzo Floor Tile	5	392
21	Toilet Accessories	6	494
Total Number of Sentences			***21232***

4.3 Sentence Labeling

The texts of all specifications were converted to Excel tabular forms where each row represents a sentence in the last step of Chapter 4.2. The term sentence refers to a

tokenized statement or a heading of a specification text in this study. Since the research aims to classify the statements based on their defect categories, it is vital to differentiate the headings from the statements throughout the text. Therefore, it was decided to label the headings besides labeling the statements having defects.

However, the focal topic in sentence labeling was to set up the principles of defect categorization, which type of defects had been observed in construction specifications, what their designations were, and how they could be implemented in ML models. Therefore, before the start of labeling the statements, an internal phase was conducted for the resolution of discussions and to construct the labeling framework. The labeling framework settled the type of deficiencies and their basic identification clues.

4.3.1 Labeling Framework

Lam et al. (2004) conducted a survey in 2002 involving participants from clients, consultants, and contractors to examine the utilization of construction specifications in Singapore. The study aimed to determine the practices surrounding construction specifications and to rank observed deficiencies based on feedback from different parties. The study also highlighted significant differences in how various stakeholders perceived specification deficiencies.

Another finding was reported by Institute (2013) based on their previous questionnaires regarding deficiencies observed in construction specifications and the preventive measures proposed to address these issues. The report emphasized that although the listed deficiencies are widely recognized, several factors hinder their resolution. One key reason is the requirement of additional supervision time and examination effort for their identification, which often discourages proactive defect prevention.

The revealed and governed problems related to construction specifications from the literature were combined and summarized in Table 4.4 below. All the identified

problems were gathered from construction stakeholders either with questionnaires or interviews. It was recognized that 17 compiled specification deficiencies can be classified under 10 categories such as Ambiguity and Vagueness, Language Error, Inconsistency, Conflict, Incompleteness, Lack of Specificity, Limited Flexibility, Outdated Information, Applicability, and Out of Scope.

Table 4.4 Defect Categories

Observed Deficiencies	Category
Ambiguous or unclear clauses	Ambiguity/Vagueness
Improper format to locate relevant instruction	Ambiguity/Vagueness
Complex grammar	Language Error
Overcrowded sentences	Language Error
Lack of coordination between specifications, sections and other documents	Inconsistency
Impractical workmanship	Inconsistency
Conflicts in and between different sections	Conflict
Missing tolerances	Incompleteness
Missing instructions for construction	Incompleteness
Missing explicit information required to verify the type, quality, and importance of a task	Lack of Specificity
Improper use or overuse of standards or regulations	Limited Flexibility
Excessive detail in materials or techniques	Limited Flexibility
Impractical tolerances	Limited Flexibility
Superseded standards and regulations	Outdated Information
Inappropriate quality level requested for outcome	Applicability
Unavailability of requested materials and standards locally	Applicability
Redundant or irrelevant requirements to the project	Out of Scope

During the categorization of deficiencies in Table 4.4, it was realized that some of the categories cannot be incorporated into the research properly. For instance, to determine the superseded standards, updated lists of standards published by different authorities must be prepared, including their broad scopes. Moreover, the selected specifications for the research were not prepared in the same year; those lists of

standards in charge had to be provided based on the construction year. Redundant requirements to the project deficiency requires also a review of general terms and conditions or scope of work documents that do not apply to the research. Likewise, determining the availability of materials in the local market of the project location necessitates coordination and historical search of materials in the project location and date, which is not achievable during the development of this study. Consequently, three of the defect categories, which are Outdated Information, Applicability, and Out of Scope, were decided to be removed from the research as it wouldn't be possible to identify deficiencies by reviewing the specifications only.

Furthermore, some of the defect categories were noticed semantically very close to each other so that combination of them would theoretically avoid labeling and validation errors. As a second step, the categories which are conceptually have relations were converged under 4 groups as indicated in Table 4.5.

By doing so, the research objective had evolved to a five-class classification problem: no defect and four defect groups. Any statement of specification would have either no defect or any one of the defect groups.

Table 4.5 Defect Grouping

Defect Group	Category
1	Language Error
	Ambiguity/ Vagueness
2	Conflict
	Inconsistency
3	Lack of Specificity
	Incompleteness
4	Limited Flexibility

The definition of each deficiency type and its most significant way of addressing were detailed in the following sections. The research primarily aimed to identify defects in how the specification text is interpreted by the reader and its usefulness as an information and requirement source from a management viewpoint. The

relevance of the mentioned execution and installation methods to industry practice, as well as whether the described technical properties of materials meet the necessary standards, were not assessed.

4.3.1.1 Defect Group-1

Defect Group-1 consists of deficiencies: ambiguity, vagueness, and language error, which have semantically different definitions despite having convenient relationships.

Ambiguity occurs when a word or sentence has multiple meanings, whereas vagueness appears when a word or sentence is unclear, lack of precision, or does not have any borderline (Solum, 2023). Furthermore, language error covers errors in syntax of grammar, faulty words, and usage of a daily language that doesn't properly fit for technical language of specifications (Kalin et al., 2011).

Ambiguity depends on the reader's perception, and various readers may interpret diverse meanings from the same statement. Readers struggle to comprehend the exact meaning of instructions that must be explicit and binding in construction specifications.

Kalin et al. (2011) express some commonly used words and phrases that are leading to ambiguity in construction specifications. For instance, the use of "any" in situations of aiming obligations as in "any materials rejected shall be removed". In that case "any" implies a choice to the contractor to remove some or all the materials. It doesn't force to remove all the rejected materials. The use of "etc." at the end of a sentence is another example of an ambiguous term. What is included in "et cetera" is unknown. Each reader may interpret it contrarily. Another example is the use of "and/or" together instead of using separately "or" or "both". The instruction "Wall shall be made from concrete brick and/or clay brick" creates ambiguity that one can interpret to construct the wall only from concrete brick, the other from both concrete

and clay brick. The improved practice is to write as “Wall shall be made from concrete brick, clay brick, or a combination of both.

Vagueness on the other hand is equivalent to every individual. It is unclearness and blurred that the anticipated information is not able to be delivered to the readers precisely.

Kalin et al., (2011) highlight the significant use of phrases built with “as” like “as allowed”. If the phrases are used with the following missing objects, the desired requirement will not be achieved due to vagueness. For example, “as allowed”, “as directed” or “as approved” raise the question of “by whom”, similarly, “as appropriate”, “as required” or “as necessary” cause the question of “according to what”. Likewise, “in a workman-like manner” or “to the satisfaction of engineer” are undefined phrases that are relative to the view of individuals.

Standards Committee (2013) advises not to use adverbs such as thoroughly, carefully, clearly and similar ones to avoid vagueness and to provide enforceability in specifications. Instead of phrases like “shall be carefully dried” which do not have any accountability of the drying task, a value in number or percentage should be stated to achieve the requirement.

One of the subjects of language errors is the inaccuracy of grammar syntax, which is briefly the order of words and phrases in a statement. Relocation of a word could undoubtedly change the meaning of the given instruction and lead to ambiguity and vagueness. Like “Sample concrete from each mixer” or “Concrete sample from each mixer”. The first sentence imposes sampling from each mixer, whereas the second one is unclear as if the sentence is divided.

The use of faulty words was also considered a language error. For example, using “insure” instead of “ensure” or confusing the word “course” with “coarse.”

Kalin et al. (2011) state that the language in specifications is more diverse than daily language, not even like the language in business communication, reports, and narratives. There is no place for fancy and imaginative expressions in construction

specifications. In contrast, instructions must be clear, concise, and correct. For instance, “shall” should be used to instruct the works to be completed by the contractor in place of “will” or “to be” which creates vagueness. “will” is used to define the actions of the client or designer. “may” has to be used only in cases proposing alternatives from the requirements. The use of “same” as a pronoun can create vagueness, such as “connected to same” or “mob floor and polish same with wax.” The expressions of “or equal” or “or equivalence” have the potential to cause ambiguity if there isn’t any technical definition of equal or equivalence.

4.3.1.2 Defect Group-2

Issues of conflict and inconsistency in the specifications were consolidated into a different group. Although there are slight differences in definitions, these two terms are adjacent to each other, so the differentiation may not be simple in every case.

Conflict can be basically defined as the contradiction or opposition of mutually exclusive ideas. Whereas inconsistency is a lack of harmony or alignment which may initiate confusion (Vocabulary.com, 2024).

Since the review of other contractual documents like the main contract, general or special conditions, or design drawings are out of the scope of the research, conflict, and inconsistency were determined within each specification text only. They were identified between different statements or within the same statement of each specification.

For instance, if a statement demands a specific material to be in conformance with a certain standard and another statement in the same specification requires the same material in conformance with the manufacturer’s standard product, there is an inconsistency between these two statements that the manufacturer’s standard product may not be in conformance with the required standard. Likewise, on the one hand, letting the material follow the recommendation of the manufacturer, on the other hand, obligating a value for the same material.

Another example of conflict is specifying a limit value for a property of a material in a statement and another value for the same material in a different material. These two statements conflict with each other; one must be provided; it is not possible to provide both at the same time. Similarly, requesting the execution of a task in conformance with a standard and later describing the execution in conflict with the standard is also a conflict.

4.3.1.3 Defect Group-3

Defect Group-3 is related to lack of specificity and incompleteness. Lack of specificity is missing of specific information required to ascertain the type, quality, or necessity of a material, execution, and testing. One example is missing specific reference standards or referring to generic standard names such as “shall conform to local standards,” “applicable European Norms,” or “shall conform to BS, DIN, or equal EN.” All those descriptions have missing principal information that all the parties at the construction site should refer to or follow. It is a question mark whether there is a local or an applicable European Norm for the denoted work or material.

Incompleteness is broadly incomplete instructions or too generic statements like referring only to recommendations, “Windows shall conform to manufacturer’s recommendations.” Who will be responsible for the quality and conformity of the material is unclear, contractor or manufacturer? Since construction specifications are rule books where especially primary materials should be defined precisely. The statements should be either descriptive, defining the properties of materials and installation methods in detail, or referred to specific standards. However, the statement of only “manufacturer’s recommendation” for defining material does not enforce a technical requirement and delivers the responsibility to the manufacturer, not to the contractor.

4.3.1.4 Defect Group-4

As a final category of deficiency, the subject of limited flexibility was figured in Defect Group-4. Although one of the aims of construction specifications is imposing descriptive requirements on contractors and engineers, on the way of providing this, excessive or improper use of standards and regulations could block the flow of tasks in opposition. Especially, when a statement requests several technical standards published by different authorities without "or" conjunction implying a requirement of the material or test of execution should be in conformance to all standards denoted. It is limiting or making it impossible to procure the required material or availability of test of execution. The best practice is to state technical standards in force in the country of construction, or if an option is intended, the standards should be listed by "or".

Listing of multiple standards even published by the same authority for a single product may be unrelated to the material which could create additional trouble in market search or approval. Multiple standards should be carefully referred to.

Unrealistic tolerances were also evaluated under limited flexibility. Tolerance is one of the factors determining the quality of final work. Moreover, they are the margins of contractors that both provide and limit the flexibility of executions. Therefore, it is crucial to determine and modify the tolerances according to each property of the construction project. Construction location, duration, budget, and even type have to be considered while determining. The tolerance required for hospital construction is unfair and applicable for residential building construction.

4.3.2 Labeling Statistics

After developing the defect groups and the principles of labeling, 21,232 sentences in 175 Excel files were reviewed one by one. Two labeling types were performed, one for the headings in the specifications and one for the statements having

deficiencies, as explained in Chapter 4.3.1. The defect group numbers were entered beside the defective statements in the Excel file of each specification.

The sequence of reviewing was selected by completing the same specification subject respectively from each project to provide consistency and to observe the differences between equivalent headings in different projects.

The labeled Excel sheets were transferred to a Python environment to calculate the statistics of labeling. A Python code was compiled to read all files and merge them into Pandas Dataframe formats to prepare the frequency distribution either according to the specification subject or project as presented in Table 4.6 and Table 4.7.

15,569 of 21,232 sentences were distinguished as specification statements, and 5,663 were headings within specifications (Table 4.7). Of the total 1,628 statements, almost 10.5% of them were labeled defective in either group. The highest share between defect groups is from Group-1 with 848 statements (52.1%), Group-3 with 485 statements, Group-2 with 153 statements, and Group-4 with 142 statements are in order, respectively, as shown in Figure 4.3.

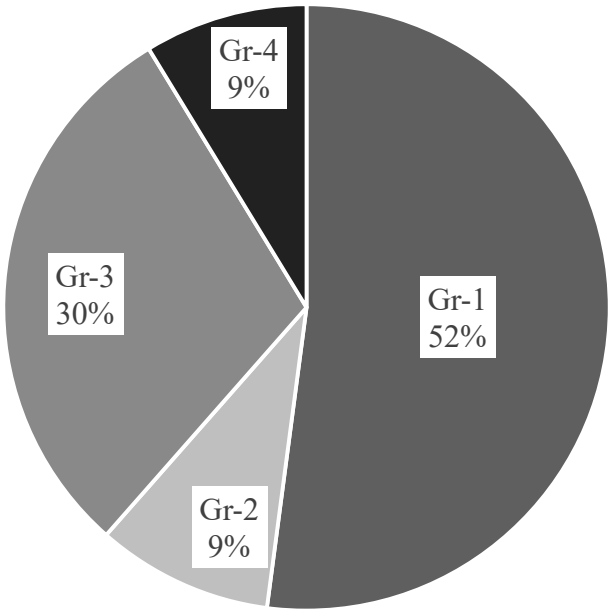


Figure 4.3. Distribution of Labeled Defect Groups in Dataset

The maximum relative defect percentage having 105 defective statements among 589 was observed in carpentry specification at 17.8%, as shown in Table 4.7. Whereas the minimum one in gypsum plastering with 4.3%, 29 of 674 statements were labeled from any one of the defects. The absolute ranking of defective statements among specifications is quite different. The maximum defects were observed in metal roof panel specifications in 194 statements, and minimum defects were observed in toilet accessories in 17 statements.

Correspondingly, Project-9 has the highest number of defects, with 363 statements, as 20.2% of the total, which also positions it first in absolute and relative ranking. Conversely, Project-8 has the lowest relative percentage of defects, with 4.7%.

During labeling, it was recognized that very few statements might be labeled more than one defect group, which would transform the research into a multiclass multilabel problem. However, it was observed that the number of multi-labeled statements would not be sufficient to develop appropriate ML models. Therefore, it was decided to label only in one class where the statement had the subjectively highest possibility of being classified in such cases.

Table 4.6 Defect Frequencies According to Projects

Project	Total Sent.	Statements	Headings	Defect Groups				Total	
				1	2	3	4		
1	1497	1097	400	69	15	15	8	107	9.8%
2	725	529	196	28	9	12	5	54	10.2%
3	1462	1070	392	71	4	4	3	82	7.7%
4	943	669	274	67	2	22	8	99	14.8%
5	1006	719	287	40	20	15	6	81	11.3%
6	646	461	185	23	1	8	0	32	6.9%
7	755	570	185	38	0	2	0	40	7.0%
8	1505	1103	402	49	0	2	1	52	4.7%
9	2435	1795	640	74	0	285	4	363	20.2%
10	2002	1495	507	93	3	7	72	175	11.7%
11	1791	1354	437	80	1	9	4	94	6.9%
12	2229	1641	588	75	6	35	16	132	8.0%
13	840	612	228	28	10	12	5	55	9.0%
14	1752	1263	489	56	72	39	0	167	13.2%
15	1086	772	314	41	9	13	9	72	9.3%
16	558	419	139	16	1	5	1	23	5.5%
Totals	***21,232***	***15,569***	***5,663***	***848***	***153***	***485***	***142***	***1,628***	***10.5%***

Table 4.7 Defect Frequencies According to Specification Titles

Spec No	Specification Title	Total Sent.	Statements	Headings	Defect Groups				Total	
					1	2	3	4		
1	Carpentry	781	589	192	88	2	13	2	105	17.8%
2	Bituminous Waterproofing	567	427	140	57	8	5	3	73	17.1%
3	Board Insulation	647	437	210	18	22	12	14	66	15.1%
4	Mineral Fiber Blanket Insulation	758	482	276	29	6	37	7	79	16.4%
5	Metal Roof Panels	1527	1156	371	137	1	32	24	194	16.8%
6	Flashing and Sheet Metal	1940	1492	448	77	11	16	0	104	7.0%
7	Firestopping	453	345	108	23	0	5	2	30	8.7%
8	Joint Sealants	759	572	187	17	1	23	4	45	7.9%
9	Steel Doors and Frames	2073	1574	499	93	0	27	19	139	8.8%
10	Wood Doors	533	373	160	9	0	10	3	22	5.9%
11	Aluminium Windows	987	700	287	26	1	41	4	72	10.3%
12	Door Hardware	1706	1275	431	49	19	25	6	99	7.8%
13	Glazing	1537	1120	417	38	0	38	3	79	7.1%
14	Supports for Gypsum Board	760	533	227	33	6	18	17	74	13.9%
15	Gypsum Plastering	885	674	211	20	4	0	5	29	4.3%
16	Stucco	371	269	102	11	0	20	10	41	15.2%
17	Gypsum Board	1589	1047	542	30	42	48	10	130	12.4%
18	Ceramic Tiling	1527	1127	400	25	9	85	1	120	10.6%
19	Acoustical Ceilings	946	711	235	34	18	24	8	84	11.8%
20	Terrazzo Floor Tile	392	283	109	22	0	4	0	26	9.2%
21	Toilet Accessories	494	383	111	12	3	2	0	17	4.4%
Totals		***21,232***	***15,569***	***5,663***	***848***	***153***	***485***	***142***	***1,628***	***10,5%***

4.4 Labeling Validation

The whole validation stage encompasses six intra-steps, as demonstrated in Figure 4.4 starting from data set preparation for validation test and ending with validation assessment. The procedures that succeeded in the main steps are detailed in the following sections.

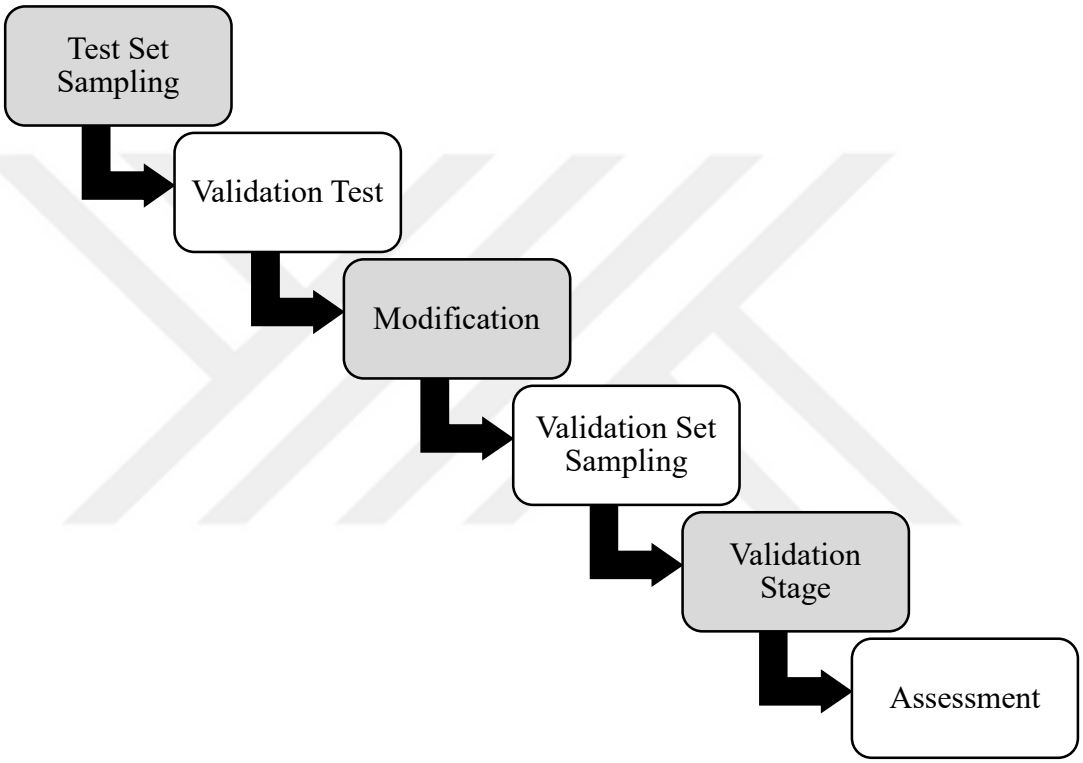


Figure 4.4. Validation Flowchart

4.4.1 Validation Test

A validation test was conducted with an experienced civil engineer before validation to reveal the missing and unclear points. The validation test set comprised research validation notes, a validation handbook, and a specification set. The validation notes introduced the participant to the research summary and described how the validation stage would be carried out. The validation handbook defined the defect categories

and groups so that participants would be consistent while labeling with the researcher. The specification set contained three different specifications, with a total statement of 536. All labels were removed before delivery so the participant would have statements on rows with 4 columns representing each defect group. It was also requested from the participant to measure the durations while reviewing each specification.

An assessment discussion was performed with the participant after the participant's labeling task. The participant noticed that reviewing each statement after a while led to a loss of focus, which was also confirmed by comparing the durations of labeling between specifications. Labeling duration of the first specification, which has 72 statements, was almost the same as the last one, having 311 statements, although the number of statements was 4 times higher.

Another question was raised while labeling by the participant. It was expected that the participant evaluated the specifications from all points of view of the stakeholders (owner, designer, consultant, contractor) in the construction industry, not only as a contractor or an owner. Some defects might be advantageous for owners, while some might be for contractors. Therefore, the labeling should be neutral, with no bias to any stakeholder.

4.4.2 Validation Stage

Based on the validation test step conclusion, it was decided to prepare the validation specifications by marking the statements having defects in contrast to the validation test set having no labels to decrease the labeling duration and prevent motivation loss of the participants. However, to conform with the research aim, principally the determination of the defects and their groups, it was chosen to also mark statements having no defects in an amount of 20% of total defects in each specification. For instance, if a specification has a total of 30 labeled statements, 6 more dummy markings are added as Group-0, which modifies the validation sets to a five-class

classification problem among marked statements only. Hence, the validation would be not only classification but also the identification of the defects.

The validation data set was planned to cover a minimum of 10% of the total defects (1628) observed in the data set. 6 specifications having a total of 173 defects were chosen from the data set as indicated in Table 4.8. While selecting the specifications, the defect group distribution shown in Figure 4.3 was also considered to represent the share globally. Those labeled statements with previously removed labels were marked with color in Excel files in addition to a total of 34 dummy statements. The participants were challenged with a total of 207 marked statements in 6 specifications without knowing whether each statement had a defect or not, and if so, its defect group.

Moreover, research notes and validation notes were also improved based on the consequence of the validation test stage. The validation stage description was modified accordingly, and some new explanations were added.

Table 4.8 Validation Set

Spec	**Statements**	**Defect Groups**				**Total Labels**	**Group-0 (No Defect)**	**Marked Statements**
		1	**2**	**3**	**4**			
1	140	5	0	0	19	24	5	29
2	149	26	0	1	0	27	5	32
3	88	19	0	1	0	20	4	24
4	79	4	0	23	0	27	5	32
5	72	3	13	3	0	19	4	23
6	232	26	0	29	1	56	11	67
Totals	***760***	***83***	***13***	***57***	***20***	***173***	***34***	***207***

The validation was performed with 3 participants having industry experiences between 10-28 years, as shown in Table 4.9. The participants independently reviewed and labeled the same validation set.

