EXTRACTION AND CLASSIFICATION OF INDIRECT COST-RELATED CLAUSES FROM CONSTRUCTION CONSULTANCY TENDERS WITH NATURAL LANGUAGE PROCESSING

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

BY

ERAY İMAMOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CIVIL ENGINEERING

SEPTEMBER 2024

Approval of the thesis:

EXTRACTION AND CLASSIFICATION OF INDIRECT COST-RELATED CLAUSES FROM CONSTRUCTION CONSULTANCY TENDERS WITH NATURAL LANGUAGE PROCESSING

submitted by **ERAY İMAMOĞLU** in partial fulfillment of the requirements for the degree of **Master of Science** 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** _____

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

Prof. Dr. İrem Dikmen Toker
Co-Supervisor, **Civil Engineering, University of Reading** _____

Examining Committee Members:

Prof. Dr. Rifat Sönmez
Civil Engineering, METU _____

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

Prof. Dr. İrem Dikmen Toker
Civil Engineering, University of Reading _____

Assoc. Prof. Dr. Hande Işık Öztürk
Civil Engineering, METU _____

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

Date: 05.09.2024

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 : Eray, İmamoğlu

Signature :

ABSTRACT

EXTRACTION AND CLASSIFICATION OF INDIRECT COST-RELATED CLAUSES FROM CONSTRUCTION CONSULTANCY TENDERS WITH NATURAL LANGUAGE PROCESSING

İmamoğlu, Eray
Master of Science, Civil Engineering
Supervisor: Assoc Prof. Dr. Güzide Atasoy Özcan
Co-Supervisor: Prof. Dr. İrem Dikmen Toker

September 2024, 106 pages

With the surge in demand for construction consultancy tenders, tender documents have grown in complexity, posing challenges in timely and accurate reviews. There is a significant need to help decision-makers, including tender engineers and sector professionals, identify cost-related clauses in tender documents so that precise, accurate, and comprehensive proposals can be developed. This research addresses the need to extract indirect cost-related clauses from construction consultancy tender documentation. This study uses Natural Language Processing (NLP) techniques to improve and accelerate the review process of identifying, extracting, and classifying indirect cost-related clauses from construction consultancy tenders to use them in their financial proposals. Hence, bidders can significantly reduce the time required for document analysis and double-checking if there is any missing clause.

The significance of the research is for the tender review process, enabling bidders to meet stringent client deadlines while maintaining high accuracy in extracting cost-

related clauses. Furthermore, the results will also contribute to the advancement of NLP applications in the consultancy domain of the construction industry.

Keywords: Natural Language Processing, Construction Tenders, Consultancy, Indirect Cost, Detecting Cost Clauses

ÖZ

İNŞAAT MÜŞAVİRLİK İHALELERİNDEN DOLAYLI MALİYET İLE İLGİLİ MADDELERİN DOĞAL DİL İŞLEME İLE ÇIKARILMASI VE SINIFLANDIRILMASI

İmamoğlu, Eray
Yüksek Lisans, İnşaat Mühendisliği
Tez Yöneticisi: Doç. Dr. Güzide Atasoy Özcan
Ortak Tez Yöneticisi: Prof. Dr. İrem Dikmen Toker

Eylül 2024, 106 sayfa

İnşaat danışmanlık ihalelerine olan talebin artmasıyla birlikte, ihale belgeleri karmaşıklaşarak zamanında ve doğru incelemeler yapılmasında zorluklar yaratmaktadır. İhale mühendisleri ve sektör profesyonelleri de dahil olmak üzere karar vericilere, ihale belgelerinde maliyetle ilgili maddeleri belirleme konusunda yardımcı olma ihtiyacı oldukça büyüktür. Bu araştırma, inşaat danışmanlık ihale belgelerinden dolaylı maliyetlerle ilgili maddeleri çıkarmaya yönelik ihtiyacı ele almaktadır. Bu çalışma, inşaat danışmanlık ihalelerinden dolaylı maliyetlerle ilgili maddeleri belirlemek, çıkarmak ve sınıflandırmak için Doğal Dil İşleme (NLP) tekniklerini kullanarak inceleme sürecini iyileştirmeyi ve hızlandırmayı amaçlamaktadır. Böylece, teklif sahipleri, belge analizine harcanan zamanı önemli ölçüde azaltabilir ve eksik bir madde olup olmadığını iki kez kontrol edebilirler.