Table 4.9 Validation Stage Participants

Participant	Occupation	Experience
1	Senior Architect	23 years
2	Senior Civil Engineer	10 years
3	QA Manager	28 years

4.4.3 Validation Assessment

4.4.3.1 Percentage Agreement

Percentage agreement is the simplest statistical tool to estimate the inter-rated agreement between two annotators. It is the percentage of matched defects between the researcher and each participant to total defects. Table 4.10 summarizes the percentage agreement results. The lowest agreement of 75% was observed with Participant 1, and the highest with 84% with Participant 2.

Percentage agreement estimates the consensus independently between the researcher and each participant. Therefore, it doesn't give enough information on whether the differences between the participants are focusing on similar defects or not.

Although there isn't a specific acceptable level of percentage (Graham et al., 2014) as the acceptable level changes based on the research domain, such as sports, education or medicine, Stemler, (2004) recommends a minimum 70% or more as a typical guideline for evaluation of the level of interrater agreement.

Table 4.10 Percentage Agreement Results

Spec	Total Colored Statements	Participant-1		Participant-2		Participant-3	
		Match	Diff.	Match	Diff.	Match	Diff.
1	29	19	10	28	1	24	5
2	32	22	10	20	12	23	9
3	24	18	6	19	5	16	8
4	32	23	9	29	3	27	5
5	23	21	2	21	2	18	5
6	67	53	14	56	11	53	14
Total	**207**	**156**	51	**173**	34	**161**	46
Percentage Agreements		**75%**	**25%**	**84%**	**16%**	**78%**	**22%**

4.4.3.2 Cohen's Kappa

Cohen's kappa is an estimation of the degree of inter-rated agreement between two raters by also considering the possibility of occurrence by chance (Stemler, 2004). Like percentage agreement, 3 kappa values were calculated for each researcher-participant pairing. A minimum of 0.66 and a maximum of 0.78 kappa values were calculated with Python programming language, as demonstrated in Table 4.11.

Table 4.11 Cohen's Kappa Results

Participant	Cohen's Kappa
1	0.66
2	0.78
3	0.69

Kappa values range between -1 and 1 (Cohen, 1960) which -1 represents perfect disagreement and 1 indicates a complete agreement between raters. The value zero confirms that the consensus between the annotators is by statistical chance only.

Landis & Koch, (1977) recommend that kappa values between 0.61 and 0.80 can be interpreted in substantial agreement.

Table 4.12 Interpretation of Kappa Value (Landis & Koch, 1977)

Kappa	Strength of Agreement
<0.00	Poor
0.00-0.20	Slight
0.21-0.40	Fair
0.41-0.60	Moderate
0.61-0.80	Substantial
0.81-1.00	Almost Perfect

4.4.3.3 Fleiss' Kappa

Fleiss Kappa is another measurement of inter-rated agreement similar to Cohen's kappa but capable of evaluating the reliability between more than two raters (Fleiss, 1971). In contrast to Cohen's kappa, which is among two raters only, Fleiss' kappa considers all three participants and the researcher at the same instance in evaluation. Fleiss' kappa score of 0.633, which is also in the range of substantial agreement (Table 4.12), was calculated between 4 raters (one researcher and 3 participants) for 5 classifications among 207 statements with Python coding.

4.4.3.4 Majority Voting

Majority voting estimation is like percentage agreement but instead of comparing the labels between the researcher and each participant, the comparison was completed between the researcher and participants' majority labels. Therefore, the most labeled defect group from participants' responses for each statement was selected as a major and agreed return as indicated in Table 4.13. In case each participant had labeled a different group, namely no majority choice, if there was one match between the researcher's label and any of the participant's responses, it was selected as the majority vote else not.

193 of 207 majority labels were matched with the researcher's labels confirming an overall agreement level of 93%. The attitude of the majority voting of the participants

in validation as a group was considered more reasoning than percentage agreement which doesn't examine the inter-relations of participants' choices in the validation process.

Table 4.13 Majority Voting Results

Statement	Defect Labels					Match?
	Researcher	Participants				
		1	2	3	Major Label	
1	4	4	4	2	4	Yes
2	1	1	1	1	1	Yes
3	0	0	0	0	0	Yes
4	4	4	4	4	4	Yes
….						
47	1	1	1	3	1	Yes
48	1	3	0	2	0	No
49	1	1	1	1	1	Yes
….						
207	1	1	0	3	1	Yes

4.4.4 Validation Outcomes

Four different inter-rated agreement statistical calculations were performed for the validation assessment of the study: Percentage agreement, Cohen's kappa, Fleiss's Kappa, and Majority voting. The average percentage agreement was calculated as 79%, the average Cohen's kappa as 0.71, Fleiss' kappa as 0.633, and Majority Voting as 93%.

The researcher, having 25 years of experience in both design and construction phases of the industry, reviewed 15,569 statements manually, identified and categorized the deficiencies according to the developed framework in chapter 4.3.1.

Among 4 statistical measures, none of them argued a poor agreement interpretation between the researcher and participants. In contrast, it was demonstrated that there is a conformity with the defect labeling of researcher and the participants above the

thresholds even though the complexity of a five-class classification problem and the requirement of subject expertise. It can be stated that the labeled dataset is free from subjectivity, does not include bias and is validated with three independent experts.

CHAPTER 5

DEVELOPMENT AND IMPLEMENTATION OF ML MODELS

5.1 Text Preprocessing

Selected specifications for research data were cleaned up and passed through an initial preprocessing stage as described in Chapter 4.2. Thereafter, the labeling of the data set was completed in Excel medium as shown in Figure 5.1. Headings in specification documents were tagged to differentiate from the statements, and the statements having any deficiency were labeled according to their defect groups defined in Chapter 4.3.1 Labeling .

Sentence tokenization was completed in the final step of the initial preprocessing to facilitate the labeling of each sentence afterward; therefore, another similar task was not required to be done in the preprocessing stage again.

	A	B	C	D	E	F	G
1		**Bituminous Waterproofing**	**CLASS**	**1**	**2**	**3**	**4**
28	27	INSPECTION OF SURFACES.	T				
29	28	Before starting the work, inspect all surfaces to be waterproofed to determine if in satisfactory condition.		1			
30	29	Check the location and setting of all embedded items.					
31	30	Place backing and blocking and perimeter framing for recessed items as required by the various trades on the project.		1			
32	31	Complete conduit, piping, and other required rough-in.					
33	32	Notify the Contracting Officer of serious defects or conditions that will prevent satisfactory application.		1			
34	33	Start application after such defects and conditions have been corrected.		1			
35	34	PREPARATION OF SURFACES.	T				
36	35	Surfaces to be treated shall be clean and dry, smooth and free from deleterious and excess materials and projections.					
37	36	Cut off or grind smooth high spots.					
38	37	Give surfaces to receive asphalt membrane waterproofing a priming coat of asphalt primer.					
		Apply priming coat at a rate not less than 4 liters per 10 square					

Figure 5.1. Sentence Labeling in Excel Medium

The sequence and selection of preprocessing techniques are dependent on the type of NLP application, type of data, and aim of the modeling (Vajjala et al., 2020). Although the position of a preprocessing technique could be different in each NLP pipeline, some of the methods have prerequisites for better performance. For instance, POS tagging is influenced by the structure of the sentence, stopword removal or lowercasing completed previously can change the grammar and could result in unexpected word tagging.

5.1.1 Text Preprocessing Steps

The following methods were successively followed in each step of the preprocessing stage of this research, as shown in Figure 5.2. Some of the techniques mentioned in Chapter 3.2.5 were decided to be excluded due to the data type and research objective.

Some of the words used as conjunctions, like “as, in, and/or, either” have significant effects on ambiguity and vagueness of statements which are categorized under Group-1 defects. Additionally, the use of inappropriate modal verbs for technical language, like “will” instead of “shall” leads to language error categorization. Although these words can be accounted for in the definition of stopword, removing them from the statements could affect the accuracy of models. Therefore, it was decided to skip the stop word removal technique in the preprocessing stage.

Construction specifications are utterly technical documents. Hence, numbers are not included only in front of the headings but also in the statements to specify an accountable requirement for a material property, a tolerance, or a level of execution. Even the standards and regulations are separated and defined by digit coding. So, numbers, except in headings, play a crucial and distinctive role in the meaning of a statement that a classification model looks for. Consequently, number removal was also disregarded in the preprocessing stage of the study.

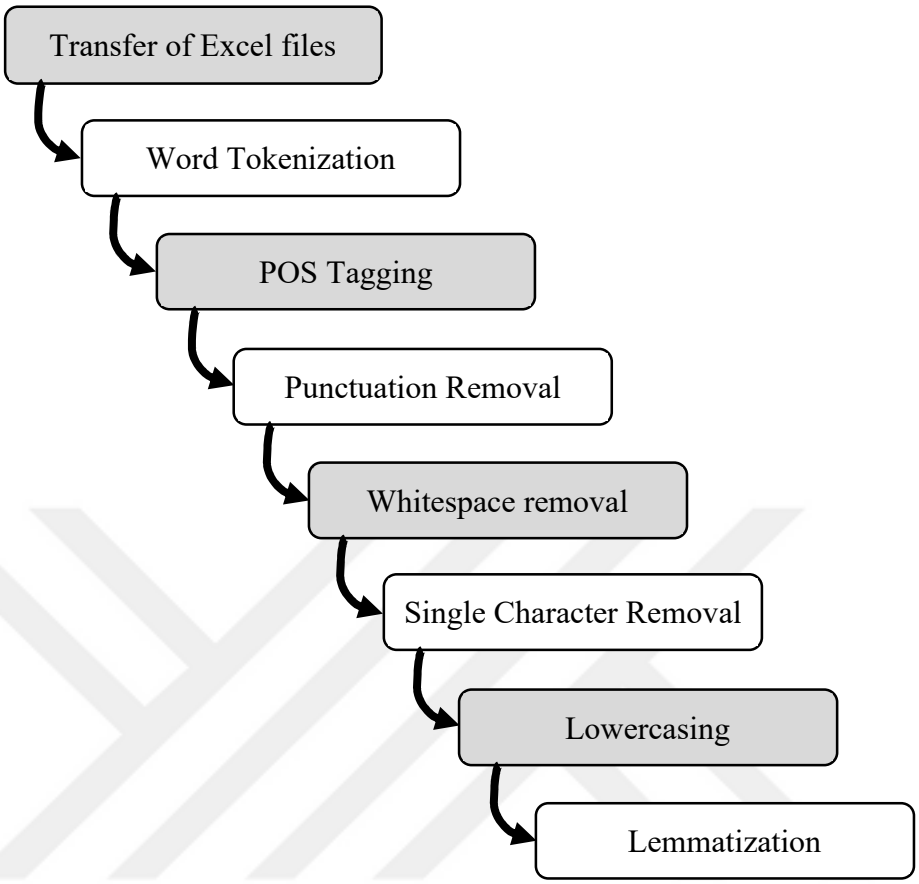


Figure 5.2. Preprocessing Steps

1. **Transfer of Excel files:** All labeled text in worksheets of project-specific Excel files were transferred to the Python coding environment, were combined in a single Pandas dataframe and were reorganized accordingly to distinguish the related project and specification subject.
2. **Word tokenization:** All sentences, including headings, were segmented into word tokens by using the word tokenizer class of the NLTK library.
3. **POS tagging:** Lexical categories of words were determined by the NLTK POS tagger class. POS tagging was performed in the early steps of preprocessing both to improve the efficiency of the tagger before disturbing the structure of a sentence and to increase the accuracy of the lemmatizer by also supplying the POS tags as inputs.

4. **Punctuation removal:** Tokens composed of only any of the punctuation marks like "." at the end of the sentences or "," inside the sentences, and the punctuation marks like "-" or "/" inside words were removed.
5. **Whitespace removal:** Since whitespaces, which are single or multiple spaces or tabs at both the start and end of the sentences, were removed in the initial preprocessing stage, spaces inside the tokens that might be generated in the word tokenization stage were removed in this step.
6. **Single character removal:** Tokens containing a single character, either alphabetic or numeric, were deleted to reduce the dimensionality and increase the performance of models combined with basic text vectorization methods.
7. **Lowercasing:** All filtered tokens were converted to their lowercase format.
8. **Lemmatization:** Tokens were transformed into their lemma forms by utilizing WordNetLemmatizer class of NLTK by inputting the POS tagging of each token. It was observed during the trial of the preprocessing stage that POS tagging of words such as noun, verb, adjective or adverb significantly increases the outcome of the lemmatization step.

5.1.2 Outcome of Text Preprocessing

Table 5.1 presents a selection of statements sampled from the dataset, showcasing both their original and preprocessed forms.

Table 5.1 Sample Preprocessed Statements

Form	Statement
Original	Bolts, nuts and washers shall be zinc-coated if exposed to weather.
Preprocessed	bolt nut and washer shall be zinccoated if expose to weather
Original	Waterproofing for foundation walls shall consist of a 1-ply hot-applied asphalt membrane system.
Preprocessed	waterproof for foundation wall shall consist of 1ply hotapplied asphalt membrane system
Original	Modify hardware as necessary to provide features indicated or specified.
Preprocessed	modify hardware as necessary to provide feature indicate or specify
Original	Wires shall conform to ASTM A641 or equal BS, DIN or EN standards, Class 1 or ASTM A580 or equal BS, DIN or EN standards, condition annealed stainless steel.
Preprocessed	wire shall conform to astm a641 or equal ts bs din or en standard class or astm a580 or equal ts bs din or en standard condition anneal stainless steel

It was decided that the preprocessing stage was completed aligning with the objectives and research target.

It is crucial to observe the numerical outcomes of preprocessing as it provides valuable information for evaluating the selection and tuning of the parameters for both text representation methods and algorithms.

A total of 252,588 tokens (including words, numbers, and alphanumeric) were obtained from 21,232 sentences after preprocessing. The vocabulary set of all specification documents comprised 5,350 distinct tokens, of which 5,271 were

derived from statements and 79 exclusively from headings. The maximum number of tokens in a sentence was 90, the minimum was 1, and the mean was 15 tokens.

Figure 5.3 illustrates the frequency distribution of tokens per statement. The top five most frequent token counts in a sentence are 10 (934 occurrences), 11 (875 occurrences), 9 (866 occurrences), 8 (860 occurrences), and 12 (817 occurrences), respectively. It was observed that longer sentences were rare in the dataset, which could reduce the dimensionality of text representation and potentially enhance the performance of the algorithms.

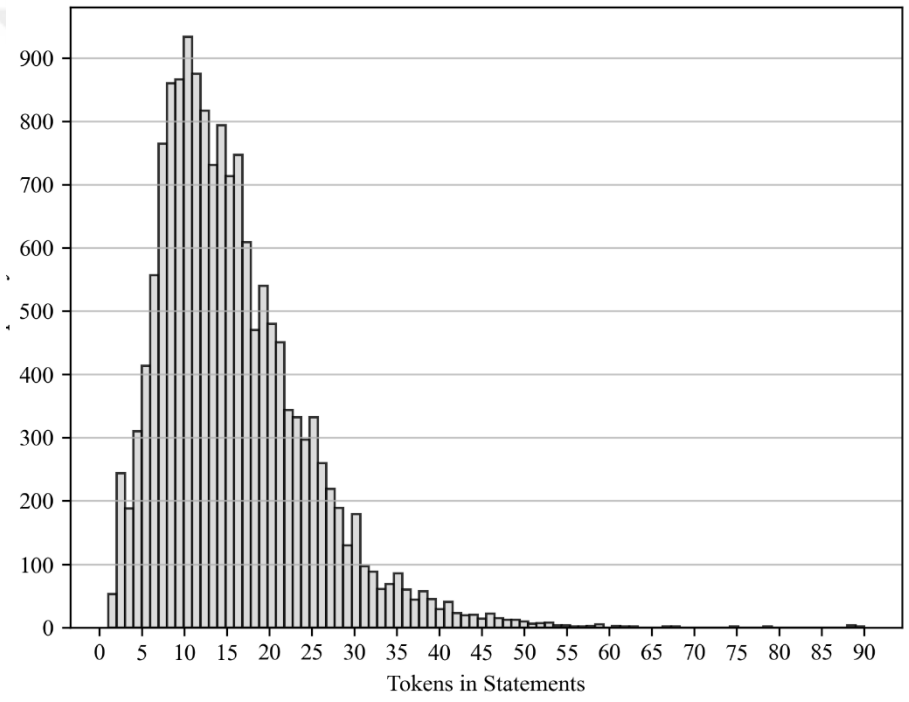


Figure 5.3. Histogram of Token Distribution

5.2 Feature Extraction from Specifications

Upon completing preprocessing, the textual data were prepared for feature extraction to enable numerical representation suitable for machine learning algorithms. 8 different text representations as outlined in Table 5.2, were generated using 6 different techniques described in Chapter 3.2.6 Text Representation.

Table 5.2 Text Representation Approaches in Research

	Text Representation Technique
1	BoW-Unigram
2	BoW-Mixed n-gram
3	TF-IDF-Unigram
4	TF-IDF- Mixed n-gram
5	Pre-trained Google W2V
6	Pre-trained Google W2V+Expanded with custom
7	Pre-trained BERT
8	Pre-trained ROBERTA

BoW: n-gram is a term describing the accounted number of tokens or set of tokens when generating the vectors. Unigram incorporates only one single token from the vocabulary set, while the Bigram model combines two consecutive tokens in a sentence and treats this combination as a single entity in creation of a vector matrix.

Two different text representations were produced using the CountVectorizer class from scikit-learn: one with unigram and another with a combination of unigram and bigram.

Since the data set contains 15,569 statements and 5,350 tokens obtained after the preprocessing stage, the unigram BoW representative consistently produced a matrix with a size of 15,569 x 5,350. Scikit-learn constructs bigrams on a sentence-by-sentence basis rather than from the entire vocabulary set. As a result, the bigram model generated a column size of 33,491, leading to a matrix of size 15,569 x 38,841 when combining unigram and bigram representations.

TF-IDF: TF-IDF vector representations of specifications were performed using scikit-learn TfidfVectorizer class for both unigram and mixed n-gram models as well. Similar matrix sizes were produced: 15,569 x 5,350 for the unigram model and 15,569 x 38,841 for the mixed n-gram model, respectively.

Pre-trained Google W2V: Google trained the Word2Vec (W2V) algorithm on the Google News dataset, which contains approximately 100 billion words, to create a

pre-trained word embedding model. This model comprises 3 million token vectors, each with a dimensionality of 300.

This pre-trained word embedding model was loaded using the Gensim library in Python. Each token's word embedding is represented as a 300-dimensional dense vector. Since the frequency of tokens in sentences ranges from 1 to 90, directly concatenating token embeddings for sentences of varying lengths would result in inconsistent dimensional representations, which are incompatible with the requirements of ML algorithms that need uniform input dimensions.

To address this, mean vectors were calculated using Gensim to represent each statement uniformly. This resulted in an output matrix with dimensions of 15,569 x 300 for the entire dataset, which was smaller than even the BoW representation and consisted of dense vectors.

Pre-trained Google W2V Expansion: The pre-trained Google Word2Vec (W2V) model, trained on the Google News dataset, performs well with common words used in daily life. However, the static nature of the word embeddings created by W2V lacks domain-specific representations. To address this limitation, custom word embeddings were generated from the research dataset to expand the pre-trained model.

The skip-gram architecture of Gensim's Word2Vec class was used to train on the statements from the specifications, producing 300-dimensional custom word vectors. In total, 5,271 token vectors were generated. A comparison with the pre-trained model revealed that 1,701 tokens from the custom set were missing in the pre-trained model. These unique tokens were incorporated into the pre-trained embeddings to create a hybrid model, combining both general-purpose and domain-specific word embeddings.

Pre-trained BERT: The lowercase base version of the pre-trained BERT model was loaded using the BERT class from the KerasNLP library. The model consists of 12 encoding layers, each producing 768-dimensional vectors. It has a vocabulary size

of 30,000 words and a sequential sentence limit of 512 tokens. Given that the longest sentence in the research dataset contains 90 tokens, the sequence length of the model was limited to 128 tokens to optimize runtime efficiency.

Pre-trained RoBERTa: Similar to BERT, the base version of RoBERTa was loaded using the KerasNLP library. Unlike BERT, the pre-trained RoBERTa model retains the case of words during training. RoBERTa also has a larger vocabulary, consisting of 50,000 tokens, compared to BERT's 30,000 tokens. The model comprises 12 transformer layers, with an embedding vector size of 768. To optimize runtime, the sequence length of tokens in the RoBERTa model was also limited to 128.

5.3 Tuning and Implementation of ML Models

The vectorization of specification texts into mathematical representations enabled the development of ML models. In this context, tuning refers to the calibration of models for implementation on training and testing datasets. This calibration involved determining the optimal model parameters to ensure efficient performance. As stated in chapter 3.2.7 ML Modelling, each ML algorithm has unique parameters that must be adjusted to transform the algorithm into a functional model. Since these parameters vary based on the type and size of the dataset, their determination required partial implementation of the algorithm on the research dataset, a process carried out using a cross-validation approach.

K-fold cross-validation is splitting the dataset into “k” equal parts, training the algorithm on k-1 sets, and validating the model with the remaining set. This process is repeated k times by striding the validation set in each step. Finally, the average of the observed metrics is calculated for a set of k times. In each set, the parameter(s) of the algorithm is changed. At the end of the k-fold cross-validation process, a correlation is established between the average metric and the parameter(s), enabling the selection of the optimal parameter(s) for the given dataset.

Therefore, prior to the implementation of each model, fine-tuning of the parameters was achieved with a k-fold cross-validation approach for shallow ML algorithms such as SVM, NB, LR, and kNN. For TextCNN, parameter tuning was carried out following the recommendations of Y. Zhang & Wallace (2015).

Below are the common assumptions and interventions executed in cross-validations of shallow ML algorithms:

- Cross validation set was separated by train-test split class of scikit-learn with shuffling random state is equal to 42. This option provided that the statements with the related labels were shuffled in the same sequence every time the code was run. Therefore, the cross-validation set of distinct algorithms was also consistent.
- 80% of the dataset was used in cross-validation.
- K is 4 folds. So, three folds were used for training and one-fold was used for testing in each step. Grid Search cross validation was implemented for SVM-RBF algorithm only due to determining the two parameters.
- Macro F1 score was selected as the evaluation metric for tuning the parameters. The average of the macro F1 score was calculated for each parameter value.
- Text representation was unigram BoW.

Upon completing cross-validation, the developed ML models, combined with various text representation methods, were implemented across three different train-test splits. All model-text representation combinations were applied to datasets split into 90% train-10% test, 80% train-20% test, and 70% train-30% test, respectively.

The split of train-test sets was performed with scikit-learn, similar to cross-validation that each model encountered with identical shuffled train-test sets.

Initially, the tuning of the model and, subsequently, various implementations are categorically given below.

5.3.1 Naïve Bayes Models

Scikit-learn library NB classifier employs an alpha parameter for smoothing the model not to challenge with zero probabilities as described in the section 3.2.7. Therefore, the model's fine-tuning requires finding the best probable alpha value that maximizes the macro F1 score.

An alpha value set between 0.00001 and 10 was iterated over k-fold cross-validation. Maximum macro F1 score was obtained in 3 values of alpha as seen in Table 5.3. The lowest value of 0.0001, which has the least significant effect on smoothing, was selected for the alpha value.

Table 5.3 Macro F1 Scores based on Alpha Parameters of NB

Alpha	Macro F1
0.00001	0.62
0.0001	0.63
0.001	0.63
0.01	0.62
0.1	0.60
0.5	0.62
1	0.63
5	0.43
10	0.30

Table 5.4 lists the 12 NB model combinations that were developed with BoW and TF-IDF text representations and their implementations on 10%, 20%, and 30% test ratios of the dataset separately. The word vectors with BoW and TF-IDF were generated both with unigram and mixed n-gram approaches. Each model was tested in conformance with the train ratio, as explained in the previous section. For instance, when 70% of the data was used for the training set, the remaining 30% of the data was tested with the model.

Table 5.4 Implemented NB Models

Model Number	Model	Text Representation	n-gram	Test Set Ratio
1	NB	BoW	Unigram	10
2				20
3				30
4			Mixed	10
5				20
6				30
7		TF-IDF	Unigram	10
8				20
9				30
10			Mixed	10
11				20
12				30

5.3.2 K-Nearest Neighbor Models

The kNN classifier from the scikit-learn library was employed to develop the kNN models. The key parameter to be determined is the number of neighbors, known as the k-value. 10 values of k were iterated through a k-fold cross-validation process to observe the highest macro F1 score as shown in Table 5.5.

Table 5.5 Macro F1 Scores based on k-values of kNN

k-value	Macro F1
1	0.729
2	0.729
3	0.738
4	0.737
5	0.741
6	0.741
7	0.732
8	0.734
9	0.732
10	0.732

Distance-based weighting, which assigns weights inversely proportional to the distance from the query point, was adopted for weight calculations. Based on the cross-validation results, the value of K=6 was chosen.

Following the methodology used for Naive Bayes (NB), 12 k-Nearest Neighbors (kNN) models were developed and implemented using the kNN tool from the scikit-learn library. These models employed a kk-value of 6 and incorporated the distance weighting method, as outlined in Table 5.6.

Table 5.6 Implemented kNN Models

Model Number	Model	Text Representation	n-gram	Test Set Ratio
13	kNN	BoW	Unigram	10
14				20
15				30
16			Mixed	10
17				20
18				30
19		TF-IDF	Unigram	10
20				20
21				30
22			Mixed	10
23				20
24				30

5.3.3 Linear Support Vector Machine Models

The Linear SVM class from the scikit-learn library was utilized for model development in the Python environment. Since the SVM algorithm is inherently a binary classifier, linear SVM employs the "one-vs-the-rest" strategy for multi-classification tasks. This approach involves creating separate sub-models for each class, treating one class as positive and the remaining classes as negative. For a 5-class classification problem, this results in 5 sub-models. The final prediction is made by selecting the class with the highest probability among the 5 sub-models.

The "squared hinge" loss function with L2 regularization was chosen for model convergence. The objective of cross-validation was to determine the optimal value for the regularization strength, represented by parameter C. Various C values, ranging from 0.001 to 5, were iterated during cross-validation.

It was observed that the macro F1 score increased rapidly after C=0.5, reached its peak at C=1, and decreased slightly thereafter, as shown in Table 5.7. Consequently, the C parameter for the implementation of Linear SVM models was set to 1.

Table 5.7 Macro F1 Scores based on C-Parameters of Linear SVM

C	Macro F1
0.001	0.492
0.5	0.817
1.0	0.822
1.5	0.821
2.0	0.820
2.5	0.818
3.0	0.811
3.5	0.811
4.0	0.809
4.5	0.808
5.0	0.809

Linear SVM classifier was implemented on three different train-test ratios of dataset with 6 different combinations of text representations using the above parameters as presented in Table 5.8. The models utilized BoW, TF-IDF, pre-trained Google W2V, and expanded pre-trained Google W2V vectorizations.

Table 5.8 Implemented Linear SVM Models

Model Number	Model	Text Representation	n-gram	Test Set Ratio
25	SVM-Linear	BoW	Unigram	10
26				20
27				30
28			Mixed	10
29				20
30				30
31		TF-IDF	Unigram	10
32				20
33				30
34			Mixed	10
35				20
36				30
37		Google W2V		10
38				20
39				30
40		Exp. Google W2V		10
41				20
42				30

5.3.4 Support Vector Machine with RBF Kernel Models

Support Vector Machine with Radial Basis Function kernel is a variant of the SVM algorithm that allows the classification of data non-linearly in high-dimensional space in contrast to linear SVM. The RBF kernel supports non-linear and more complex data to be classified with flexible boundaries rather than linear. However, another parameter, gamma, is introduced with RBF besides the regularization parameter of C. This makes the model more complicated to fine-tune. Gamma specifies the level of a single training example's effect. The higher gamma value influences the other examples less.

The scikit-learn library was used to cross-validate and develop the SVM with the RBF model. Though there were two distinct parameters to tune, the GridSearchCV tool of scikit-learn was utilized, which creates combinations of two parameters to

maximize the score during cross-validation. SVM with RBF models uses “hinge” loss function with L2 regularization by default. The proposed parameter combination with GridSearchCV was gamma=0.01 and C=250 with 4 k-folds.

18 different SVM with RBF fine-tuned models created with the scikit-learn library were implemented on 10%, 20%, and 30% test ratios, as shown in Table 5.9. Unlike linear SVM, SVM with RBF follows a one-vs-one approach for multiclassification. The model internally develops a binary classification model between classes. For a five-class classification problem, 10 different binary classification models were developed.

Table 5.9 Implemented SVM with RBF Models

Model Number	Model	Text Representation	n-gram	Test Set Ratio
43	SVM-RBF	BoW	Unigram	10
44				20
45				30
46			Mixed	10
47				20
48				30
49		TF-IDF	Unigram	10
50				20
51				30
52			Mixed	10
53				20
54				30
55		Google W2V		10
56				20
57				30
58		Exp. Google W2V		10
59				20
60				30

5.3.5 Logistic Regression Models

Logistic Regression was another shallow ML algorithm that was developed and implemented in the research. For multiclass cases, the LR tool of the scikit-learn library operates to minimize multinomial loss function having L2 regularization with the optimizer saga, which is a version of Stochastic Average Gradient (sag) and optimizes faster in large data sets. The C-regularization strength parameter was the target to determine in the cross-validation step. C values ranging from 0.1 to 500 were iterated through the LR validation model as values seen in Table 5.10. The macro F1 score attains its maximum value at C=100, followed by a minor decrease.

Table 5.10 Macro F1 Scores based on C-Parameters of LR

C	Macro F1
0.1	0.677
1	0.782
10	0.802
100	0.803
250	0.803
500	0.801

Table 5.11 details the implementation of fine-tuned Logistic Regression models, which followed a similar strategy to that utilized in linear SVM and SVM with RBF. 18 models were executed with 4 different text representations on 3 test ratio sets.

Table 5.11 Implemented LR Models

Model Number	Model	Text Representation	n-gram	Test Set Ratio
61	LR	BoW	Unigram	10
62				20
63				30
64			Mixed	10
65				20
66				30
67		TF-IDF	Unigram	10
68				20
69				30
70			Mixed	10
71				20
72				30
73		Google W2V		10
74				20
75				30
76		Exp. Google W2V		10
77				20
78				30

5.3.6 TextCNN Models

The development of the TextCNN models in this study followed the primary model created by Kim (2014) with the exception that a dense layer was added between the connected and classifier layers as an intermediate step to decrease the dimensionality of the feature maps. Figure 5.4 represents the developed architecture of the TextCNN model.

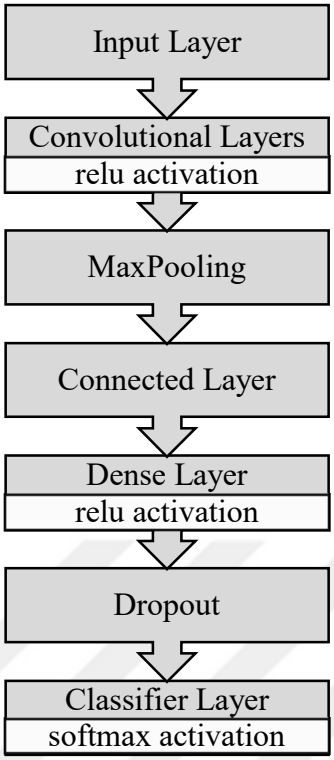


Figure 5.4. Developed TextCNN Architecture

Tuning any DL algorithm is comprehensive, unlike shallow learning ML algorithms. Due to the considerable number of parameters in a DL algorithm, it is a deep and expensive computation process. Y. Zhang & Wallace (2015) studied the sensitivity of the primary model to the parameters and observed that some attributes had governed the others regarding the model performance. They recommended a series of tuning steps which were followed in this research while developing the TextCNN model.

The same 80% of the data used to tune shallow ML models was the input for fine-tuning hyperparameters. The 4 k-fold cross-validation approach was employed to observe the macro F1 scores to evaluate the models' performance. A pre-trained Google W2V model from the Gensim library in Python was selected as the text representation for all tested models. The Keras library was the main backbone for constructing the TextCNN architecture in Python.

Common assumptions based on the recommendations were using the Relu activation function for convolutional and dense layers and a softmax function for the classifier layer. It was also advised to select a smaller dropout rate so 0.2 was chosen as a dropout for all models. Moreover, as the performance of the non-static version was better than static in previous research, a similar approach was applied in this study, which means that the pre-trained word embeddings were fine-tuned during the backpropagation of the convolutional layers.

The first step was discovering the best single kernel size, giving the highest score in the above TextCNN architecture. Initially, two parallel kernels with the same filter region were included in the convolutional layer. An average of 15 tokens per sentence was observed in the specification text in the section 0. Hence, it was decided to limit the maximum kernel size to 6. Also, there wasn't any recommendation about the epoch size of the TextCNN architecture. Epochs of a DL model decide how many times train data to pass through each neural network layer, which affects the model's learning of patterns. Consequently, besides iterating various kernel sizes, training, and testing models in different epochs was another necessity. Figure 5.5 demonstrates the first step in fine-tuning TextCNN models for kernel sizes 3, 4, 5, and 6 and epochs 5, 10, and 20. It was observed that while epochs are 20, the F1 score (0.855) of kernel size 3 governs the others. With a minor low value of 0.854, kernel size is positioned second.

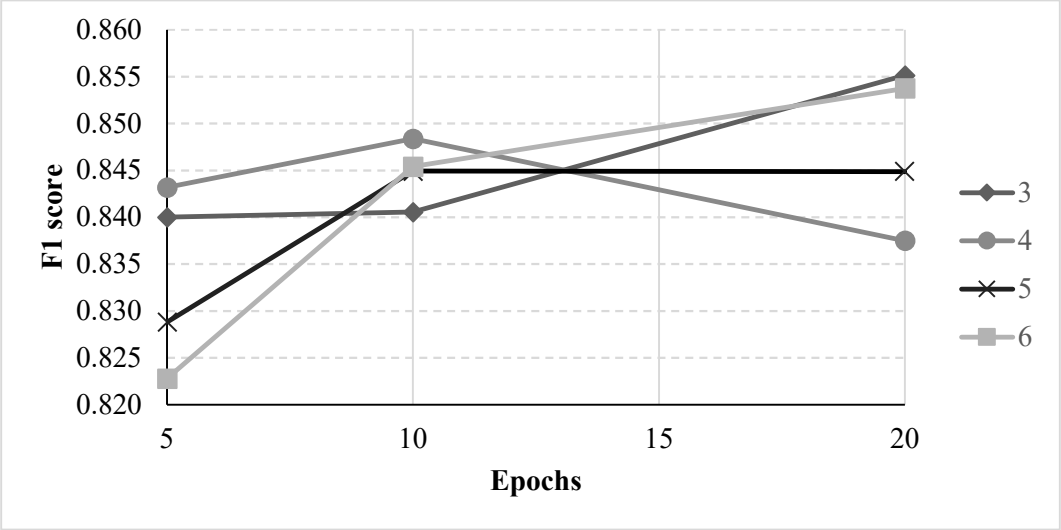


Figure 5.5. Epochs vs F1 Scores for Various Kernels

The next step was to develop kernel sets around the best single kernels to map the features concurrently in convolutional layers. Since the kernel size of 6 performed very close to 3, it was also selected as a second center of the set. Two sets were developed; the first one was from kernels 3, 4, and 5, and the other set was 5, 6, and 7. The set composed of 3, 4, and 5 kernels outperformed with an F1 score of 0.852, as shown in Table 5.12.

Table 5.12 F1 Scores According to Kernel Sets

Kernel Sets	Epoch	Filter Size	Precision	Recall	F1	Accuracy
3.4.5	20	128	0.873	0.838	0.852	0.966
5.6.7	20	128	0.867	0.825	0.841	0.966

The final step was the determination of filter size, in other words, the dimensionality of the feature maps from the input layer. The filter size also establishes the number of weights used in the convolutional layer. Up to this step, filter size was assumed as 128. Checking the model's performance with the set parameters for higher filter sizes was the last step of fine-tuning. Three filter sizes were considered initially 128, 256, and 384. In case there might be an increase in the F1 score, more upper levels would be tested. However, as spotted in Figure 5.6, the F1 scores of the model

decreased while the filter size increased. Therefore 128 was selected as the best option for the model.

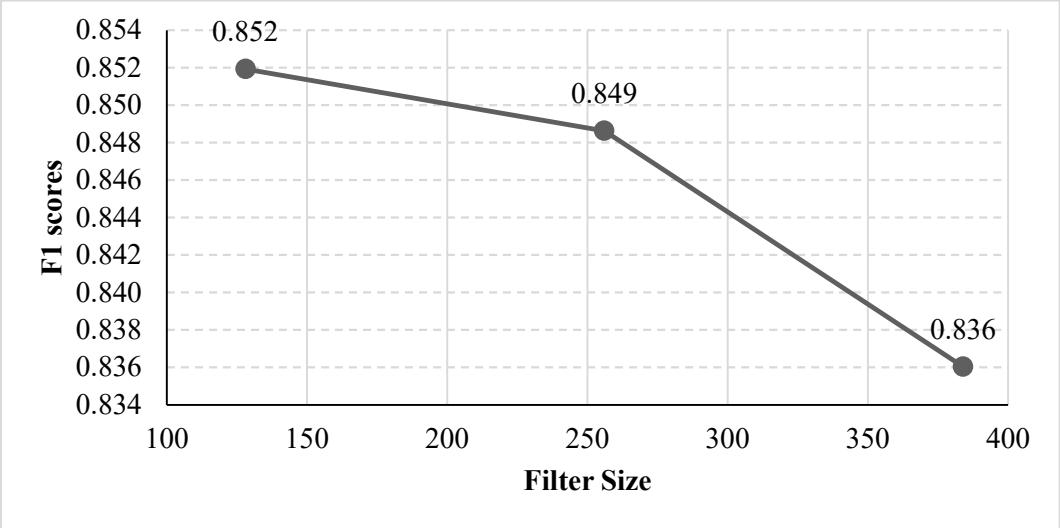


Figure 5.6. F1 Score vs Filter Size

After fine-tuning the TextCNN model, the following hyperparameters were decided to be implemented in the models for various text representations and test set percentage combinations.

Kernel set sizes= 3, 4, 5

Epochs= 20

Filter Size=128

Dropout= 0.2

6 models were developed with the TextCNN algorithm with pre-trained Google W2V and expanded pre-trained Google W2V text representations in Python environment with the advantage of the Keras library. All models were trained correspondingly and tested on 10%, 20% and 30% of data sets (Table 5.13). The implementation of each model was performed in two stages; firstly, the model was trained with a learning rate of 0.001 and tested. In the second stage, the same model was trained with a learning rate of 0.00001 to observe the fine-tuning performance of the same model with a lower learning rate.

Table 5.13 Implemented TextCNN Models

Model Number	Model	Text Representation	Test Set Ratio
79	TextCNN	Google W2V	10
80			20
81			30
82		Exp. Google W2V	10
83			20
84			30

5.3.7 BERT & RoBERTa Models

BERT and RoBERTa are the two-transformer based fine-tuned models as described in sections 3.2.6 and 3.2.7. Once a classifier layer is added to the end of the pre-trained model, the overall model becomes a combined text representation and classification model. Therefore, minor optional parameters are required while implementing BERT and RoBERTa models with the KerasNLP library.

The first one was to limit the sequence length to 128 instead of 512. Since the longest sentence has 90 tokens, it would be favorable to remove unnecessary length to ease the computational efforts.

Epochs are also a determinative parameter in the performance of transformer models. Due to the high requirements of running efforts for BERT and RoBERTa, 10 and 20 epochs were considered as alternatives in implementation to all data sets in preference to conduct a cross-validation step. Finally, each alternate model was applied in two continual steps, such as a learning rate of $5\ 10^{-5}$ and then with a lower value of $5\ 10^{-6}$.

12 separate train-test combinations were implemented in Python with KerasNLP, of which 6 with BERT and 6 with RoBERTa as seen in Table 5.14.

Table 5.14 Implemented BERT & RoBERTa Models

Model Number	Model	Text Representation	Epochs	Test Set Ratio
85	BERT	BERT Base Emb.	10	10
86				20
87				30
88			20	10
89				20
90				30
91	RoBERTa	RoBERTa Base Emb.	10	10
92				20
93				30
94			20	10
95				20
96				30

CHAPTER 6

PERFORMANCE RESULTS

6.1 Evaluation Strategy

A total of 96 distinct applications were performed using 8 ML algorithms combined with 6 different text representation techniques, as described in section 5.3 to identify and categorize deficiencies in construction specifications. Table 6.1 provides a summary of the implemented models.

Table 6.1 Numbering of Implemented ML Models

ML Models	Text Representation Techniques					
	BoW	**TF-IDF**	**Google W2V**	**Exp. Google W2V**	**BERT Emb.**	**RoBERTa Emb.**
NB	1,2,3,4, 5,6	7,8,9,10, 11,12				
kNN	13,14,15, 16,17,18	19,20,21, 22,23,24				
SVM-Linear	25,26,27, 28,29,30	31,32,33, 34,35,36	37,38,39	40,41,42		
SVM-RDF	43,44,45, 46,47,48	49,50,51, 52,53,54	55,56,57	58,59,60		
LR	61,62,63, 64,65,66	67,68,69, 70,71,72	73,74,75	76,77,78		
TextCNN			79,80,81	82,83,84		
BERT					85,86,87, 88,89,90	
RoBERTa						91,92,93, 94,95,96

The evaluation metrics and confusion matrices for each model listed in the table were calculated using the scikit-learn Python library. The class distribution was found to be skewed towards Group-0, representing the "no defect" category, as shown in

Figure 6.1. In skewed distributions, the accuracy metric can be misleading, as high accuracy rates often result from the dominance of the majority class.

To address this, the macro F1 score, which is the arithmetic mean of the F1 scores for each prediction class, was selected as the primary metric for evaluating overall model performance. Similarly, the weighted F1 score, which is influenced by class imbalance, was also deemed unsuitable as it could lead to the same inaccurate interpretation as accuracy.

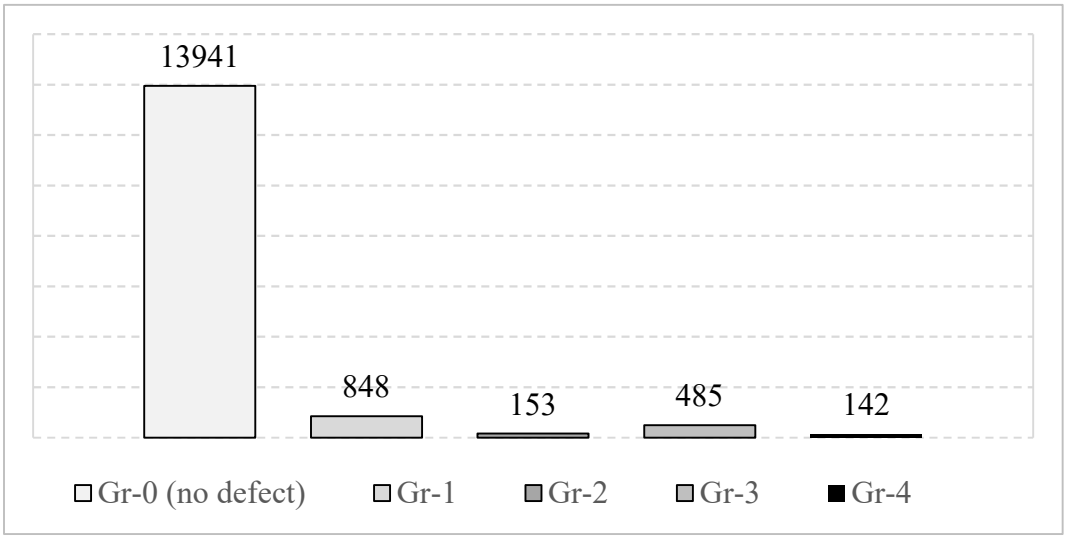


Figure 6.1. Frequencies of Defect Groups

6.2 Performances of Implemented ML Classification Models

The results of each implemented ML model type are presented in this section separately. Comparative analyses are presented afterward.

6.2.1 NB Models

The highest macro F1 score of 0.692 was achieved by Model-7, which utilized the unigram TF-IDF text vectorization method on a 90% train and 10% test dataset split, as shown in Table 6.2. Subsequently, the unigram TF-IDF combination on a 20%

test split (Model-8) followed with a macro F1 score of 0.689. The mixed n-gram approach on a 20% test ratio (Model-11) ranked third among the NB models reaching a macro F1 score of 0.686.

Table 6.2 Performances of NB Models

Model #	T.R.	Type	Test Set	F1	Precision	Recall	Accuracy
1	BoW	Unigram	10%	0.676	0.600	0.816	0.911
2	BoW	Unigram	20%	0.685	0.612	0.802	0.915
3	BoW	Unigram	30%	0.651	0.576	0.780	0.906
4	BoW	Mixed	10%	0.670	0.594	0.826	0.909
5	BoW	Mixed	20%	0.681	0.599	0.836	0.914
6	BoW	Mixed	30%	0.644	0.565	0.803	0.905
7	TF-IDF	Unigram	10%	**0.692**	0.746	0.687	0.931
8	TF-IDF	Unigram	20%	**0.689**	0.747	0.662	0.936
9	TF-IDF	Unigram	30%	0.656	0.700	0.645	0.928
10	TF-IDF	Mixed	10%	0.684	0.623	0.802	0.920
11	TF-IDF	Mixed	20%	**0.686**	0.615	0.816	0.922
12	TF-IDF	Mixed	30%	0.660	0.589	0.793	0.915

It was observed that, except for the unigram TF-IDF text representation, other combinations consistently developed their performance on the 20% test ratio as compared to the 10% test ratio but experienced a sudden decline at the 30% test ratio. All results for the 30% test ratio were evaluated as the poorest among the 12 implementations (Figure 6.2).