Araştırmanın önemi, teklif sahiplerinin sıkı müşteri teslim sürelerine uymalarını sağlarken, maliyetle ilgili maddeleri çıkarmada yüksek doğruluğu korumalarına yardımcı olan ihale inceleme sürecindedir. Ayrıca, sonuçlar, inşaat sektöründe danışmanlık alanında NLP uygulamalarının ilerlemesine de katkıda bulunacaktır.

Anahtar Kelimeler: Doğal Dil İşleme, İnşaat İhaleleri, Danışmanlık, Dolaylı Maliyet, Maliyet Maddelerinin Tespiti

Dedication to my family

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my thesis advisor Assoc. Prof. Dr. Güzide Atasoy Özcan and co-advisor Prof. Dr. İrem Dikmen Toker, for their invaluable guidance, encouragement, and support throughout this period.

I am thankful to my family, whose unhesitant support and love have been my greatest source of strength. To my beloved wife, thank you for your endless patience, understanding, and encouragement. To my mother and father, your sacrifices and unconditional love have shaped who I am today, and I am forever grateful. To my sister, thank you for always being there and for your steady support.

A special thanks to Dr. Hüseyin Erol, who introduced me to the fascinating world of natural language processing (NLP).

I would also like to extend my sincere thanks to Yüksel Proje A.Ş. for their support and contributions, which have been invaluable in the completion of this thesis.

Lastly, I am grateful to all my friends, colleagues, and everyone who has supported me in any way during this academic endeavor. Thank you all for being a part of the study.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xiv

LIST OF FIGURES xvi

LIST OF ABBREVIATIONS xvii

CHAPTERS

1 INTRODUCTION 1

2 LITERATURE REVIEW 7

2.1 Natural Language Processing 7

2.2 Natural Language Processing in Construction Industry 8

3 METHODOLOGY 15

3.1 Dataset Preparation 16

3.2 Text Preprocessing 16

3.3 Feature Extraction 17

3.4 Sentence Classification 17

3.5 Model Tuning and Hyperparameter Optimization 17

3.6 Classification Results 18

4 COST-RELATED CLAUSE SELECTION AND DATASET PREPARATION 19

4.1 Direct and Indirect Costs 19

4.1.1 Direct Costs 19

4.1.2 Indirect Costs 20
4.2 Dataset 26
4.2.1 Selection Techniques 26
4.2.2 Dataset Information 26
4.2.3 Labeling Techniques 27
5 CLASSIFICATION PROCESS AND MODEL PREPARATION 33
5.1 Preprocessing the Data 33
5.1.1 Data Sampling & Equalization of Data 34
5.1.2 Stopword Removal 35
5.1.3 Punctuation Removal 36
5.1.4 Lowercasing 36
5.1.5 Tokenization 37
5.1.6 Stemming and Lemmatization: 37
5.2 Text Representations and Word Embeddings 38
5.2.1 Bag-of-Words (BoW) 38
5.2.2 Term Frequency-Inverse Document Frequency (TF-IDF) 39
5.2.3 Word Embeddings 41
5.3 Machine Learning and Deep Learning Algorithms 47
5.3.1 Naïve Bayes (NB) 47
5.3.2 Support Vector Machine (SVM) 48
5.3.3 Logistic Regression (LR) 49
5.3.4 Random Forest (RF) 50
5.3.5 Convolutional Neural Network (CNN) 51
5.3.6 Recurrent Neural Network (RNN) 51

5.3.7 Long Short-Term Memory (LSTM) 52
5.3.8 Bi-Directional Long Short-Term Memory (bi-LSTM) 52
5.3.9 Bidirectional Encoder Representations from Transformers (BERT) 53
5.4 Hyper-parameter Optimization 56
6 CLASSIFICATION RESULTS 65
6.1 Evaluation Methods 65
6.2 Classification Results and Interpretation of the Results 70
6.2.1 Performance of Machine Learning Algorithms 70
6.2.2 Performance of Deep Learning Algorithms – Binary Classification 74
6.2.3 Performance of BERT-Binary Classification 78
6.2.4 Performance of BERT-Multilabel Classification 79
6.2.5 Analysis of Wrongly-Labeled Sentences: Binary 81
7 CONCLUSIONS 85
REFERENCES 91