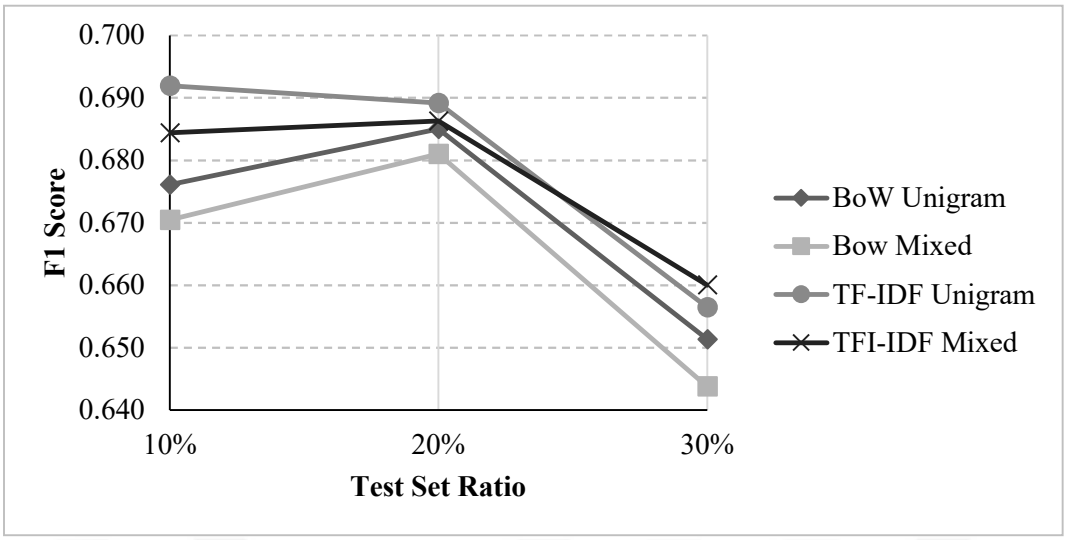


Figure 6.2. Test Set Ratio vs F1 Scores of NB Models

It was also pointed out that the NB models with BoW vectorizations, both with unigram and mixed n-gram approaches, have lower performances compared to TF-IDF (Figure 6.3).

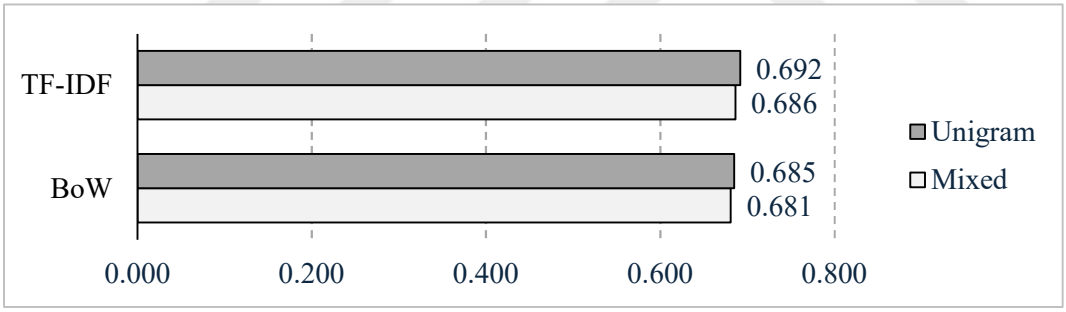


Figure 6.3. Max. F1 Scores of NB Models for Different Text Representations

6.2.2 kNN Models

The top three models based on macro F1 scores among 12 kNN models were Models 16, 13, and 14 respectively. Model 16, utilizing unigram BoW text representation on 10% of test data, outperformed the others with a score of 0.813. Model 13 and 14 with a very slight difference performed as second and third with scores of 0.800 and

0.799. Comparing the best and worst models revealed a 4.1% difference in macro F1 score, favoring the best-performing model.

Table 6.3 Performances of kNN Models

Model #	T.R.	Type	Test Set	F1	Precision	Recall	Accuracy
13	BoW	Unigram	10%	**0.800**	0.869	0.747	0.955
14	BoW	Unigram	20%	**0.799**	0.863	0.750	0.957
15	BoW	Unigram	30%	0.786	0.855	0.739	0.957
16	BoW	Mixed	10%	**0.813**	0.894	0.753	0.958
17	BoW	Mixed	20%	0.785	0.858	0.732	0.955
18	BoW	Mixed	30%	0.778	0.861	0.723	0.957
19	TF-IDF	Unigram	10%	0.787	0.818	0.761	0.955
20	TF-IDF	Unigram	20%	0.772	0.812	0.741	0.953
21	TF-IDF	Unigram	30%	0.778	0.821	0.745	0.955
22	TF-IDF	Mixed	10%	0.780	0.809	0.755	0.954
23	TF-IDF	Mixed	20%	0.782	0.822	0.754	0.953
24	TF-IDF	Mixed	30%	0.774	0.824	0.742	0.954

The sensitivity of the kNN algorithm with different vectorizations to the test set ratio is not as consistent as NB models, as illustrated in

Figure 6.4. Model 16, which has the highest score in the 10% test set ratio, had a sharp decrease in the 20% test ratio (Model 17), locating below Model 14. This attitude was a relatively considerable fall in 2.8% of the macro F1 score. Another remarkable observation was that almost all models have lower scores on the 30% test ratio except the unigram TF-IDF models. Unigram TF-IDF models accomplished a slightly higher score of 0.778 compared to the 20% test (0.772).

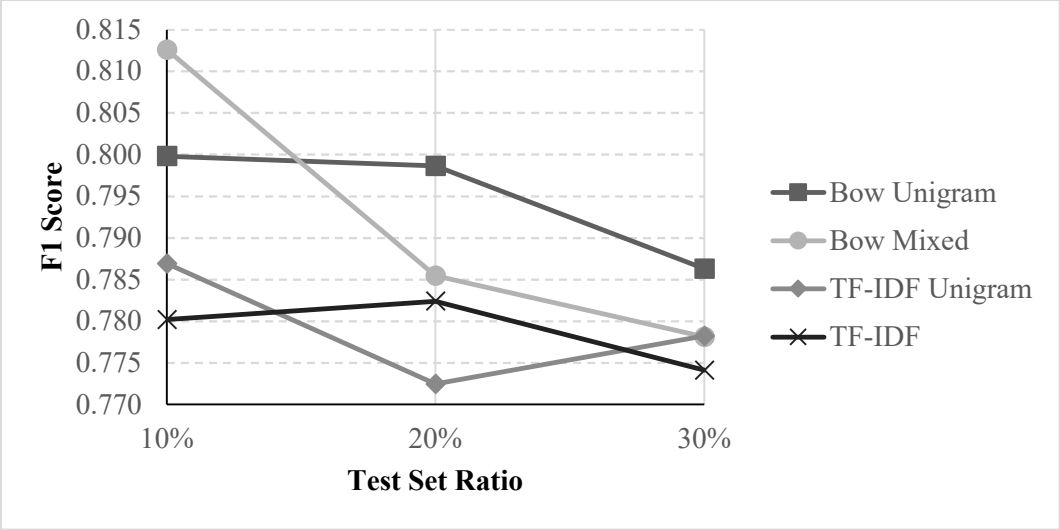


Figure 6.4. Test Set Ratio vs F1 Score of kNN Models

kNN models using BoW vectorization surpassed the TF-IDF ones both in unigram and mixed n-gram approaches in contrast to NB models (Figure 6.5).

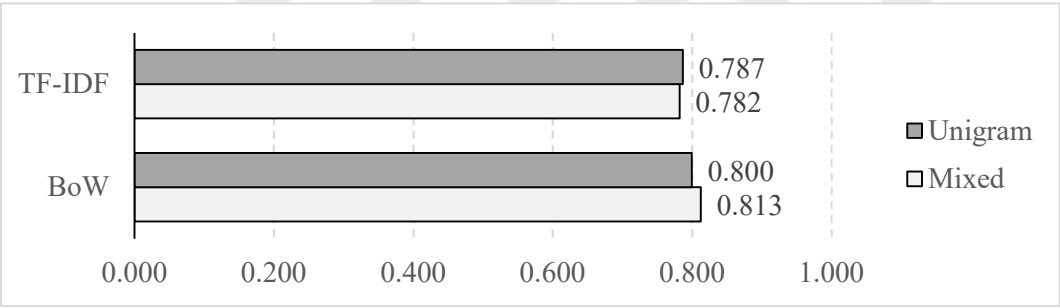


Figure 6.5. Max. F1 Scores of kNN Models for Different Text Representations

6.2.3 Linear SVM Models

Model 25, utilizing unigram BoW and implemented on a 10% test set, achieved the highest macro F1 score among the 18 linear SVM model combinations, as shown in Table 6.4. Afterward, Models 29 and 28, which employed mixed n-gram approaches, ranked second and third, respectively. The performances of the two semantic text representation methods, pre-trained Google W2V and its expanded version, were significantly worse than basic vectorization methods. For instance, Model 25 with a

macro F1 score of 0.866 performed almost 52% better than Model 38 having a macro F1 score of 0.569.

Table 6.4 Performances of Linear SVM Models

Model #	T.R.	Type	Test Set	F1	Precision	Recall	Accuracy
25	BoW	Unigram	10%	**0.866**	0.895	0.841	0.967
26	BoW	Unigram	20%	0.852	0.887	0.821	0.967
27	BoW	Unigram	30%	0.840	0.876	0.810	0.967
28	BoW	Mixed	10%	**0.853**	0.892	0.819	0.967
29	BoW	Mixed	20%	**0.859**	0.890	0.832	0.968
30	BoW	Mixed	30%	0.847	0.886	0.814	0.969
31	TF-IDF	Unigram	10%	0.849	0.906	0.801	0.967
32	TF-IDF	Unigram	20%	0.839	0.927	0.774	0.967
33	TF-IDF	Unigram	30%	0.829	0.917	0.766	0.968
34	TF-IDF	Mixed	10%	0.849	0.925	0.791	0.969
35	TF-IDF	Mixed	20%	0.846	0.923	0.788	0.969
36	TF-IDF	Mixed	30%	0.840	0.926	0.780	0.970
37	Google W2V		10%	0.572	0.706	0.515	0.924
38	Google W2V		20%	0.569	0.694	0.507	0.928
39	Google W2V		30%	0.575	0.681	0.519	0.929
40	Exp. Google W2V		10%	0.652	0.855	0.576	0.929
41	Exp. Google W2V		20%	0.650	0.918	0.560	0.934
42	Exp. Google W2V		30%	0.658	0.853	0.574	0.934

All models employing BoW and TF-IDF decreased their performances in the 30% test set as compared to 10%. The major fall was observed in models using BoW unigram approach in a percentage of 3 (Figure 6.6). In contrast to BoW and TF-IDF, a slight rise in F1 scores of models with pre-trained Google W2V and expanded version vectorizations was noticed over 30% of the test set in percentages of 0.5 and 0.9, respectively.

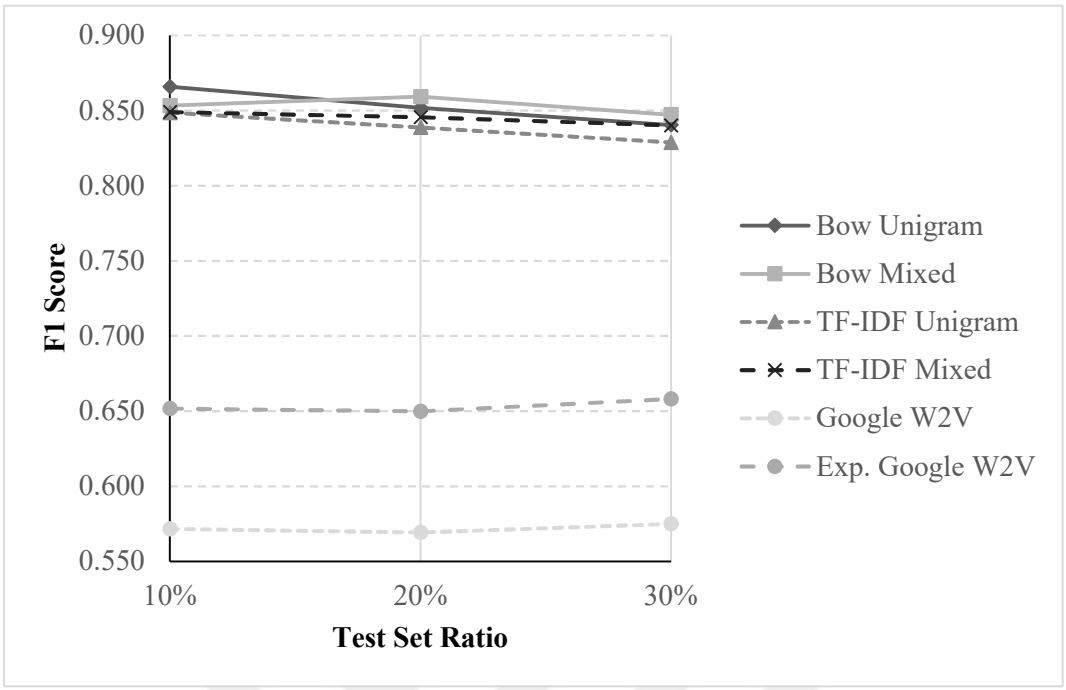


Figure 6.6. Test Set Ratio vs F1 Score of Linear SVM Models

Linear SVM models using BoW feature extraction performed superior to others, like the kNN models, while the maximum F1 score was 1.7% higher than TF-IDF. Although the expanded Google W2V vectorization ranked in the third stage, it was apparently noted that improving the vectorization of the Google W2V model resulted in a 14.4% increase in performance.

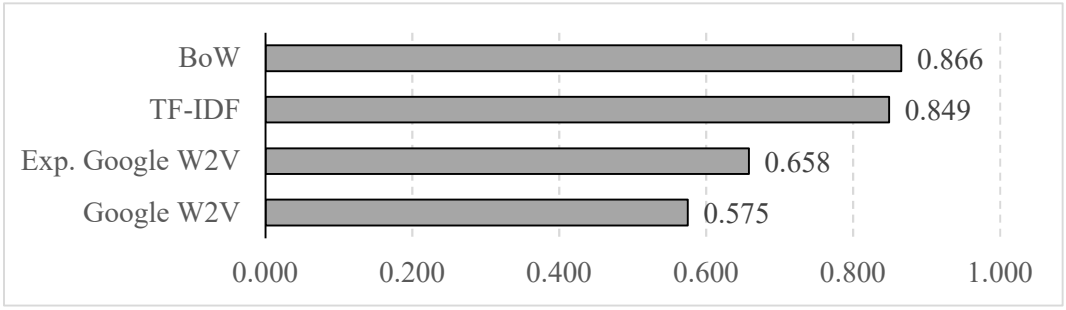


Figure 6.7. Max. F1 Scores of Linear SVM Models for Different Text Rep.

6.2.4 SVM-RBF Models

Models 52, 43, and 53 were the best-performed models among the 18 combinations created with SVM-RBF (Table 6.5). The macro F1 score (0.880) of Model 52, a combination of mixed n-gram TF-IDF on a 10% test set, was slightly higher than the macro F1 of Model 43 (0.877). Similar to the results of linear SVM models, SVM-RBF models with both pre-trained Google W2V and expanded pre-trained Google W2V performed poorer practices compared to basic statistical text representation techniques. An almost 23% difference in the F1 score was recognized between the best and worst models, which was noticeably lower than the observation in linear SVM.

Table 6.5 Performances of SVM-RBF Models

Model #	T.R.	Type	Test Set	F1	Precision	Recall	Accuracy
43	BoW	Unigram	10%	**0.877**	0.918	0.843	0.969
44	BoW	Unigram	20%	0.854	0.891	0.821	0.968
45	BoW	Unigram	30%	0.839	0.873	0.809	0.968
46	BoW	Mixed	10%	0.843	0.905	0.793	0.968
47	BoW	Mixed	20%	0.851	0.891	0.815	0.968
48	BoW	Mixed	30%	0.841	0.894	0.796	0.969
49	TF-IDF	Unigram	10%	0.860	0.892	0.833	0.969
50	TF-IDF	Unigram	20%	0.861	0.894	0.833	0.969
51	TF-IDF	Unigram	30%	0.841	0.878	0.809	0.968
52	TF-IDF	Mixed	10%	**0.880**	0.935	0.834	0.972
53	TF-IDF	Mixed	20%	**0.867**	0.906	0.832	0.970
54	TF-IDF	Mixed	30%	0.856	0.908	0.813	0.971
55	Google W2V		10%	0.678	0.911	0.594	0.939
56	Google W2V		20%	0.646	0.869	0.576	0.940
57	Google W2V		30%	0.675	0.885	0.595	0.943
58	Exp. Google W2V		10%	0.725	0.865	0.655	0.940
59	Exp. Google W2V		20%	0.716	0.872	0.631	0.943
60	Exp. Google W2V		30%	0.719	0.876	0.636	0.943

All models implemented using the SVM-RBF algorithm achieved lower F1 scores on the 30% test set compared to the 10% test set. While the unigram BoW option ranked second highest in performance on the 10% dataset, it experienced a notable decline of 3.8% in the F1 score when evaluated on the 30% test set, as shown in Figure 6.8. Additionally, the sensitivity of W2V models to variations in test set percentage was lower than that of the BoW and TF-IDF vectorizations.

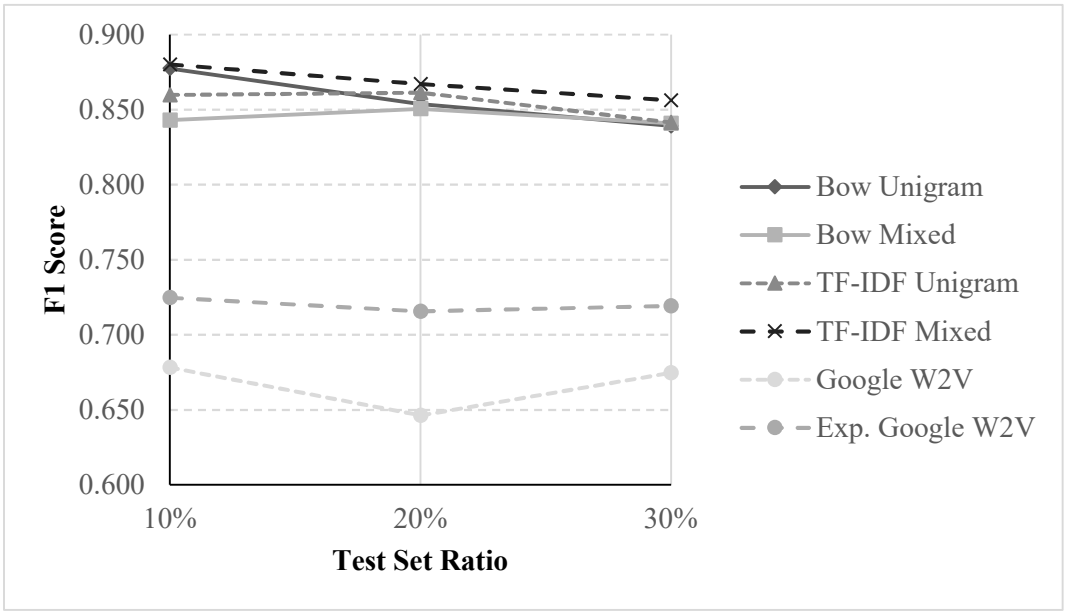


Figure 6.8. Test Set Ratio vs F1 Score of SVM-RBF Models

Both TF-IDF (0.880) and BoW (0.877) word vectorizations generated remarkably higher maximum macro F1 scores than W2V word embeddings as highlighted in Figure 6.9. Although it was lower in percentage than in linear SVM, improving W2V word embedding enhanced the performance of the SVM-RBF algorithm by 4.7%.

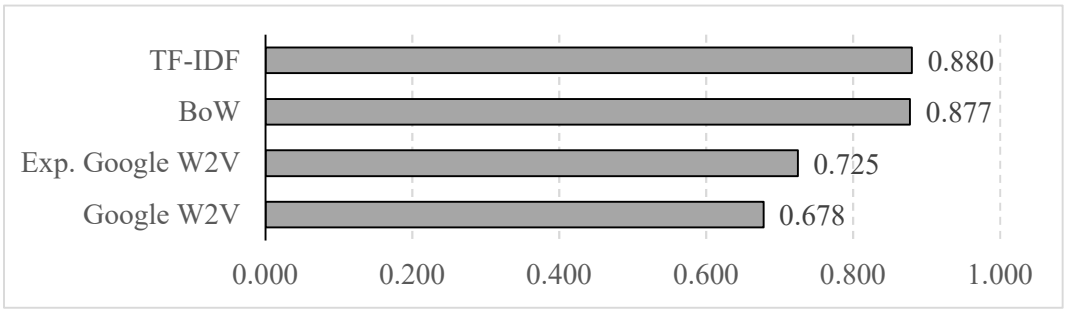


Figure 6.9. Max. F1 Scores of SVM-RBF Models for Different Text Rep.

6.2.5 LR Models

Model 67, which used unigram TF-IDF features on a 10% test set, performed slightly better than models 65 and 61 with a macro F1 score of 0.863 (Table 6.6). Model 65, which utilized mixed n-gram BoW implemented on 20% of dataset, achieved an F1 score of 0.861 while Model 61 using unigram BoW implemented on 10% test data resulted with a minor lower value of 0.859. Like SVM models, W2V word embedding alternatives exhibited the poorest performances among the LR models too.

Table 6.6 Performances of LR Models

Model #	T.R.	Type	Test Set	F1	Precision	Recall	Accuracy
61	BoW	Unigram	10%	**0.859**	0.903	0.821	0.968
62	BoW	Unigram	20%	0.858	0.896	0.825	0.969
63	BoW	Unigram	30%	0.826	0.862	0.795	0.966
64	BoW	Mixed	10%	0.846	0.909	0.795	0.969
65	BoW	Mixed	20%	**0.861**	0.916	0.815	0.972
66	BoW	Mixed	30%	0.842	0.906	0.793	0.969
67	TF-IDF	Unigram	10%	**0.863**	0.898	0.832	0.968
68	TF-IDF	Unigram	20%	0.856	0.909	0.812	0.969
69	TF-IDF	Unigram	30%	0.834	0.893	0.788	0.968
70	TF-IDF	Mixed	10%	0.838	0.918	0.780	0.969
71	TF-IDF	Mixed	20%	0.840	0.895	0.795	0.968
72	TF-IDF	Mixed	30%	0.833	0.906	0.778	0.969
73	Google W2V		10%	0.637	0.722	0.600	0.924
74	Google W2V		20%	0.659	0.732	0.616	0.927
75	Google W2V		30%	0.649	0.698	0.620	0.927
76	Exp. Google W2V		10%	0.713	0.810	0.656	0.933
77	Exp. Google W2V		20%	0.724	0.810	0.669	0.937
78	Exp. Google W2V		30%	0.712	0.774	0.666	0.935

Figure 6.10 demonstrated that the performance of the models using BoW and TF-IDF decreased when the train set ratio was decreased also, in other words, altering

the train-test sets from 90%-10% to 70%-30%. The highest fall was noticed in the unigram BoW with 3.3%. W2V embeddings performed contradicting results, such as pre-trained Google W2V increased 1.9% of its performance in a 30% test set, whereas the expanded version remained almost similar in both test ratios.

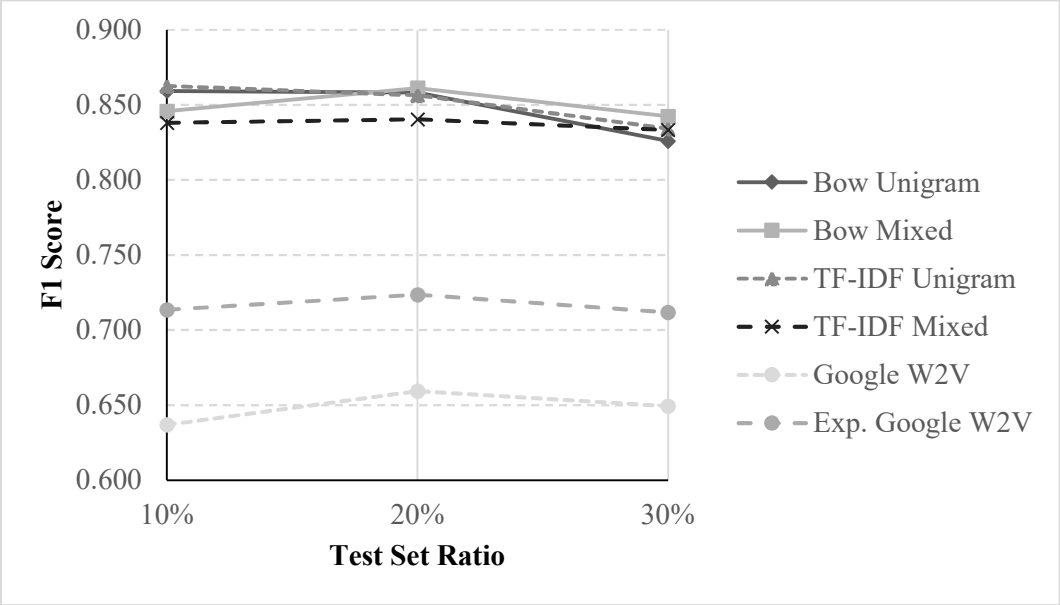


Figure 6.10. Test Set Ratio vs F1 Score of LR Models

The utilization of simple text representation techniques of TF-IDF and BoW in LR model implementations also dominated the word embedding alternatives (Figure 6.11). The TF-IDF feature extraction outcome had a maximum F1 score of 0.863, which was almost 31% better than the pre-trained Google W2V. Consistent with SVM results, the expanded word embedding uplifted the performance by about 10%.

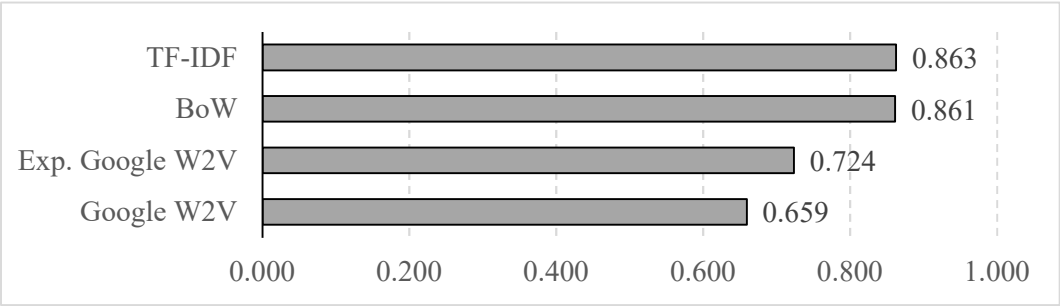


Figure 6.11. Max. F1 Scores of LR Models for Different Text Representations

6.2.6 TextCNN Models

6 models were developed and implemented with the TextCNN DL algorithm with two types of word embeddings: the pre-trained Google W2V and the expanded pre-trained Google W2V. The training process was conducted in two steps: initially, the models were trained with a higher learning rate, followed by retraining the same models with a lower learning rate to evaluate the impact of fine-tuning. Table 6.7 underlined that Models 82 and 83, utilizing the expanded version of W2V word embedding, scored the highest F1 metrics as 0.893 and 0.892, respectively, even in both the first training and the fine training steps. This was a consistent consequence with the outcomes of shallow learning ML models presented formerly. Model 82 performed almost 1.4% better than Model 79, which is both implemented on a 10% test set ratio.

Table 6.7 Performances of TextCNN Models

T.R.		Google W2V			Exp. Google W2V		
Model #		79	80	81	82	83	84
Test Set		10%	20%	30%	10%	20%	30%
1. Train-Test Learning Rate (1e-03)	**F1**	0.881	0.867	0.863	**0.893**	**0.892**	0.879
	Precision	0.906	0.910	0.886	0.908	0.905	0.907
	Recall	0.858	0.832	0.842	0.885	0.884	0.853
	Accuracy	0.969	0.969	0.970	0.970	0.972	0.970
Fine Train-Test Learning Rate (1e-05)	**F1**	0.885	0.885	0.864	**0.891**	**0.887**	0.875
	Precision	0.913	0.922	0.890	0.918	0.907	0.902
	Recall	0.859	0.852	0.842	0.870	0.872	0.850
	Accuracy	0.971	0.972	0.971	0.972	0.972	0.971

When the models' performances were examined according to the test set ratios, it was persistently revealed with prior models that the macro F1 scores tended to decrease as the size of the training set decreased. A DL model like TexCNN did not change this attitude even after the fine training step, as indicated in Figure 6.12. A

maximum decline in the percentage of 2.3 was observed in models with Google W2V word embedding after the fine learning steps.

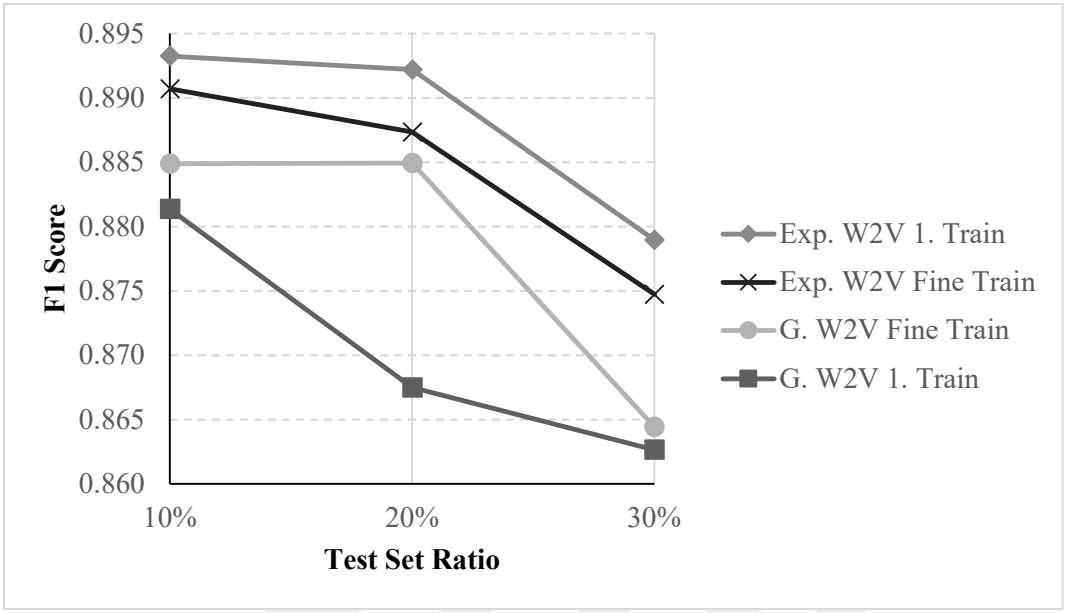


Figure 6.12. Test Set Ratio vs F1 Score of TextCNN Models

The fine learning had different impacts on the two forms of word embeddings as shown in Figure 6.13. The models using pre-trained Google W2V vectorization improved their performances after the fine learning step whereas the models with the expanded version had lower F1 scores than the first training.

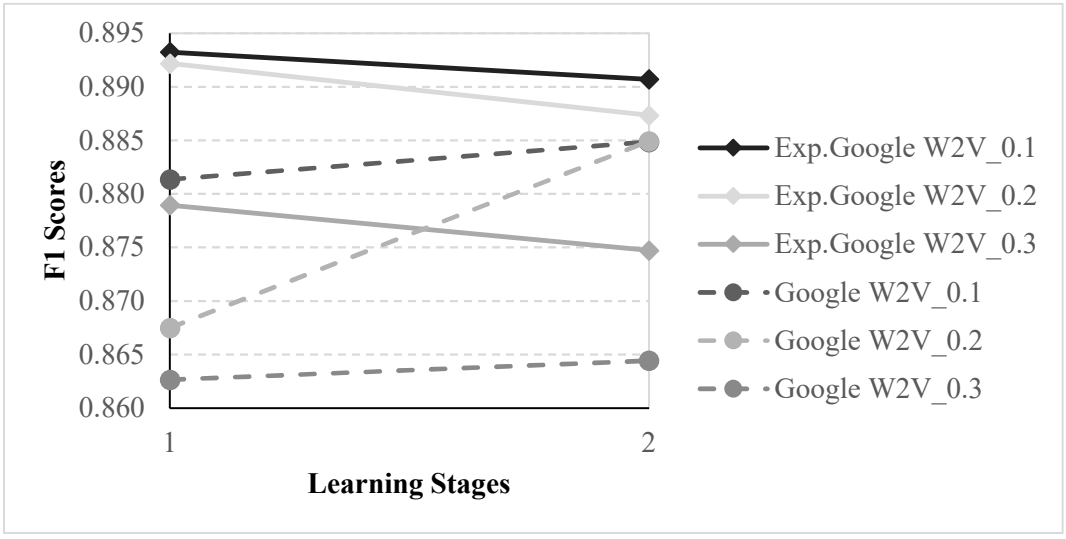


Figure 6.13. F1 Scores vs Learning Stages of TextCNN Models

6.2.7 BERT Models

The Implementation of BERT models carried out similar steps to the TextCNN models, covering an initial training and testing step followed by a fine-tuning step with a lower learning rate. All models utilized the pre-trained BERT word embedding model for text vectorization. Among 6 combinations tested, Model 86 generated the highest macro F1 scores both in the first and fine-tuning steps as highlighted in Table 6.8. The best performance, with a score of 0.907, was achieved testing on 20% of the dataset with 10 epochs, whereas the lowest one was Model 88 run in 20 epochs tested on 10% of the data with a score of 0.882. The difference in macro F1 score between the best and worst-performing models was approximately 2.5%.

Table 6.8 Performances of BERT Models

Epochs		10			20		
Model #		85	86	87	88	89	90
Test Set		10%	20%	30%	10%	20%	30%
1. Train-Test Learning Rate (5e-05)	**F1**	0.874	**0.902**	0.880	0.883	0.902	0.896
	Precision	0.895	0.915	0.880	0.887	0.906	0.915
	Recall	0.879	0.894	0.883	0.895	0.902	0.879
	Accuracy	0.967	0.974	0.972	0.969	0.975	0.975
Fine Train-Test Learning Rate (5e-06)	**F1**	0.893	**0.907**	0.892	0.882	0.895	0.894
	Precision	0.900	0.912	0.899	0.878	0.895	0.908
	Recall	0.897	0.909	0.886	0.901	0.900	0.882
	Accuracy	0.972	0.975	0.975	0.970	0.974	0.975

The response of the BERT models to various test set ratios contradicted all the models specified in the previous sections, where the lowest scores were obtained on 30% of test sets. However, almost all BERT models produced the lowest scores on 10% test set (Figure 6.14). The highest ones were consistently found in 20% of test data, whereas 30% of test sets were positioned in the second rank for both 10 and 20 epochs.

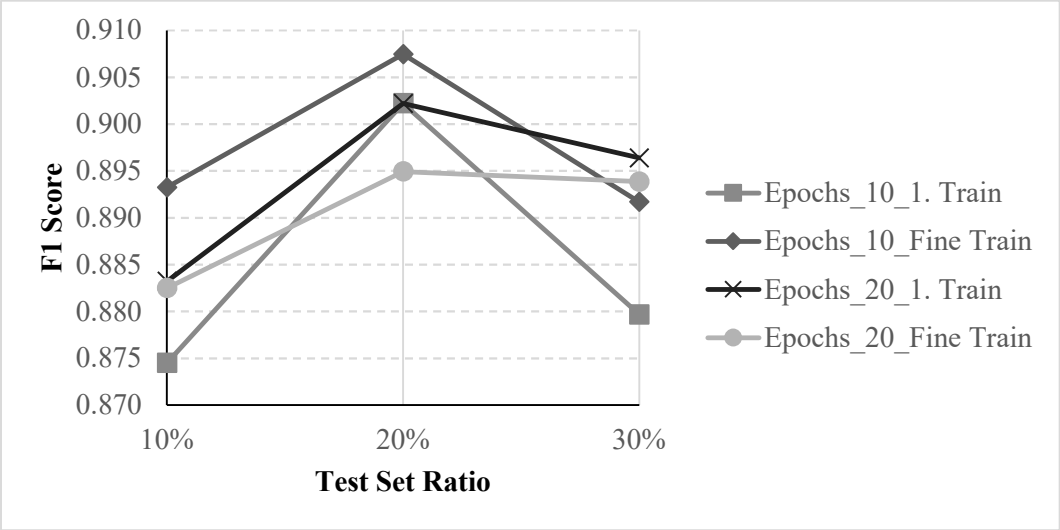


Figure 6.14. Test Set Ratio vs F1 Score of BERT Models

It was observed that lowering the learning rate in the second step inversely affected the model performances based on the epochs size (Figure 6.15). While all models trained in 10 epochs increased the F1 scores, the outcomes of the models trained in 20 epochs were negatively affected and conversely decreased after the fine learning stage.

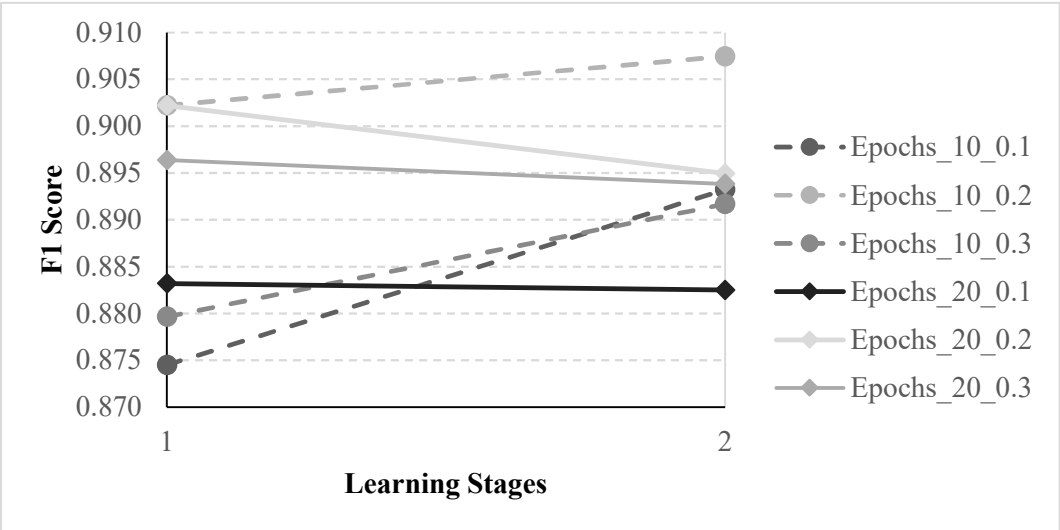


Figure 6.15. F1 Scores vs Learning Stages of BERT Models

6.2.8 RoBERTa

Similar combinations were applied to RoBERTa models as those employed for BERT, given that the development of RoBERTa is based on BERT. The highest macro F1 score of 0.912 was achieved in Model 92 after fine training with 10 epochs tested on 20% of the data set even though the initial testing outcome of 0.865 F1 score was one of the poorest combinations (Table 6.9). Contrarily, Model 94 dominated the other models in the first training and testing step with a score of 0.885. However, it dropped its performance to 0.879 F1 score after the fine-tuning step.

Table 6.9 Performances of RoBERTa Models

Epochs		10			20		
Model #		91	92	93	94	95	96
Test Set		10%	20%	30%	10%	20%	30%
1. Train-Test Learning Rate (5e-05)	**F1**	0.872	0.865	0.860	**0.885**	0.881	0.874
	Precision	0.879	0.917	0.827	0.899	0.897	0.865
	Recall	0.888	0.832	0.904	0.889	0.870	0.887
	Accuracy	0.967	0.966	0.969	0.970	0.971	0.973
Fine Train-Test Learning Rate (5e-06)	**F1**	0.894	**0.912**	0.902	0.879	0.892	0.900
	Precision	0.904	0.916	0.911	0.873	0.886	0.904
	Recall	0.899	0.911	0.895	0.901	0.902	0.898
	Accuracy	0.972	0.977	0.976	0.970	0.973	0.976

Another observation was that the performance of the models decreased as the training set ratio was reduced during the initial training phase. However, after fine-tuning, the F1 scores tested on the 30% dataset were higher than those tested on the 10% dataset, as illustrated in Figure 6.16.

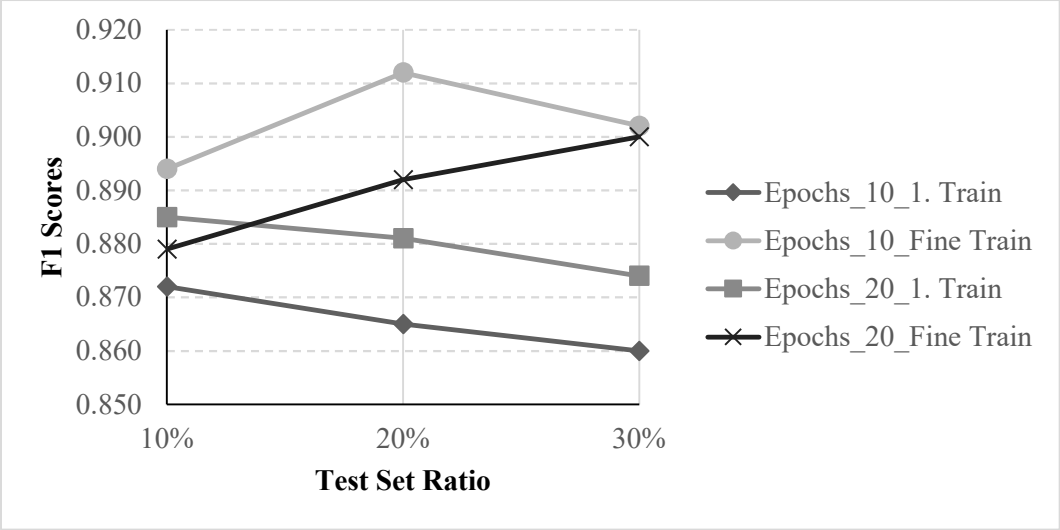


Figure 6.16. Test Set Ratio vs F1 Score of RoBERTa Models

It was captured from Figure 6.17 that except for model 92, all other models developed their performance after fine training, especially the models running in 10 epochs.

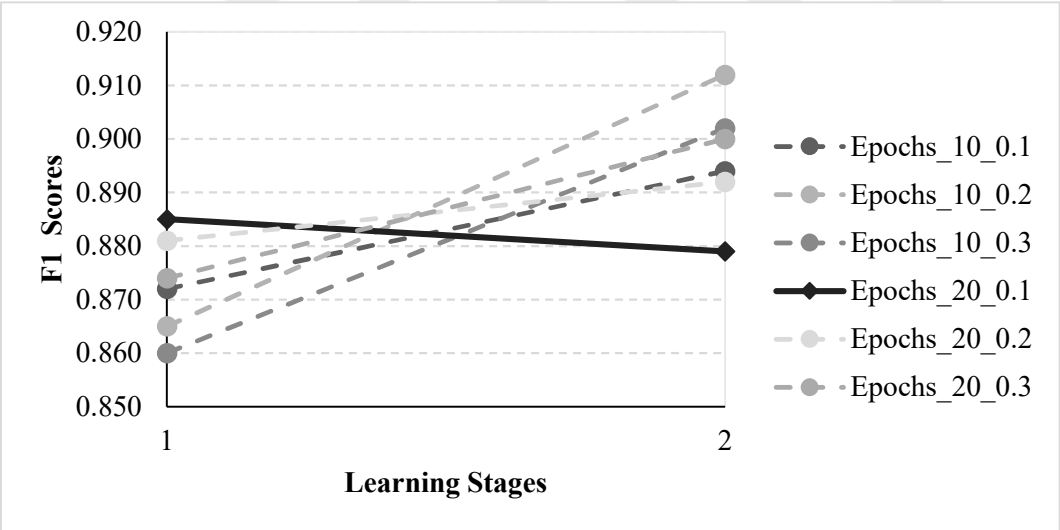


Figure 6.17. F1 Scores vs Learning Stages of RoBERTa Models

6.3 Comparative Evaluation

The previous section analyzed the implemented models' performances for defect identification and categorization in detail, focusing on combinations of the same algorithm with various text representation techniques and parameter settings. This section presents a broader comparison of the distinct algorithms utilized in the study. The best-performing combinations from each algorithm set were selected, analyzed, and compared with one another. The study assessed how each model interacted with text vectorization techniques and responded to different training sizes, providing insights from a macro perspective.

Figure 6.18 demonstrated the comparison of shallow ML models employing basic text representation methods of BoW and TF-IDF. With a macro F1 score of 0.880, the SVM-RBF algorithm generated the best model among the shallow ML algorithms. The properties and handling of non-linear data with adaptable classification boundaries of the SVM-RBF algorithm played a vital role in the five-class classification problem of this research. The poorest outcomes from models with kNN and NB algorithms proved that those algorithms were not proper for the classification purpose of this research.

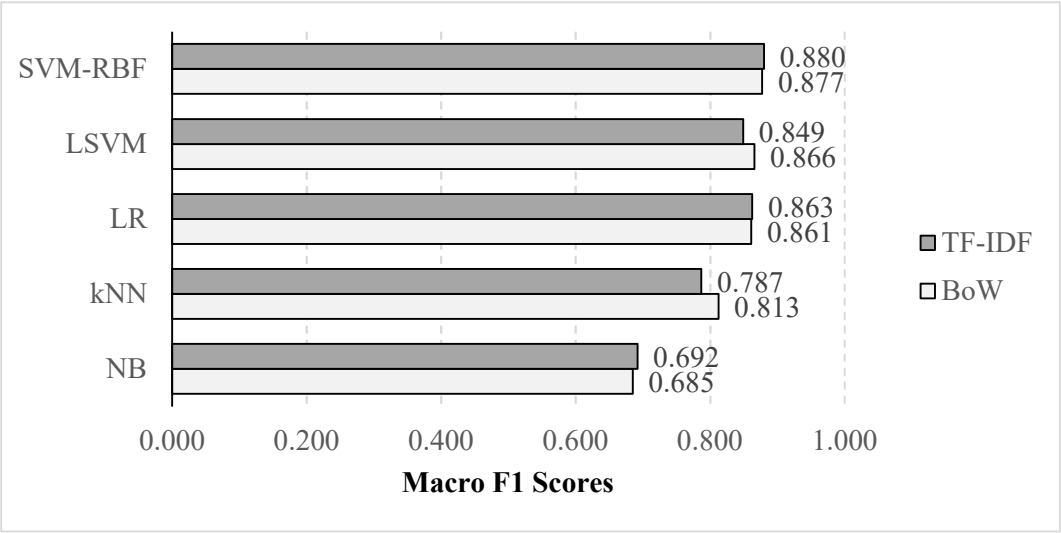


Figure 6.18. Comparison of Results of Shallow ML Models

A significant observation was that the shallow ML models were affected by the size of the train set; the larger the train data, the higher the model's performance. All models in this category generated the lowest scores in 70% train-30% test sets whereas the highest scores were in 90% train-10% test sets as shown in Figure 6.19. As the calculations are either statistical or algebraic in shallow ML algorithms, the more train data is available, the higher the model could represent the patterns embedded in the data.