LIST OF TABLES

TABLES

Table 4-1 Comparison of Indirect Cost Terminology 24
Table 4-2 Examples of Sentences from Dataset (cont'd) 24
Table 4-3 Dataset Information 27
Table 4-4 Number of Clauses in Dataset 28
Table 4-5 Example of Indirect Cost-Related Clause 28
Table 4-6 Example of Non-Cost Related Clause 29
Table 4-7 Example of Multilabeling (1 Label) 29
Table 4-8 Example of Multilabeling (>1 Label) 29
Table 4-9 Number of Sentences in Labels 30
Table 4-10 Multi-Label Combinations 30
Table 4-11 Demonstrative Examples of Additional User Requirements 31
Table 5-1 BERT Hyperparameters 54
Table 5-2 Classification Types and Model&Vectorizer Information 55
Table 5-3 Word2Vec & LSTM Hyperparameter Optimization Result 62
Table 5-4 Word2Vec & Bi-LSTM Hyperparameter Optimization Result 62
Table 5-5 Word2Vec & CNN Hyperparameter Optimization Result 63
Table 5-6 Word2Vec & RNN Hyperparameter Optimization Result 63
Table 5-7 GloVe & LSTM Hyperparameter Optimization Result 63
Table 5-8 GloVe & Bi-LSTM Hyperparameter Optimization Result 64
Table 5-9 GloVe & CNN Hyperparameter Optimization Result 64
Table 5-10 GloVe & RNN Hyperparameter Optimization Result 64
Table 6-1 Confusion Matrix 66
Table 6-2 Performance of Machine Learning Algorithms 72
Table 6-3 Performance of Algorithms with Word2Vec 75
Table 6-4 Performance of Algorithms with GloVe 76
Table 6-5 Performance Metrics of BERT 78

Table 6-6 Multi-Lable Classification Performance - BERT..... 80

LIST OF FIGURES

FIGURES

Figure 3-1 Methodology 15
Figure 4-1 Sentence Labeling Steps 28
Figure 5-1 Validation Loss vs Training Loss (Ying, 2019) 58
Figure 6-1 Multi-label Classification Metrics 68
Figure 6-2 Machine Learning Classification Steps 71
Figure 6-3 Accuracy of Different ML Algorithms using BoW and TF-IDF 73
Figure 6-4 Confusion Matrix - BERT Performance 79

LIST OF ABBREVIATIONS

ABBREVIATIONS	
ADB	Asian Development Bank
AFD	French Development Agency
AI	Artificial Intelligence
AIIB	Asian Infrastructure Investment Bank
BERT	Bidirectional Encoder Representations from Transformers
BIM	Building Information Modelling
Bi-LSTM	Bidirectional Long Short-Term Memory
BoW	Bag-of-Words
CNN	Convolutional Neural Networks
CRC	Cost-Related Clause
EBRD	European Bank for Reconstruction and Development
EIB	European Investment Bank
FIDIC	International Federation of Consulting Engineers
GCC	General Conditions of Contract
ITT	Instruction to Tenderers
LR	Logistic Regression
LSTM	Long Short-Term Memory
MDB	Multi Development Bank
ML	Machine Learning

NB	Naive Bayes
NER	Named Entity Recognition
NLP	Natural Language Processing
O&M	Operation and Maintenance
RF	Random Forest
RNN	Recurrent Neural Network
SCC	Specific Conditions of Contract
SVM	Support Vector Machine
TF-IDF	Term Frequency-Inverse Document Frequency
TOR	Terms of Reference
WB	World Bank
QA	Question Answer

CHAPTER 1

INTRODUCTION

The life cycle of construction projects and interaction between the consultants and clients begin with a call for tenders. The client of the project publishes a tender package comprising multiple documents, including technical and contractual dossiers, and the consultants have a strict deadline to submit technical and financial proposals. There are many documents among the tender packages, which consist of lots of clauses that impact the bidder's financial proposal. The construction consultancy tender generally consists of terms of reference with specifications presenting the general scope of work to the potential bidder. In addition to the Terms of Reference (ToR), it also includes GCC and SCC, which describe the contract conditions. Moreover, the tender package includes ITT and a data sheet that gives crucial information to potential bidders.

The tenderers must investigate the Cost-Related Clauses to prepare the financial proposal and calculate the overall cost. Generally, the financial proposal can be divided into two main concepts: direct and indirect costs. While direct costs indicate the materials, workmanship, and equipment directly related to the projects, such as material and equipment costs, indirect costs include administrative and personnel-related costs, such as licenses, travel, accommodation, food and beverage, and site office. Due to the nature of the consultancy construction tenders, the labor and expert costs to be deployed to the project represent the majority of the direct cost of the project. On the other hand, the indirect costs can vary significantly depending on the project, geography, and even the client and the investor. Therefore, sector experts must be careful when extracting and reviewing indirect cost-related clauses, as these will directly affect the project's profitability.