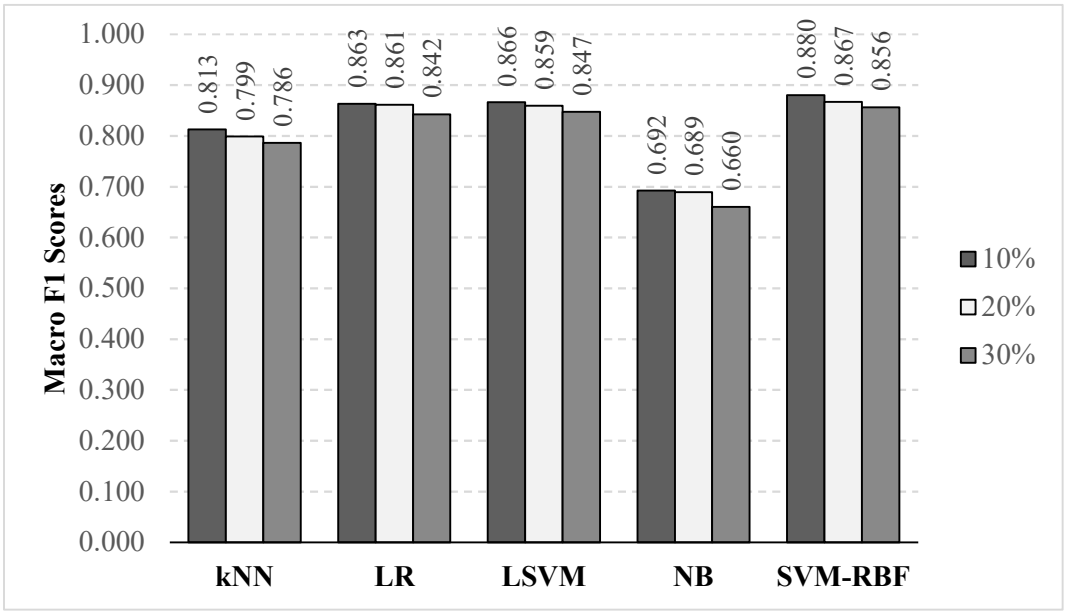


Figure 6.19. Performances of Shallow ML Models vs Test Sets

W2V word embeddings were utilized with shallow ML algorithms of LR, linear SVM, SVM-RBF and the DL algorithm, TextCNN. Shallow ML models with W2V word embeddings performed poorer than the same algorithms with BoW and TF-IDF text vectorizations as explained in previous sections individually. As TextCNN utilizes DL architecture with neural network computation, its performance with pre-trained word embeddings derived from Word2Vec demonstrated superiority to other methods, as depicted in Figure 6.20. It was a remarkable consequence that expanded pre-trained Google word embedding, improved with out-of-vocabulary derived from the data set, produced better classification performance than the original version, even with shallow ML models.

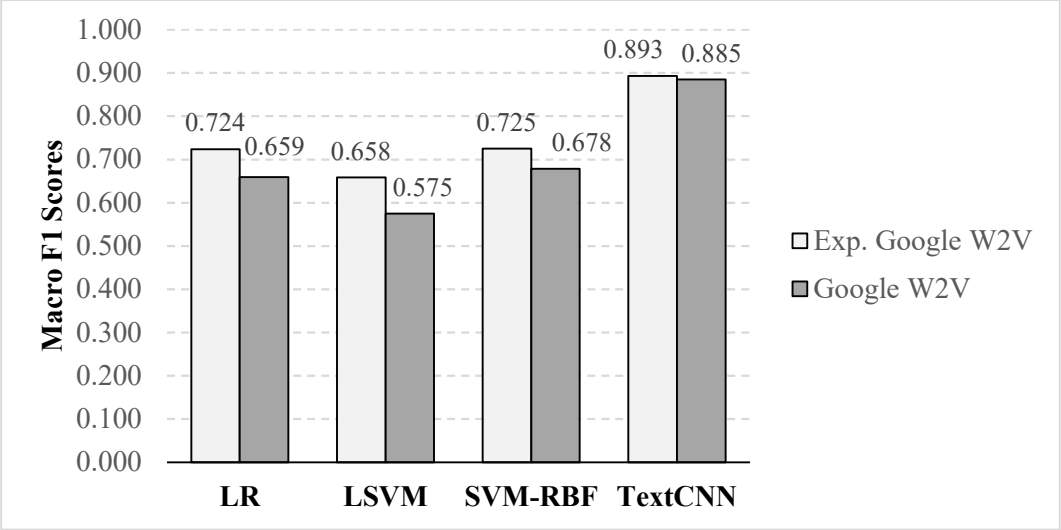


Figure 6.20. W2V Word Embedding Impact on Models

Previously, the responses of shallow ML models on several test set ratios were compared, and it was detected that the performance of the models decreased consistently with the declining training set. However, when the DL models were evaluated similarly, transformer models performed differently, as the 20% test set had the highest score; moreover, the 30% test set had better results than the 10% test set. This also demonstrated the dynamic learning mechanism of transformer models while training on the data set.

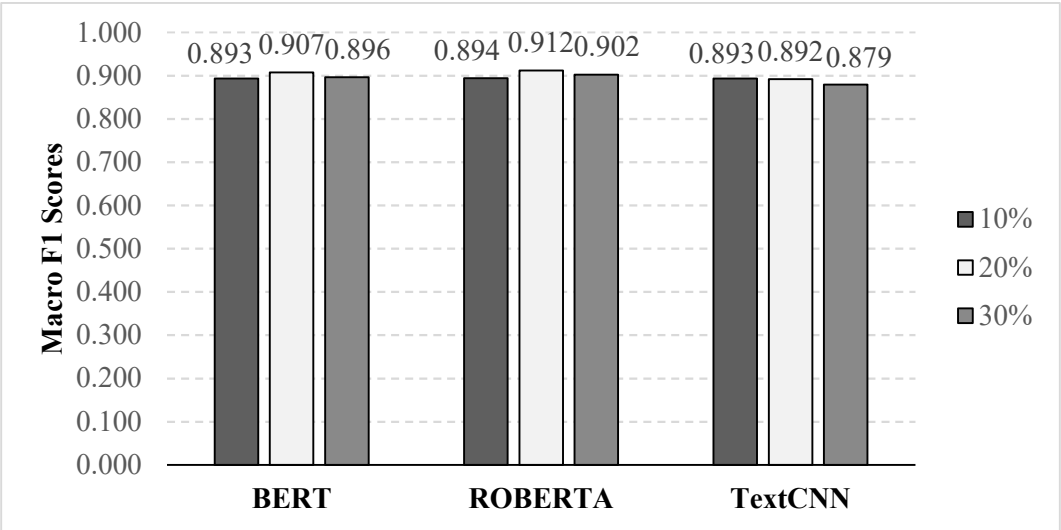


Figure 6.21. Performances of DL Models vs Test Sets

Although the combinations of each DL model generated conflicting results when the best score of each algorithm was compared, as in Figure 6.22. In contrast to TextCNN, the transformer models BERT and RoBERTa tended to increase their performances after the fine training stage with a lower learning rate.

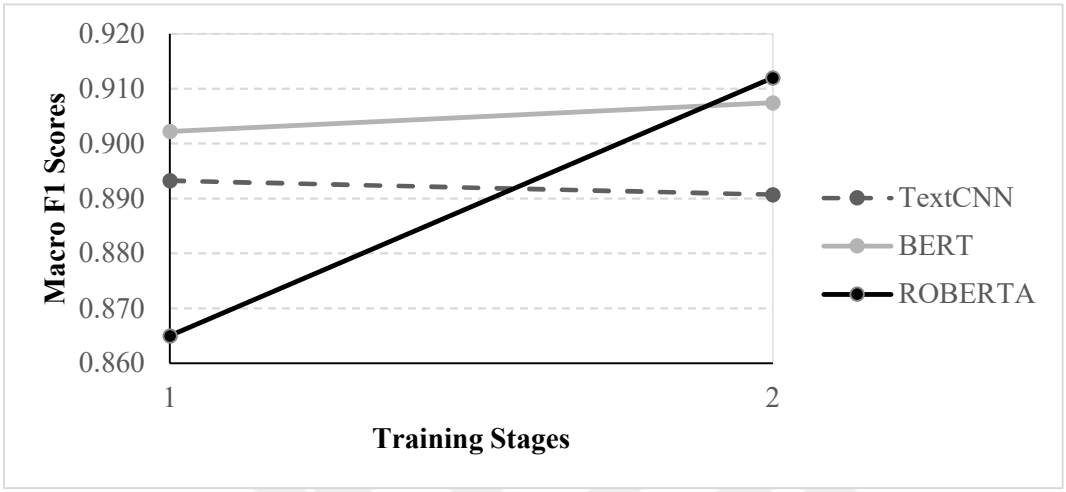


Figure 6.22. F1 Scores vs Learning Stages of DL Models

It was recognized that when a fine training step was applied to the transformer models, BERT and RoBERTa, epochs size of 10 achieved higher performances that could eliminate the necessity of larger epochs which also increase the computational cost (Figure 6.23)

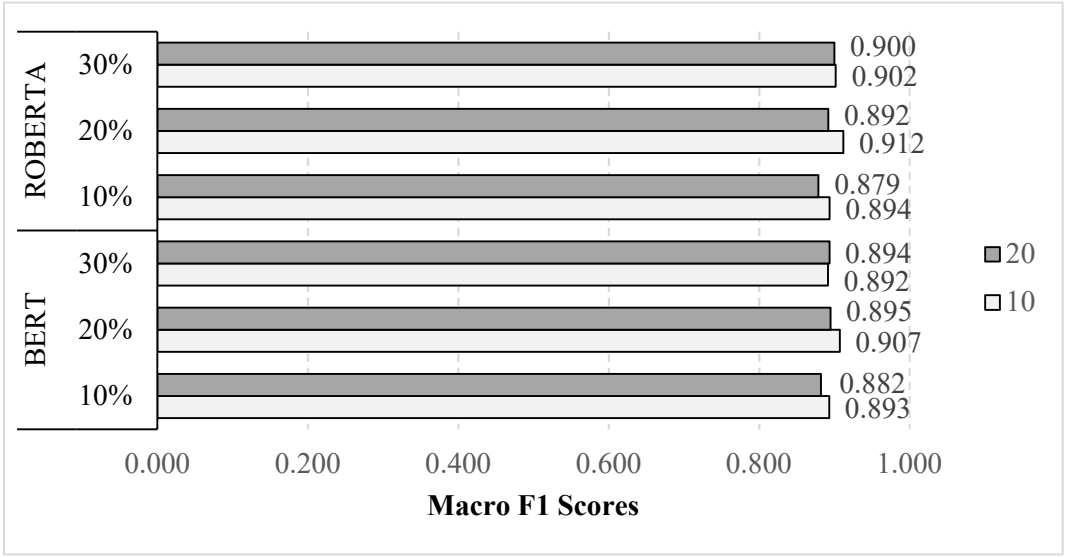


Figure 6.23. Performances of Transformers according to Epochs Size

The best model to detect and classify the deficiencies observed in construction specifications amongst the 96 distinct model combinations was Model 92, a RoBERTa model (Figure 6.24). Model 92 acquired its highest macro F1 score of 0.912 while training on an 80% data set and testing on a 20% data set. The worst performance was observed from the NB algorithm, which could only generate the highest macro F1 score of 0.692, which is almost 24% less than the performance of model 92.

The classification realization of the SVM-RBF algorithm was evaluated remarkably high, considering that model development was easier, and the running cost was manageable compared to DL models. It achieved the highest score of 0.880, which was only 3.5% less than the best model.

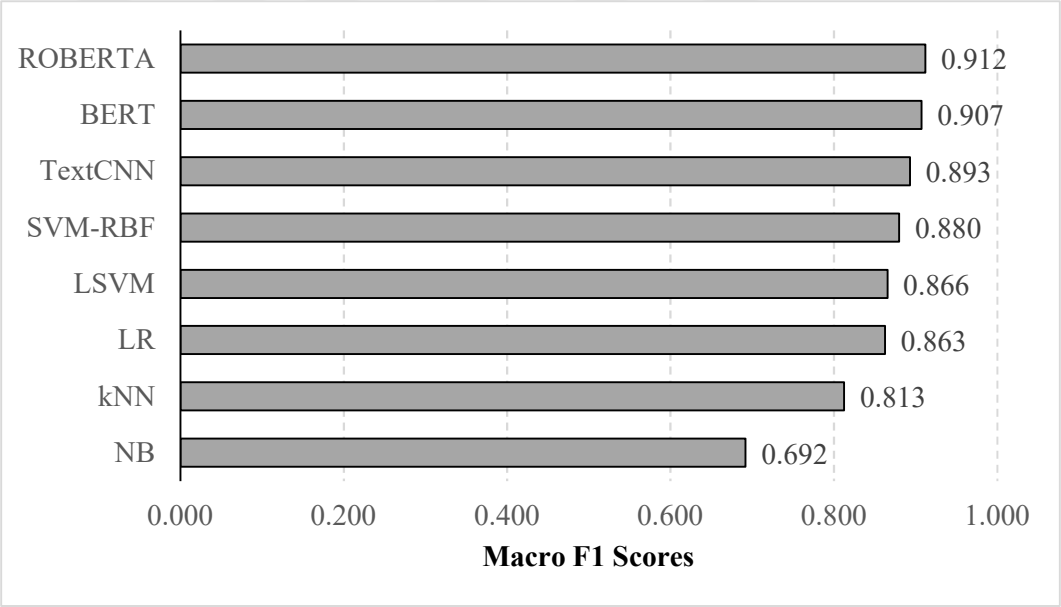


Figure 6.24. Highest Macro F1 Scores of Implemented Models

6.4 Validation of Results via Project-Based Investigation

The content of the dataset used in the research was described in detail in 4.3.2 Labeling Statistics. A total of 15,569 specification statements expressed in 21 different architectural works from 16 projects were constructing the complete data set. While splitting the train-test data, the whole data was shuffled to prevent bias, improve convergence, and balance the data between train and test sets. In other words, the labeled statements in a train set were selected randomly from all 16 projects, and they were provided to distribute any group of deficiencies clustered or focused on a project to the train and test data set uniformly. All implemented 96 ML models were based on this commonly followed data preparation.

A question was raised during the research: how well would those ML models perform if they were not trained with project-originated defects? Could the ML models predict the deficiencies in a completely new project based on the training of others? Therefore, another data-splitting strategy was created to investigate the answers to those questions.

A Python code was compiled that randomly selects a set of projects having a total number of statements above 1,500 which is almost 10% of the total data set. Five different test sets were created each having the statements of the selected projects as shown in Table 6.10. The frequency and distribution of deficiencies in the specifications of selected projects were not considered while generating the test sets. The remaining projects of the dataset were utilized distinctly for the training purpose of each test set. The test sets were not included in the training of ML models for each train-test effort.

Table 6.10 Test Sets of Project-Based Classification Investigation

Project No	Set-1	Set-2	Set-3	Set-4	Set-5
1		**test**			
2			**test**	**test**	
3		**test**			**test**
4				**test**	
5					
6					
7					**test**
8	**test**				
9					
10					
11					
12					
13	**test**			**test**	
14					
15			**test**		
16			**test**		

Three ML models representing shallow ML, DL, and transformer were selected to run in this part of the research as SVM-RBF, TextCNN, and RoBERTa. Similar hyperparameters were used as in the section 5.3. All 3 models were implemented using text representations giving the highest score in the previous section.

The average train-test percentages of 5 trial sets were calculated as 88% and 12% as indicated in Table 6.11. A maximum macro F1 score of 0.908 was achieved in Set-4 with RoBERTa. The average macro F1 scores of 5 sets for SVM-RBF, TextCNN, and RoBERTa models were calculated as 0.843, 0.853, and 0.886 individually.

Table 6.11 Performance Results of Project-Based Classification Investigation

Set #	Number of Statements (train)	Number of Statements (test)	Test Set Ratio	Macro F1 Scores		
				SVM-RBF	TextCNN	RoBERTa
1	13854	1715	11%	0.861	0.879	0.877
2	13402	2167	14%	0.869	0.866	0.882
3	13849	1720	11%	0.848	0.843	0.882
4	13759	1810	12%	0.839	0.875	0.908
5	13929	1640	11%	0.797	0.804	0.882
		Averages	**12%**	**0.843**	**0.853**	**0.886**
		Maximums		**0.869**	**0.879**	**0.908**

The outcomes of the project-based classification investigation were assessed against the results of the primary research. The averages and maximums of the three ML models presented in Table 6.11 were compared with the highest macro F1 scores gained with the same models on 10% of test sets.

The highest F1 score of 0.908, calculated with RoBERTa among the 5 sets of project-based classification investigation was remarkably higher than the research score of 0.894 obtained with RoBERTa on the 10% test set, although the average 0.886 was lower. Figure 6.25 visualized that the average scores achieved with SVM-RBF and TextCNN models were 2.6% less than that of the model's highest score obtained in the main research. However, this difference was 2.2% for the RoBERTa model. In a comparison of maximums between the main and project-based studies, this difference dropped to a level of about 1.4%. Considering different projects have different requirements and technical staff in preparation of specifications, there is an absolute bias in specification language and the deficiencies. However, this investigation proved that well-performed ML models developed in this research generated reliable and substantial results in the classification of defects for new projects too.

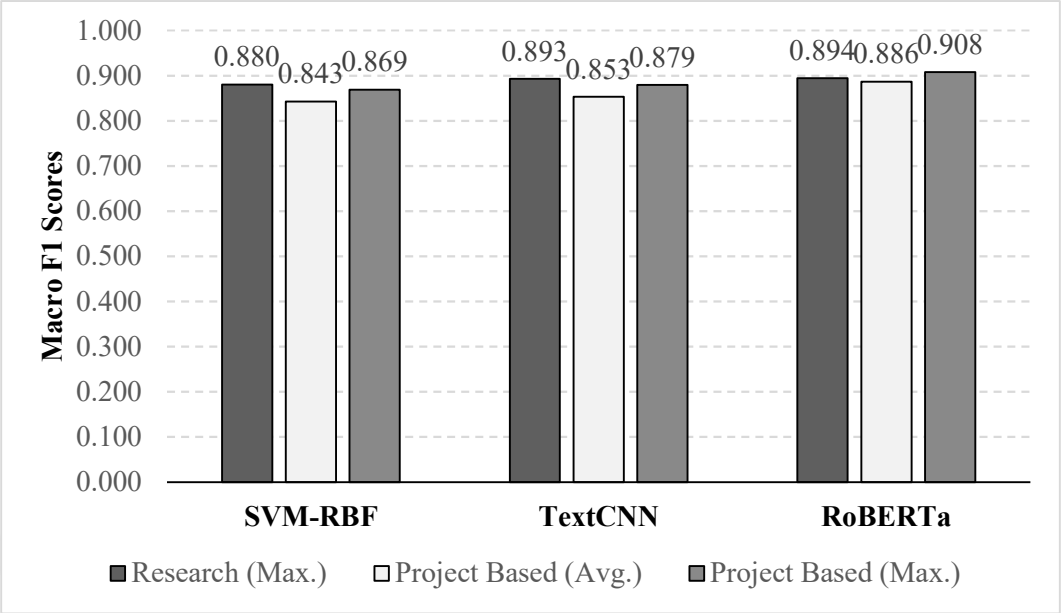


Figure 6.25. Comparison of Research and Project-Based Results

6.5 Test of GPT in Defect Determination

Advancements in computer science and internet technology have enabled access to vast amounts of textual data, driving the development of more sophisticated and capable deep learning (DL) models, such as Large Language Models (LLMs). LLMs are a specialized type of DL model trained on extensive textual datasets using a self-supervised learning approach, with parameters ranging from several hundred million to billions (Raiaan et al., 2024). Introduction of transformer architecture (Vaswani et al., 2017) into DL has revolutionized the foundation of LLMs. Its core features, including the self-attention mechanism and the ability to process sequences in parallel, have significantly advanced NLP methods such as text analysis, translation, sentiment analysis, question answering, and particularly text generation.

The growth of applications based on LLMs has been increasing exponentially, driven by larger and more dynamic data sources, such as user feedback, higher parameter counts, and more finely tuned subject-specific algorithms. LLaMa (Large Language Model Meta AI) developed by Meta (formerly Facebook), PaLM (Pathways Language Model) produced by Google AI, and GPT (Generative Pre-trained

Transformer) developed by OpenAI are a couple of highly advanced LLMs of today as generative AI (Ghimire et al., 2024). Although developing advanced LLMs is a relatively new area in NLP, various studies have explored the implementation of generative AI in the construction industry. These studies highlight the enhanced potential of LLMs to handle large, unstructured documents compared to conventional methods efficiently.

J. Zhang (2023) compared the conversion of building code requirements into a computable format for automatic compliance checking between rule-based models and ChatGPT-3. One more study conducted by Prieto et al. (2023) to generate the schedule for a simple construction by stating the technical descriptions of a simple construction in ChatGPT-3.5. Similarly, You et al. (2023) examined the advanced reasoning capability of ChatGPT for sequence planning of robotic-based assembly in construction projects. Accident prediction is another subject studied with generative pre-trained transformers. Yoo et al. (2024) developed fine-tuned ChatGPT-2/3 models to forecast six accident types within the 15,000 unstructured actual accident reports and compared them with shallow and DL models. J. Lee et al. (2024) conducted comparative research on the performance of two advanced LLMs, LLaMA2 and ChatGPT-3. Both models were fine-tuned with a training dataset comprising technical specifications regarding pressure vessels and tested through a question-and-answer evaluation to assess their accuracy, comprehension, and ability to extract relevant information from the provided documents.

Two LLMs, BERT and RoBERTa, were utilized in the course of this study as described, and the results were presented in previous sections. The implemented pre-trained BERT base model consists of 12 layers and 110 million parameters, while the pre-trained RoBERTa base model similarly features 12 layers but with 125 million parameters (Chollet & others, 2015). ChatGPT-4, released in 2023 by OpenAI, features 120 layers and an estimated 1.8 trillion parameters (Howarth, 2024; Raiaan et al., 2024). The context length, the maximum number of tokens the model can process concurrently, is 512 for BERT and RoBERTa whereas it is 32,768 for ChatGPT-4.

Although it was not in the main scope of this research, an experimental study was performed to observe the performance of GPT in identifying and categorizing the defects in construction specifications and comparing it with the efficiency of the previously implemented models. The experimentation was conducted with ChatGPT-4o, which was released by Open AI in two different variants in May 2024.

The same data set used in the validation stage of the research was also applied in testing with ChatGPT-4o, as shown in Table 4.8. The dataset comprises 760 statements from 6 different construction specifications with a total of 173 labeled defects.

1. Testing:

The first test was conducted using a completely unsupervised approach. A prompt text was prepared, including descriptions of the defect categories and their representative terms, such as: " *Category-1: The statements that have ambiguity, vagueness or language error.*" Additionally, basic rules were provided to guide the identification and review process, such as: " *Each statement will be reviewed based on the above defect categories*" and "*Based on above explanations, classify the statements of below specification and perform the review. List the statements that have defects and their categories*".

Initially, the prepared prompt text and the text of six specifications were entered separately into GPT within the same chat window. This trial was repeated twice in different chat sessions. The responses were exported to Excel, and the performance metrics were manually calculated based on the results as indicated in Table 6.12.

Both trials yielded consistent results, with macro F1 scores of 0.194 and 0.198, respectively, in categorizing defects within the 760 statements. The GPT model failed to predict any deficiencies in Group-2 and Group-4. Consequently, a value of 0.1 was manually assigned as True Positive (TP) to avoid division by zero. Since 587 statements were classified as having no determined deficiencies (Group-0), the distribution of classifications was skewed towards Group-0. The individual F1 score

of 0.830 for Group-0, though high, does not accurately reflect the overall performance of the model.

Table 6.12 Results of 1. Test with ChatGPT

Cat.	1. Trial			2. Trial		
	F1	**Precision**	**Recall**	**F1**	**Precision**	**Recall**
Gr-0	0.830	0.774	0.896	0.829	0.782	0.881
Gr-1	0.075	0.167	0.048	0.127	0.259	0.084
Gr-2	0.006	0.005	0.008	0.005	0.003	0.008
Gr-3	0.052	0.100	0.035	0.024	0.038	0.018
Gr-4	0.006	0.007	0.005	0.005	0.006	0.005
Macro F1	**0.194**			**0.198**		

Given ChatGPT's poor performance in categorizing defects, the results of the 2. trial were converted into a binary classification of "defect" and "no-defect" to evaluate how effectively the model could identify defects without assigning them to specific categories. Table 6.13 presents that the recognition of defects in the binary option is remarkably better than the categorization of five defects with the highest macro F1 score of 0.521.

Table 6.13 Results of 1. Test with ChatGPT as Binary Class

Cat.	2.Trial		
	F1	**Precision**	**Recall**
No Defect	0.829	0.782	0.881
Defect	0.213	0.293	0.168
Macro F1	**0.521**		

2. Testing:

The second test was performed by supervising the ChatGPT with the development of detailed prompt text. The prompt text used in the first test was further improved by adding the definitions of defect, such as "*Ambiguity: The statement that has multiple meanings or interpretation of the meaning changes according to reader.*", giving actual examples of defects for each category from construction specifications, such as "*Framing shall be installed as indicated.*", and semantic reasons why the

example statements can be considered as defects, such as "*Where? Which contract document is referred? Are those provided by owner or contractor developed?*".

In the second test, a different text entry sequence was applied. Each of the six specifications was submitted in a separate chat window following the prompt text. Initially, the prompt text was used to supervise the chat, and then each specification was individually provided for defect identification and categorization.

The second trial of the second test produced a higher macro F1 score of 0.256 than the first test, as indicated in Table 6.14. In the second trial, the model successfully predicted at least one true positive for the Group-4 category.

Table 6.14 Results of 2. Test with ChatGPT

	1. Trial			**2.Trial**		
Cat.	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**
Gr-0	0.808	0.775	0.843	0.858	0.810	0.913
Gr-1	0.113	0.171	0.084	0.153	0.257	0.108
Gr-2	0.053	0.040	0.077	0.148	0.143	0.154
Gr-3	0.022	0.029	0.018	0.067	0.094	0.053
Gr-4	0.005	0.005	0.005	0.054	0.059	0.050
Macro F1	**0.200**			**0.256**		

Like the first test, the results of the second test were also converted to a binary classification problem to see the defect identification performance of the second model as presented in Table 6.15. Although the macro F1 score of 0.603 is higher than the first test, the model still fails to predict most of the defects in the statements.

Table 6.15 Results of 2. Test with ChatGPT as Binary Class

	2.Trial		
Cat.	**F1**	**Precision**	**Recall**
No Defect	0.858	0.810	0.913
Defect	0.347	0.480	0.272
Macro F1	**0.603**		

Observations:

Some issues were also observed throughout the test with ChatGPT. Although they are not frequent, it is worth noting these anomalies occur and may require further attention.

- Multiple labeling: Although the prompt stated that each statement should have only one defect category, multiple categorizations of the same statement were observed, and neither of the assigned labels was correct.
- Rephrasing or Shortening: The responses from ChatGPT typically involved first restating the original statement and then specifying the detected defect category. However, in a limited number of instances, the original statements were either rephrased or long sentences were shortened.
- Addition: In a few cases, new statements that were not part of the original or the prompt text were introduced in the responses.
- Wrong categorization: Although ChatGPT explicitly stated the reason for the defect, the statement was still assigned to the wrong category. For example, the defect was identified as vagueness, yet the statement was placed in Group-3 instead of Group-1.
- Inconsistent categorization: Similar sentences of different specifications were classified in different defect categories.

One of the remarkable consequences of this preliminary investigation is that the individual precision score of 0.480 for the defect class indicated in Table 6.15 implies that almost half of the statements predicted as defective do not actually contain any defects. Additionally, the 0.272 recall score indicates that nearly 75% of the defects were not recognized by the ChatGPT model. This experimental study implies that without fine-tuning the GPT model with domain-specific knowledge, having a large database and advanced reasoning capabilities are still not sufficient to accurately identify and categorize construction specification deficiencies, even supervising and directing the model with definitions and examples.

CHAPTER 7

DISCUSSION

7.1 Major Findings and Comparison with Existing Research

The individual performance of each model developed in this research, along with their comparative evaluation, was presented in the previous chapter. Subsequently, a verification study was conducted by training models exclusively on specifications from a group of construction projects and testing them on specifications from other projects. Finally, an experimental investigation was carried out using ChatGPT-4o. A comprehensive assessment is presented in this chapter, which also covers a comparison with previously published related studies and an in-depth outcome analysis of the preeminent performing ML model.

Table 7.1 summarizes the evaluation metrics of the highest scored combination among each implemented algorithm sets in main research scope. The table is ranked according to the F1 scores. The study confirmed that pre-trained transformer-based ML models have the highest performance in identification and categorization of deficiencies within the statements of construction specifications.

With a defect classification F1 score of 0.912, the RoBERTa model outperformed all other implemented models. Following closely, the BERT model ranked in second with an F1 score of 0.903. Since RoBERTa was developed on the basis of BERT, with the argument of superior performance due to a larger dataset, longer training time, and dynamic masking (Y. Liu et al., 2019), the 1% higher F1 score demonstrates the effectiveness of these optimizations in enhancing the model's ability to generalize and perform better across tasks. The dominance of RoBERTa and BERT stems from their pre-training on massive datasets, their ability to understand complex relationships through the multi-head attention mechanism in the transformer architecture, and the use of dynamic word embeddings during fine-

tuning. Gao et al. (2024) also reached a similar conclusion in their research on the identification of risk-related sentences within news articles about the construction industry. Among the models tested, RoBERTa and BERT performed the best, achieving F1 scores of 0.851 and 0.820, respectively, outperforming other models such as CNN, LSTM, and LR.

Although Google's Word2Vec (W2V) model provides static semantic word embedding, unlike the pre-trained RoBERTa and BERT models, expanding the W2V vectorization with domain-specific text enabled better feature extraction from construction specifications. This improvement positioned the TextCNN model in third place, with a performance slightly lower than that of BERT. A similar outcome was reported by Zheng et al. (2024) in classifying building codes based on their machine interpretability. They achieved a weighted F1 score of 0.880 with BERT which was marginally higher than that of TextCNN, which scored 0.863.

Table 7.1 Highest Evaluation Metrics Achieved from Implemented Models

Model #	Model	T.R.	Type	Test Set	F1	Precision	Recall	Accuracy
92	RoBERTa	RoBERTa Emb.	10 Epochs	20%	0.912	0.916	0.911	0.977
86	BERT	BERT Emb.	10 Epochs	20%	0.907	0.912	0.909	0.975
82	TextCNN	Exp. Google W2V		10%	0.893	0.908	0.885	0.970
52	SVM-RBF	TF-IDF	Mixed	10%	0.880	0.935	0.834	0.972
25	LSVM	BoW	Unigram	10%	0.866	0.895	0.841	0.967
67	LR	TF-IDF	Unigram	10%	0.863	0.898	0.832	0.968
16	kNN	BoW	Mixed	10%	0.813	0.894	0.753	0.958
7	NB	TF-IDF	Unigram	10%	0.692	0.746	0.687	0.931

Support Vector Machine (SVM) algorithm with Radial Basis Function (RBF) kernel surpassed the other shallow ML models, reaching an F1 score of 0.880. SVM with

RBF is a powerful ML algorithm capable of handling non-linear boundaries for classification problems in higher-dimension spaces. Salama & El-Gohary (2016) succeeded to denote the ability of SVM with RBF kernel in classifying clauses from the general conditions section of contracts for automatic compliance checking. Other studies, such as Hassan & Le (2021) on the determination of design-build contract requirements and Baker et al. (2020) on the identification of accident causes within the accident reports, have also demonstrated the strong performances of SVM algorithms, even when utilizing the linear kernels.

With an F1 score 3.5% lower than RoBERTa and 1.5% lower than the DL algorithm TextCNN, the performance of SVM-RBF can be considered substantial. It requires fewer computational resources (e.g., GPU acceleration and large RAM), is less time-consuming, and offers greater interpretability. For instance, the BERT and RoBERTa models were developed in a Google Colab environment having a computing engine backbend 83.5 GB RAM and 40 GB GPU, whereas all other model implementations were performed in a personal computer having a capacity of 16 GB RAM and 4 GB GPU. This represents a trade-off between performance requirements, computational budget, and acceptable error margins.

Except for the transformer models, all others achieved their best performance on the 10% dataset used for testing. This signifies that static word embeddings and shallow ML algorithms require larger training datasets to perform effectively, unlike pre-trained transformer models, which can achieve high performance with smaller amounts of task-specific training data due to their extensive pretraining.

It was confirmed why the macro F1 score is a better evaluation metric than accuracy for the classification of a skewed data set as shown in Table 7.1. Although the difference between the F1 scores of the best and worst models is 22%, this difference in accuracy is only 4.6%. The sensitivity of the accuracy is very low as compared to macro F1 score. For instance, the difference between the accuracies of Models 92 and 52 is only 0.5%, while the F1 is approximately 3.2%. Similarly, the accuracy of Model 52 is higher than the one of Model 82 whereas the F1 score is lower.

Therefore, it was confirmed that the macro F1 score, rather than the weighted F1 score, was better for evaluating and interpreting performance metrics, as it provides a more balanced assessment than accuracy.

The results of implemented models in the main research, project-based verification, and the experimental study with ChatGPT are summarized in Table 7.2. As previously defined, the project-based study was conducted by constructing a training set from all specifications of randomly selected projects and a testing set from all specifications of the remaining projects. The primary aim of this verification was to evaluate the capability of identifying and classifying defects in scenarios where bias was introduced by different teams working on other construction projects. Despite the non-uniform distribution of defects between the training and testing datasets, the RoBERTa model delivered consistently strong results, achieving a maximum F1 score of 0.908 and an average F1 score of 0.886. The verification study denoted that the trained model can be implemented on other construction projects, achieving high efficiency.

Table 7.2 F1 Scores in Main Research, Project-based, and GPT

Model	F1 Score	Remarks
RoBERTa (Main research)	0.894	Maximum in 10% Test
	0.912	Research Maximum
RoBERTa (Project-Based)	0.886	Average (12% Test)
	0.908	Maximum (12% Test)
ChatGPT-4o	0.256	5-class classification
	0.603	Binary classification

However, ChatGPT's performance, even in the converted binary classification case, was remarkably poor despite its dynamic learning capabilities, advanced reasoning and context understanding, and extensive processing parameters. This suggests limitations in its ability to handle domain-specific tasks effectively without substantial fine-tuning or specialized training. It is still necessary to fine-tune ChatGPT with a labeled dataset that incorporates domain-specific terminology,

context, and classification criteria. This approach can significantly enhance the model's ability to learn and perform targeted tasks more effectively.

7.2 Error Analysis

So far, the performances of the implemented ML models have been evaluated based on the levels and comparisons of their F1 scores. While these assessments provide an overall view of effectiveness, it is equally vital to conduct a detailed analysis of the models' errors. Such an analysis leads to identifying specific patterns or types of misclassifications, revealing potential weaknesses in the models, and providing insights into how these limitations can be addressed.

The error analysis was carried out using two different approaches on the best-performing Model-92, RoBERTa shown in Table 7.1. The first assessment treated the problem as a binary classification task, evaluating whether the model could identify the presence of any defect in the statements of the construction specifications or not. A confusion matrix (Table 7.3) was prepared showing also precision and recall scores by transforming the five-class classification results of the Model-92 into a binary classification format.

Table 7.3 Binary Classification Confusion Matrix of Model-92

		Prediction			
		No Defect	**Defect**	**Ac. Total**	**Recall**
Actual	**No Defect**	2756	33	*2789*	98.8%
	Defect	37	288	*325*	88.6%
	Pr. Total	*2793*	*321*		
	Precision	98.7%	89.7%		

Table 7.3 describes that actually 2,789 statements in the test set do not have any defect, while 325 statements encompass any type of defect. The model predicted a total of 2793 statements as free of defects, whereas 321 statements were predicted to

contain one of the four defect categories defined in the research. The RoBERTa model successfully guessed 288 out of 325 defects, succeeding in a percentage of almost 89%. However, 33 statements predicted as defective were not actually defective, resulting in an error of about 10% in the precision of defect identification. Additionally, the model failed to identify 37 defects, incorrectly classifying them as belonging to the no-defect category. While this constitutes only around 1% of the no-defect predictions, it accounts for approximately 11% of the actual defects in the test set.

The second approach covers preparing the original confusion matrix of the same model as a five-class classification problem, as indicated in Table 7.4 which also presents the model's categorization performance in detail. The lowest categorization performance was revealed in Group-2 defects, where the model could precisely detect 31 defects out of 36, giving a recall score of approximately 86%. While the guess for Group-2 defects was superior, there was only 1 misprediction among 31, which gave a precision score of 97%. It is interpreted that a smaller number of estimates is enough to predict 86% of the defects.

Table 7.4 Five Class Confusion Matrix of Model-92

		Prediction						
		Gr-0	**Gr-1**	**Gr-2**	**Gr-3**	**Gr-4**	**Ac. Total**	**Recall**
Actual	**Gr-0**	2756	20	1	7	5	*2789*	98.8%
	Gr-1	20	135	0	0	1	*156*	86.5%
	Gr-2	4	0	31	1	0	*36*	86.1%
	Gr-3	12	1	0	87	0	*100*	87.0%
	Gr-4	1	0	0	0	32	*33*	97.0%
	Pr. Total	*2793*	*156*	*32*	*95*	*38*		
	Precision	98.7%	86.5%	96.9%	91.6%	84.2%		

In contrast, the model achieved its highest performance in Group-4, correctly predicting 32 out of 33 defects, equivalent to 97%. From the precision point of view,

6 out of the 38 predictions for Group-4 defects were incorrect and were not actually in that category, resulting in the lowest precision score of 84%. Although the detection performance for Group-4 defects was high, the precision in identifying the correct defect category was relatively low.

From a numerical perspective, the highest missed defect prediction was noticed in Group-1 defects, 21 out of 156 belonging to Group-1 were not able to be spotted. Besides, 20 statements labeled as having Group-1 defects did not have any type of defect.

Figure 7.1 illustrates the overall efficiency of the best-performing ML model in this research, represented by defect-wise F1 scores. The model functioned skilled in capturing Group-2 defects, which include conflicts and inconsistencies but presented a weaker performance in detecting Group-1 defects, which consist of ambiguity, vagueness, and language errors. The lower performance in identifying Group-1 defects is attributed to the significant diversity of sources, which contributes to ambiguity and vagueness as compared to other defect categories. Although, the F1 scores are ranked according to decreasing number of deficiencies in dataset, it is observed that there is not any significant relationship between the number of deficiencies and the model's precision or recall from the above table.

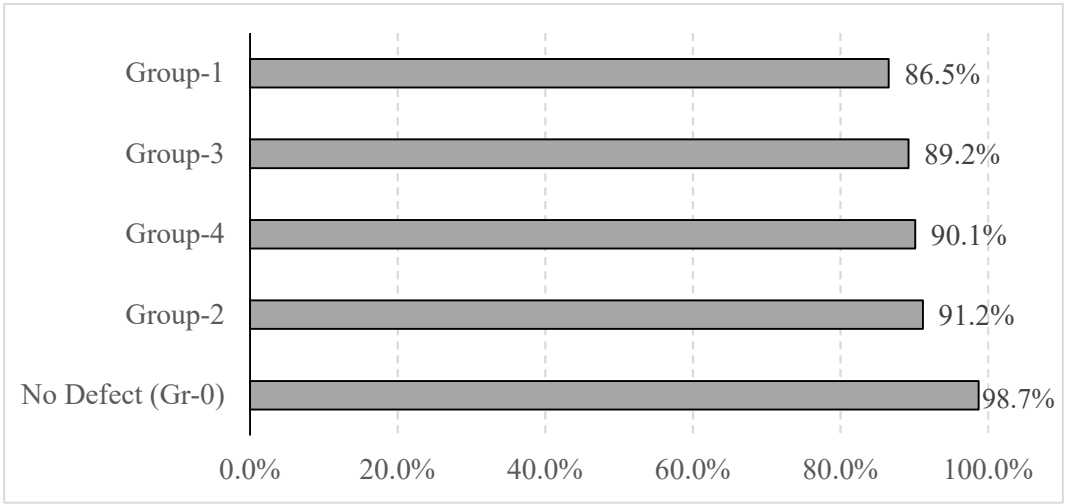


Figure 7.1. F1 Scores of RoBERTa Model-92

Finally, 73 statements out of 3,114, as shown in Table 7.4, were contrarily classified by the NLP model compared to the researcher, resulting in a discrepancy rate of 2.3%. This discrepancy highlights the challenges of aligning NLP models for complicated tasks like defect classification. The differences may result from the model's difficulty in understanding hidden semantics or complex criteria. Improving the model with more domain-specific training, better datasets, or combining rule-based and machine learning approaches could help to reduce errors and improve reliability.

CHAPTER 8

CONCLUSION

8.1 Overview of the Study

The root causes and consequences of the deficiencies embodied in construction specifications are well documented in the literature. However, the construction industry continues to struggle with the removal and detection of those defects and is obligated to solve the recognized problems belatedly. This delay, or sometimes omission, results in technical failures in construction, cost overruns, schedule disruptions, and disputes between project stakeholders that might end in court.

This research focused on identifying and categorizing defects in construction specifications using text classification techniques of Natural Language Processing (NLP). The study aimed to develop and implement supervised Machine Learning (ML) algorithms on unstructured documents to recognize defect patterns and classify them into predefined categories, enabling early detection and systematic analysis of construction specification deficiencies.

The study was founded on a literature review exploring the reasons and outcomes of specification deficiencies initially, then carried on the implementation of rule-based and supervised NLP techniques in construction-related subjects. The review designated that the closest domain-specific NLP studies were carried out on risk identification and assessment in main contract clauses as well as the information retrieval and extraction from technical documents like codes and regulations. However, there are minor studies on construction specifications, and none of them address deficiencies.

Based on the information gathered from the literature review, the research methodology was designed to develop the NLP pipeline that best fits with the aim of

the study. This covered the development of a defect categorization framework, document labeling and validation, text preprocessing methods, text representation techniques, and the selection of suitable ML algorithms. Various models from different types of ML algorithms were developed and implemented in 96 unique combinations. In the beginning, the internal performances of the ML models were evaluated using different test sets and text representation combinations. Successively, the algorithms were compared to identify the most effective model for defect detection and classification. The final steps involved error analysis, project-based verification, and an experimental study with Generative Pre-Trained Transformer (GPT) to ensure the strength and reliability of the best-performing model.

8.2 Key Outcomes and Contributions

The research highlighted that the transformer-based RoBERTa model outperformed all other implemented models, including BERT, not only in detecting defects in construction specifications but also in categorizing them. With evaluation metrics of approximately 92% precision, 91% recall, and 98% accuracy, the model demonstrates significant potential for real-world applications.

The appreciated performance of the same pre-trained RoBERTa model in a project-based study confirms that the ML models do not necessarily need to be trained exclusively with custom data. Instead, construction specifications developed or modified by different experts can also produce satisfactory results in defect detection.

The performance of SVM models in defect classification problems is appraised to be outstanding although it is worse than RoBERTa, which puts it as a viable option in situations where lower computational resources can be supplied.

Supervision of GPT by prompt text containing rules, definitions, and examples is not enough to reach the success level of ML algorithms, even the shallow ones, by

training with the labeled data. It is apparent that domain knowledge is still required to benefit from GPT's capabilities.

The labeling strategy in this research contributes a methodological framework to other researchers for defect recognition. The strategy creates a systematic way which can be also utilized to find out the deficiencies in specifications of business areas other than construction.

This study demonstrates the efficiency of NLP tools in defect recognition in construction specifications, contributing practically the industry stakeholders to benefit from it in a faster and more accurate way. Designers and specification writers can utilize such an NLP approach for specification quality control and design improvement, while preparing contract documents in the design stage before delivering to clients. Owners, designers and consultants can foresee the defects in specifications before issuing, thus giving them a chance to mitigate from potential disputes and claims. There is high evidence on early detection of these defects with developed NLP method that support construction contractors to proactively anticipate project risks caused by construction specifications. Accordingly, this improves also the project planning efficiency in the bidding stage and reducing the disputes by addressing defects at an early stage, minimizing their impact on project timelines and budgets. Construction professionals can supplement historical data for training the NLP models, increasing the tool's effectiveness over time which could separate technical defect identification from the experience of the staff and gain time savings for other critical subjects of project management.

8.3 Limitations and Recommendations

Limitations:

The creation of the dataset by a single researcher may be viewed as a limitation due to the potential subjectivity in defect labeling. However, as demonstrated in Section

4.4 Labeling Validation, independent industry experts validated the dataset, confirming that it does not contain any bias.

The research dataset was developed from only architectural specifications under 21 headings. Specifications from other disciplines in the construction industry, such as structural, civil, electrical, mechanical, or fire engineering, were not included in the study.

Some of the detected defects in literature, such as outdated information, applicability, and out-of-scope, were not included in the classification problem due to the impossibility of detecting those defects using the specifications as the only dataset.

The defect identifications and classifications were conducted sentence-wise. NLP models were trained and tested at the single sentence level, which did not consider the previous and next sentences.

Only supervised ML models were developed throughout the study. A rule-based approach was not utilized, which may produce remarkable performance but is generally preferred in cases of a low dataset.

Recommendations:

Support of ML models in classification with other NLP techniques may foster defect identification performance. Techniques like Named Entity Recognition (NER) can be utilized to identify entities in statements before defect recognition.

Electrical and mechanical installations have a substantial number of requirements, standards, and details, which regularly result in larger page volumes of specifications than architectural works. The crowding of text is one reason for poor specification development; therefore, other construction-related disciplines may be worth studying in defect detection.

Other contractual documents, such as agreements and general or special conditions, can be incorporated with construction specifications to recognize the defects under

the heading of “out of scope”. Out of scope is a critical defect that determines the relation between the scope of the works and the contents of the specification. Irrelevant content within a specification has a high potential to lead to disputes and claims, emphasizing the importance of addressing this type of defect.

For researchers who demand the advantage of the developed deficiency framework in their NLP implementation studies related to specifications, it is recommended to follow the NLP Pipeline route as indicated in Figure 3.1. The framework that categorizes the deficiencies and addresses them for identification stated in section 4.3.1 Labeling Framework, is the approach to generate train and test datasets of their own. Moreover, Table 7.1 is a supportive source of selecting the superior ML models with its most collaborative text representation technique and associated train-test set ratio.