Indirect Cost-Related Clauses (CRCs) can be found in any tender documentation. It can be related to key expert or company registration in the country where the project is located, or it can be related to the accommodation or transportation of the personnel and/or equipment. Hence, the potential bidders must thoroughly investigate and examine the whole tender documentation to avoid overlooking crucial indirect CRCs.

However, some risks are included for the bidders in this examination process. First, the clients can be too demanding, so they publish voluminous documentation, resulting in bidders losing time. Second, the bidders can face tight deadlines, which leaves little chance for a thorough evaluation of the tenders. Otherwise, if they read all the documentation carefully and one by one, they might not be able to submit the tender until the deadline. Lastly, there is a human risk of missing essential CRCs. The traditional method of the manual reading process is time-consuming and susceptible to oversights. Accordingly, there are two options that the bidders might face by overlooking a crucial cost-related clause. One is submitting a non-profitable financial proposal, and the other is submitting a non-competitive tender.

Manual compliance checking and reading whole documentation is costly and time-consuming (Eastman et al., 2009; Boukamp, F., 2006). Additionally, not understanding the entire tender package causes disputes among the contracting parties, and these disputes cause significant loss of time and money for the project parties (Walsh, 2017).

Several studies in the literature aim to extract the required information and clauses from the tender documentation. The reason is that the experts still carry out the review and reading processes manually (J. Liu et al., 2022), and experts must read, comprehend and review more than 50,000 clauses carefully (Moon et al., 2021). Additionally, in recent years, the clients have not allowed sufficient timeframes for the tenderers to fully understand the project scope and prepare their documents thoroughly (Lee et al., 2019). Therefore, researchers are trying to find ways to process the clauses by using artificial intelligence algorithms. Accordingly, Hassan

& Le (2020) conducted an experiment to compare the performances of Support Vector Machine (SVM) and sector experts in identifying the requirements from the construction contract documents. The experts are divided into two groups, a speed group and a baseline group, and they are given 15 and 30 minutes, respectively. On the other hand, the SVM model processed all the requirements in just 1.88 seconds and performed the best in recall value, which shows the performance of finding all the important project requirements. It also demonstrates satisfactory performance/above average in accuracy and precision.

Accordingly, Hassan & Le (2020) aimed to extract requirements from the construction contract documents using NLP and ML Techniques. Jafari et al. (2021) proposed a methodology to extract reporting requirements and to predict the time-cost of reporting requirements using NLP, Machine Learning, and stochastic simulations. Jeon et al. (2021) extracted the quality requirements from technical specifications using NLP techniques by applying different word embedding techniques and neural network combinations. Lee et al. (2019) developed a model to identify the poisonous clauses in the contract using Rule-Based NLP. Moon et al. (2022) detected the risk clauses from the specifications using BERT. The importance of the study is that the model is the first attempt at using BERT in construction document review. The model aims to detect seven risk categories that are prone to disputes. Candaş & Tokdemir (2022) developed a multilabel text classification model to classify sentences into five classes: financial, operational, planning, technical, and contractual. They used NLP and supervised machine learning algorithms to train and obtain better metrics with support vector classifiers. Caldas et al. (2002) also aimed to classify the information into 13 concepts.

Jianan et al., 2024 focused on classifying various knowledge types within engineering consultancy service documents. They employed pre-trained language models to extract textual features and applied a multi-label classification approach to categorize the texts. The study identified several distinct knowledge types, including terminology, operation and maintenance, construction, and investment. The proposed Longformer-RoBERTa-based model demonstrated strong

performance in classifying these knowledge types, proving effective in handling text classification tasks in whole-process engineering consulting.

While the previous research studies focus on extracting requirement clauses, quality-related clauses, and poisonous clauses from the construction contract documents, there is a gap in identifying indirect cost-related clauses from the construction consultancy tender documents due to the niche of the topic. To fill this gap, this study proposes Natural Language Processing Algorithms to extract Indirect Cost-Related Clauses from tender documentation.

In an era marked by fast technological advancement, the application of artificial intelligence (AI) has affected various sectors, revolutionizing traditional practices and augmenting efficiency and speed. Among the sectors affected by AI, the construction industry and tender processes have seen significant transformations.