For practical purposes, users can select RoBERTa or SVM-RBF based on their computational resources to recognize the deficiencies within their architectural works specifications. Trained RoBERTa model requires higher computational resources but performs superior. Also, dynamic learning capability of the model can handle OOV words that may provide flexibility in executing the RoBERTa model on architectural subjects other than the trained ones. Trained SVM-RBF model can efficiently run with a regular personal computer, requires considerably lower computational resources. The interpretability of the outcome is higher than DL based models. However, it is performance is poorer than RoBERTa and unable to compensate OOV words. The test specification document should be vectorized according to the corpus of the trained model to match the dimensionality of the train model in SVM-RBF. Therefore, the users should assess their conditions and choose the trained ML model that best fits their performance expectations.

Below presents typical steps to implement the trained ML model to identify and categorize deficiencies within architectural specifications in practice.

1. **Preprocessing:** Specification documents shall be cleaned and free from the noises. After sentence separation, the sentences shall be preprocessed.

2. **Feature Extraction:** Vectorization of preprocessed specification text shall be performed with TF-IDF Mixed n-gram approach for SVM-RBF model and pre-trained RoBERTa embedding for RoBERTa model.
3. **ML Model Implementation:** Previously trained ML models either SVM-RBF or RoBERTa shall be implemented on target text data for prediction of deficiencies.
4. **Results Tabulation:** Defect group predictions for each statement shall be tabulated in the form of number between zero to four. The created table may be transferred to Excel file for more user-friendly observation.

REFERENCES

Abrosimova, E. B., Mitrokhin, V. V., Orekhova, O. Y., Sedelnikova, I. M., & Yakusheva, A. M. (2022). Legal Definition of Defects in Construction-Related Areas. *Journal of Legal Affairs and Dispute Resolution in Engineering and Construction*, *14*(1). https://doi.org/10.1061/(ASCE)LA.1943-4170.0000513

Al Qady, M., & Kandil, A. (2015). Automatic classification of project documents on the basis of text content. *Journal of Computing in Civil Engineering*, *29*(3). https://doi.org/10.1061/(ASCE)CP.1943-5487.0000338

Ayodele, T., & Kajimo-Shakantu, K. (2021). *Data Sharing in the Construction Industry: An Assessment of Stakeholders' Perception*.

Babaeian Jelodar, M., Yiu, K. T. W., & Wilkinson, S. (2015). Dispute Manifestation and Relationship Quality in Practice. *Journal of Legal Affairs and Dispute Resolution in Engineering and Construction*, *8*. https://doi.org/10.1061/(ASCE)LA.1943-4170.0000171

Baek, S., Jung, W., & Han, S. H. (2021). A critical review of text-based research in construction: Data source, analysis method, and implications. *Automation in Construction*, *132*. https://doi.org/10.1016/J.AUTCON.2021.103915

Baker, H., Hallowell, M. R., & Tixier, A. J. P. (2020). Automatically learning construction injury precursors from text. *Automation in Construction*. https://doi.org/10.1016/j.autcon.2020.103145

Balush, I., Vysotska, V., & Albota, S. (2021). Recommendation System Development Based on Intelligent Search, NLP and Machine Learning Methods. *MoMLeT+ DS*, 584–617.

Bennett, W. B., & Blocker, M. J. (2016). Interpretation and Requirements of Contract Specifications. In P. Levin (Ed.), *Construction Contract Claims, Changes, and Dispute Resolution* (Third, pp. 91–116). ASCE.

Bilal, M., Oyedele, L. O., Qadir, J., Munir, K., Ajayi, S. O., Akinade, O. O., Owolabi, H. A., Alaka, H. A., & Pasha, M. (2016). Big Data in the construction industry: A review of present status, opportunities, and future trends. *Advanced Engineering Informatics*, *30*(3), 500–521. https://doi.org/10.1016/j.aei.2016.07.001

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media Inc. https://www.nltk.org/book/

Caldas, C. H., & Soibelman, L. (2003). Automating hierarchical document classification for construction management information systems. *Automation in Construction*, *12*(4), 395–406. https://doi.org/10.1016/S0926-5805(03)00004-9

Chollet, F., & others. (2015). *Keras*. https://keras.io

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, *20*, 37–46. https://doi.org/10.1177/001316446002000104?journalCode=epma

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*.

Ding, Y., Ma, J., & Luo, X. (2022). Applications of natural language processing in construction. *Automation in Construction*, *136*. https://doi.org/10.1016/J.AUTCON.2022.104169

Erdis, E., & Ozdemir, S. A. (2013). Analysis of technical specification-based disputes in construction industry. *KSCE Journal of Civil Engineering*, *17*(7), 1541–1550. https://doi.org/10.1007/s12205-013-0016-7

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological Bulletin*, *76*(5), 378–382. https://doi.org/10.1037/h0031619

Friel, A. M. (2022, January 5). *Construction data: what can be collected and how it can help*. https://www.pinsentmasons.com/out-law/analysis/construction-data-what-can-be-collected-and-how-it-can-help?utm_source=chatgpt.com

Gao, N., Touran, A., Wang, Q., & Beauchamp, N. (2024). Construction risk identification using a multi-sentence context-aware method. *Automation in Construction*, *164*. https://doi.org/10.1016/j.autcon.2024.105466

Ghazizadeh, E., & Zhu, P. (2021). A Systematic Literature Review of Natural Language Processing: Current State, Challenges and Risks. In S. and B. R. Arai Kohei and Kapoor (Ed.), *Proceedings of the Future Technologies Conference (FTC) 2020, Volume 1* (pp. 634–647). Springer International Publishing.

Ghimire, P., Kim, K., & Acharya, M. (2024). Opportunities and Challenges of Generative AI in Construction Industry: Focusing on Adoption of Text-Based Models. *Buildings*, *14*(1). https://doi.org/10.3390/buildings14010220

Graham, M., Milanowski, A., & Westat, J. (2014). *Measuring and Promoting Inter-Rater Agreement of Teacher and Principal Performance Ratings*.

Guo, K., Minchin, E., & Ferragut, T. (2005). The shift to warranties and performance specifications: What of method specifications? *Construction Management and Economics*, *23*(9), 953–963. https://doi.org/10.1080/01446190500183842

Haeb-Umbach, R., Watanabe, S., Nakatani, T., Bacchiani, M., Hoffmeister, B., Seltzer, M. L., Zen, H., & Souden, M. (2019). Speech Processing for Digital

Home Assistants: Combining signal processing with deep-learning techniques. *IEEE Signal Processing Magazine*, *36*(6), 111–124. https://doi.org/10.1109/MSP.2019.2918706

Hamdy, K., Abdelrasheed, I., Essawy, Y. A. S., & Gamal Eldeen, A. (2024). Automated Risk Analysis for Construction Contracts Using Neural Networks. *Journal of Legal Affairs and Dispute Resolution in Engineering and Construction*, *16*(4). https://doi.org/10.1061/JLADAH.LADR-1149

Hassan, F., & Le, T. (2020). Automated Requirements Identification from Construction Contract Documents Using Natural Language Processing. *Journal of Legal Affairs and Dispute Resolution in Engineering and Construction*, *12*(2). https://doi.org/10.1061/(ASCE)LA.1943-4170.0000379

Hassan, F., & Le, T. (2021). Computer-assisted separation of design-build contract requirements to support subcontract drafting. *Automation in Construction*, *122*. https://doi.org/10.1016/J.AUTCON.2020.103479

Honnibal, M., Montani, I., Van Landeghem, S., Boyd, A., & others. (2020). *spaCy: Industrial-strength natural language processing in python*.

Howarth, J. (2024, August 6). *Number of Parameters in GPT-4*. https://explodingtopics.com/blog/gpt-parameters

IBM Data and AI Team. (2023a, July 6). *AI vs. machine learning vs. deep learning vs. neural networks: What's the difference?* IBM. https://www.ibm.com/think/topics/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

IBM Data and AI Team. (2023b, November 3). *What is a machine learning algorithm?* IBM. https://www.ibm.com/topics/machine-learning-algorithms

Ibrahim, A. M., Hassanain, M. A., & Salaheldin, M. H. (2023). Assessment of specifications' usability for engineering projects of the built environment.

International Journal of Building Pathology and Adaptation. https://doi.org/10.1108/IJBPA-11-2022-0176

Institute, C. (2013). Preparation of Construction Specifications for Civil Projects. In *Preparation of Construction Specifications for Civil Projects*. https://doi.org/10.1061/9780784413043

Institute, T. C. S. (2004). *The Project Resource Manual (PRM) : CSI Manual of Practice, 5th Edition*. Mcgraw-hill.

Jackson, J. T. (1990). Technical specifications' effect on construction. *Journal of Construction Engineering and Management*, *116*(3), 463–467. https://doi.org/10.1061/(ASCE)0733-9364(1990)116:3(463)

Jagannathan, M., & Delhi, V. S. K. (2019). Litigation Proneness of Dispute Resolution Clauses in Construction Contracts. *Journal of Legal Affairs and Dispute Resolution in Engineering and Construction*, *11*(3). https://doi.org/10.1061/(ASCE)LA.1943-4170.0000301

Jagannathan, M., & Delhi, V. S. K. (2020). Litigation in Construction Contracts: Literature Review. *Journal of Legal Affairs and Dispute Resolution in Engineering and Construction*, *12*(1). https://doi.org/10.1061/(ASCE)LA.1943-4170.0000342

Jiang, K., & Lu, X. (2020). Natural Language Processing and Its Applications in Machine Translation: A Diachronic Review. *2020 IEEE 3rd International Conference of Safe Production and Informatization (IICSPI)*, 210–214. https://doi.org/10.1109/IICSPI51290.2020.9332458

Kalin, M., Weygant, R. S., Rosen, H. J., & Regener, J. R. (2011). *Construction specifications writing: Principles and procedures*. John Wiley & Sons.

Keserer, E. (2024, January 5). *The six main subsets of AI: (Machine learning, NLP, and more)*. Akkio. https://www.akkio.com/post/the-five-main-subsets-of-ai-machine-learning-nlp-and-more

Khalef, R., & El-Adaway, I. H. (2024). Investigating Causes of Disputes Resulting in Litigation in Airport Development Projects in the United States Using Graph-Based Techniques. *Journal of Management in Engineering*, *40*(3). https://doi.org/10.1061/JMENEA.MEENG-5903

Khan, N., Nadeau, S., Xuan-Tan, P., & Boton, C. (2024). Exploring associations between accident types and activities in construction using natural language processing. *Automation in Construction*, *164*. https://doi.org/10.1016/j.autcon.2024.105457

Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: state of the art, current trends and challenges. *Multimedia Tools and Applications*, *82*(3), 3713–3744. https://doi.org/10.1007/s11042-022-13428-4

Kim, T., & Chi, S. (2019). Accident Case Retrieval and Analyses: Using Natural Language Processing in the Construction Industry. *Journal of Construction Engineering and Management*, *145*(3). https://doi.org/10.1061/(ASCE)CO.1943-7862.0001625

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 1746–1751. https://doi.org/10.3115/v1/d14-1181

Koroteev, M. (2021). *BERT: A Review of Applications in Natural Language Processing and Understanding*. https://doi.org/10.48550/arXiv.2103.11943

Kululanga, G. K., & Price, A. D. (2005). Measuring Quality of Writing of Construction Specifications. *Journal of Construction Engineering and Management*, *131*(8), 859–865. https://doi.org/10.1061/(asce)0733-9364(2005)131:8(859)

Lahiri, D., Kandimalla, P. C. P., & Jeysekar, A. (2023). Hybrid Multi Purpose Voice Assistant. *2023 2nd International Conference on Applied Artificial*

Intelligence and Computing (ICAAIC), 816–822. https://doi.org/10.1109/ICAAIC56838.2023.10140365

Lam, P. T. I., Kumaraswamy, M. M., & Ng, S. T. (2004). The use of construction specifications in Singapore. *Construction Management and Economics*, *22*(10), 1067–1079. https://doi.org/10.1080/0144619042000213265

Landis, J., & Koch, G. (1977). The Measurement Of Observer Agreement For Categorical Data. *Biometrics*, *33*, 159–174. https://doi.org/10.2307/2529310

Lang, N. (2022, January 26). *Reinforcement Learning - simply explained!* Data Base Camp. https://databasecamp.de/en/ml/reinforcement-learnings

Lee, G., Moon, S., & Chi, S. (2022). Reference section identification of construction specifications by a deep structured semantic model. *Engineering, Construction and Architectural Management*. https://doi.org/10.1108/ECAM-10-2021-0920

Lee, J., Jung, W., & Baek, S. (2024). In-House Knowledge Management Using a Large Language Model: Focusing on Technical Specification Documents Review. *Applied Sciences (Switzerland)*, *14*(5). https://doi.org/10.3390/app14052096

Lee, J., Yi, J.-S., & Son, J. (2019). Development of Automatic-Extraction Model of Poisonous Clauses in International Construction Contracts Using Rule-Based NLP. *Journal of Computing in Civil Engineering*, *33*(3). https://doi.org/10.1061/(ASCE)CP.1943-5487.0000807

Liu, K., & El-Gohary, N. (2017). Ontology-based semi-supervised conditional random fields for automated information extraction from bridge inspection reports. *Automation in Construction*, *81*, 313–327. https://doi.org/10.1016/j.autcon.2017.02.003

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. https://arxiv.org/abs/1907.11692v1

McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, 56–61. https://doi.org/10.25080/MAJORA-92BF1922-00A

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*. https://arxiv.org/abs/1301.3781v3

Moon, S., Lee, G., Chi, S., & Oh, H. (2021). Automated Construction Specification Review with Named Entity Recognition Using Natural Language Processing. *Journal of Construction Engineering and Management*, *147*(1). https://doi.org/10.1061/(ASCE)CO.1943-7862.0001953

Nailman, A. (2024, May 27). *Rule-based vs. Machine Learning for NLP: Which Approach Is Superior?* https://machinelearningmodels.org/rule-based-vs-machine-learning-for-nlp-which-approach-is-superior/?utm_source=chatgpt.com

Otten, N. Van. (2024, November 1). *Text Representation: A Simple Explanation Of Complex Techniques*. https://spotintelligence.com/2024/10/01/text-representation-a-simple-explanation-of-complex-techniques/

Pan, Y., & Zhang, L. (2021). Roles of artificial intelligence in construction engineering and management: A critical review and future trends. In *Automation in Construction* (Vol. 122). Elsevier B.V. https://doi.org/10.1016/j.autcon.2020.103517

Pascucci, F., Savelli, E., & Gistri, G. (2023). How digital technologies reshape marketing: evidence from a qualitative investigation. *Italian Journal of*

Marketing, *2023*(1), 27–58. https://doi.org/10.1007/S43039-023-00063-6/TABLES/3

Patil, R., Boit, S., Gudivada, V., & Nandigam, J. (2023). A Survey of Text Representation and Embedding Techniques in NLP. *IEEE Access*, *PP*, 1. https://doi.org/10.1109/ACCESS.2023.3266377

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*(85), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

Prieto, S. A., Mengiste, E. T., & García de Soto, B. (2023). Investigating the Use of ChatGPT for the Scheduling of Construction Projects. *Buildings*, *13*(4). https://doi.org/10.3390/buildings13040857

Raiaan, M. A. K., Mukta, M. S. H., Fatema, K., Fahad, N. M., Sakib, S., Mim, M. M. J., Ahmad, J., Ali, M. E., & Azam, S. (2024). A Review on Large Language Models: Architectures, Applications, Taxonomies, Open Issues and Challenges. *IEEE Access*, *12*, 26839–26874. https://doi.org/10.1109/ACCESS.2024.3365742

Ranathunga, S., Lee, E. S. A., Prifti Skenduli, M., Shekhar, R., Alam, M., & Kaur, R. (2023). Neural Machine Translation for Low-resource Languages: A Survey. *ACM Computing Surveys*, *55*(11). https://doi.org/10.1145/3567592

Řehuřek, R., & Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. *Proceedings of the LREC 2010 Workshop on New for NLP Frameworks*, 45–50.

Renz, A., & Solas, M. Z. (2016). *Shaping the Future of Construction: A Breakthrough in Mindset and Technology*. https://www3.weforum.org/docs/WEF_Shaping_the_Future_of_Construction_full_report__.pdf

Salama, D. M., & El-Gohary, N. M. (2016). Semantic Text Classification for Supporting Automated Compliance Checking in Construction. *Journal of Computing in Civil Engineering*, *30*(1). https://doi.org/10.1061/(ASCE)CP.1943-5487.0000301

Senouci, A., Alsarraj, A., Gunduz, M., & Eldin, N. (2017). Analysis of change orders in Qatari construction projects. *International Journal of Construction Management*, *17*(4), 280–292. https://doi.org/10.1080/15623599.2016.1211973

Shafi, P. M., Jawalkar, G. S., Kadam, M. A., Ambawale, R. R., & Bankar, S. V. (2020). AI—Assisted Chatbot for E-Commerce to Address Selection of Products from Multiple Products. *Studies in Systems, Decision and Control*, *266*, 57–80. https://doi.org/10.1007/978-3-030-39047-1_3/FIGURES/6

Shaikh, M. (2023, July 13). *Top 10 Programming Languages for AI and Natural Language Processing*. Insider Monkey. https://www.insidermonkey.com/blog/top-10-programming-languages-for-ai-and-natural-language-processing-1167654/

Solum, L. B. (2023, December 10). *Legal Theory Lexicon 051: Vagueness and Ambiguity*. Legal Theory Lexicon. https://lsolum.typepad.com/legal_theory_lexicon/2006/08/legal_theory_le.html

Standards Committee. (2013). Ch.19-Standard Contract Documents and Unique Special Provisions. In *Indiana Design Manual* (2021st ed.). Indiana Department of Transportation.

Stemler, S. (2004). A Comparison of Consensus, Consistency, and Measurement Approaches to Estimating Interrater Reliability. *Practical Assessment, Research, and Evaluation*, *9*, 1–19.

Thomas, H. R., Smith, G. R., & Wirsching, S. M. (1995). Understanding Defective Specifications. *Journal of Construction Engineering and Management*, *121*(1), 55–65. https://doi.org/10.1061/(ASCE)0733-9364(1995)121:1(55)

Tixier, A. J. P., Hallowell, M. R., Rajagopalan, B., & Bowman, D. (2016). Automated content analysis for construction safety: A natural language processing system to extract precursors and outcomes from unstructured injury reports. *Automation in Construction*. https://doi.org/10.1016/j.autcon.2015.11.001

Vajjala, S., Majumder, B., Gupta, A., & Surana, H. (2020). *Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems*. O'Reilly Media. https://books.google.co.uk/books?id=hvrrDwAAQBAJ

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, *2017-December*, 5999–6009. https://arxiv.org/abs/1706.03762v7

Verma, P., Zhong, S., Liu, X., & Rajan, A. (2023). Seasonality Based Reranking of E-commerce Autocomplete Using Natural Language Queries. *The 6th Workshop on E-Commerce and NLP (ECNLP 6)*. https://arxiv.org/abs/2308.02055

Vocabulary.com. (2024, December). *Conflict & Inconsistency*. https://www.vocabulary.com/dictionary/inconsistency

Wankhade, M., Rao, A. C. S., & Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review*, *55*(7), 5731–5780. https://doi.org/10.1007/s10462-022-10144-1

Weber, P., & Ludwig, T. (2020). (Non-) Interacting with conversational agents: perceptions and motivations of using chatbots and voice assistants. In *Proceedings of Mensch und Computer 2020* (pp. 321–331).

Wong, S., Zheng, C., Su, X., & Tang, Y. (2024). Construction contract risk identification based on knowledge-augmented language models. *Computers in Industry*, *157–158*. https://doi.org/10.1016/j.compind.2024.104082

Wu, C., Li, X., Guo, Y., Wang, J., Ren, Z., Wang, M., & Yang, Z. (2022). Natural language processing for smart construction: Current status and future directions. *Automation in Construction*, *134*. https://doi.org/10.1016/J.AUTCON.2021.104059

Yoo, B., Kim, J., Park, S., Ahn, C. R., & Oh, T. (2024). Harnessing Generative Pre-Trained Transformers for Construction Accident Prediction with Saliency Visualization. *Applied Sciences (Switzerland)*, *14*(2). https://doi.org/10.3390/app14020664

You, H., Ye, Y., Zhou, T., Zhu, Q., & Du, J. (2023). Robot-Enabled Construction Assembly with Automated Sequence Planning Based on ChatGPT: RoboGPT. *Buildings*, *13*(7). https://doi.org/10.3390/buildings13071772

Zaneldin, E. K. (2020). Investigating the types, causes and severity of claims in construction projects in the UAE. *International Journal of Construction Management*, *20*(5), 385–401. https://doi.org/10.1080/15623599.2018.1484863

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press.

Zhang, J. (2023). How Can ChatGPT Help in Automated Building Code Compliance Checking? *Proceedings of the International Symposium on Automation and Robotics in Construction*, 63–70. https://doi.org/10.22260/ISARC2023/0011

Zhang, J., & El-Gohary, N. M. (2016). Semantic NLP-Based Information Extraction from Construction Regulatory Documents for Automated Compliance Checking. *Journal of Computing in Civil Engineering*, *30*(2). https://doi.org/10.1061/(ASCE)CP.1943-5487.0000346

Zhang, Y., & Wallace, B. C. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. *CoRR*, *abs/1510.03820*. http://arxiv.org/abs/1510.03820

Zheng, Z., Zhou, Y.-C., Chen, K.-Y., Lu, X.-Z., She, Z.-T., & Lin, J.-R. (2024). A text classification-based approach for evaluating and enhancing the machine interpretability of building codes. *Engineering Applications of Artificial Intelligence*, *127*. https://doi.org/10.1016/j.engappai.2023.107207

Zhong, B., Shen, L., Pan, X., Zhong, X., & He, W. (2024). Dispute Classification and Analysis: Deep Learning-Based Text Mining for Construction Contract Management. *Journal of Construction Engineering and Management*, *150*(1). https://doi.org/10.1061/JCEMD4.COENG-14080

Zhong, B., Xing, X., Luo, H., Zhou, Q., Li, H., Rose, T., & Fang, W. (2020). Deep learning-based extraction of construction procedural constraints from construction regulations. *Advanced Engineering Informatics*. https://doi.org/10.1016/j.aei.2019.101003

Zou, Y., Kiviniemi, A., & Jones, S. W. (2017). Retrieving similar cases for construction project risk management using Natural Language Processing techniques. *Automation in Construction*, *80*, 66–76. https://doi.org/10.1016/j.autcon.2017.04.003

CURRICULUM VITAE

Surname, Name: Madenli, Özgür

EDUCATION

Degree	Institution	Year of Graduation
MS	METU Civil Engineering	2002
BS	METU Civil Engineering	1999
High School	Özel Ar Fen Lisesi, Samsun	1994

FOREIGN LANGUAGES

Advanced English, Beginner Russian

PUBLICATIONS

1. Madenli Ö., Kibar Arif A. “Yapılarda Ferahlık: Ardgermeli Beton”, AnbaRapor (03) 90-91 (2010)

2. Karaesmen E., Erkay C., Aksoylar C., Kibar A., Madenli Ö. “17-Ağustos-1999 Marmara Depremine Toplu Bakış”, TMH Dergi (403), 24-36 (1999)

3. Karaesmen E., Erkay C., Aksoylar C., Kibar A., Madenli Ö. “17-Ağustos-1999 Güney Marmara (Kocaeli) Depremi İlk Gözlemlere Dayalı Ön Değerlendirmeler”, İMO Ankara Şubesi Bülteni, (1999)