The proposed algorithm processes entire sentences from the tender documentation and classifies them into cost-related and non-cost-related clauses. Precision, recall, and F1 scores are utilized to measure the algorithm's performance. Shortly, Recall measures a model's ability to identify all relevant data points in a dataset, while precision assesses the model's ability to return only the relevant data points. The F1 score combines recall and precision by calculating their harmonic mean. Various technique combinations of Machine Learning Algorithms and NLP algorithms have been employed to achieve maximum precision, recall, and F1 score in identifying the cost-related clauses. These techniques include BERT, LSTM, bi-LSTM, CNN and RNN, implemented through Bag-of-Words (BoW) and TF-IDF, GloVe and Word2Vec methodologies. These techniques have advantages and disadvantages when classifying the clauses. For example, BERT may give more accurate classification results because it is pre-trained on a large corpus (Pham & Han, 2023). LSTMs excel at handling sequential data, but they can be slow to train due to their backward propagation through time (BPTT) (Yu et al., 2019). Bi-LSTM enhances this by processing the data in both directions (past and future input features) (Huang et al., 2015). CNNs are beneficial for extracting sentence features and patterns,

though they might miss long-range dependencies. However, Dauphin et al. (2017) argue that fine-tuned gated CNNs can also model long-context dependency. RNNs perform effectively across various tasks, except for recognition tasks, such as sentiment detection and QA matching. (Yin et al., 2017). The BoW method is a simple and unigram-based representation but lacks the relation between the unigrams/words (George & Joseph, 2014). TF-IDF weights the important terms but misses the semantic relationships between the words in a sentence (Wendland et al., 2021). Word2Vec provides dense word embeddings that capture semantic relationships, but such semantic relationships are only limited to local information of the language words (Johnson et al., 2024). Unlike Word2Vec, GloVe shows how to engage the global statistical information contained in a document (Dharma et al., 2022). In summary, algorithms' advantages, capabilities, and uses vary according to the needs.

The first research questions of this study are: “How are the performance of the classical machine learning algorithms such as SVM, NB, LR, and RF as well as deep learning algorithms such as LSTM, Bi-LSTM, CNN, and RNN for extracting indirect cost items from tender documents?” The overall study also queries BERT performance on binary classification.

The second research question is, “How can indirect cost-related clauses be categorized to help bidders distinguish the clauses by their content and reduce the allocated time for preparing the indirect cost of the financial proposal?”

The classification performance of binary classification was compared to that of other classifications in terms of accuracy, precision, and F1 score. On the other hand, multi-label classification performance was measured using the appropriate metrics, such as accuracy and macro and micro F1 scores.

To address the research questions stated above, this study aims to identify, extract, and categorize Indirect Cost-Related Clauses within the tender documentation using various techniques. Once the clauses are identified, this study uses a BERT model to train and process the Indirect Cost-Related Clauses and categorize each sentence into

four different categories, including Accommodation/Transportation (AT), Administrative Costs (AC), Office/Equipment/Laboratory (OEL), Personnel Related Payments (P).

This thesis consists of the following chapters: Chapter 2 provides essential background information on NLP, highlighting its significance and usage in the construction industry. Chapter 3 outlines the methodology of performing classification tasks and necessary transactions. Chapter 4 summarizes the techniques for selecting and labeling clauses, offering insights into preparing the dataset. Chapter 5 explains the comprehensive classification process and the steps involved in model preparation. After that, Chapter 6 gives the results of both binary and multi-label classification tasks, along with an analysis and commentary on the outcomes. Finally, Chapter 7 concludes the study by summarizing the findings and highlighting the contributions made by this research.

CHAPTER 2

LITERATURE REVIEW

This section provides background information on NLP and its application in the construction industry.

2.1 Natural Language Processing

Natural Language Processing (NLP) is a sub-field of Artificial Intelligence (AI), concerning the interaction between computers and humans through natural language. The main objective of NLP is to enable AI to understand, interpret, and generate language in a meaningful and serviceable way. K.R. Chowdhary (2020) states that NLP is a set of computational methods for automatically analyzing and representing human languages driven by theoretical foundations.

There are different tasks for which Natural Language Processing stands for. Text Understanding is analyzing and extracting meaning from text using various techniques, such as sentiment analysis, which is the computational treatment of opinions, sentiments, and subjectivity (Medhat et al., 2014). Named entity recognition (NER) involves identifying mentions of specific entities in the text that belong to predefined semantic categories, such as people, locations, and organizations (J. Li et al., 2022). Text classification, which is utilized in this paper, categorizes text into predefined labels, aiding in organization and analysis. Text Generation is another sub-field of NLP that creates coherent and contextually appropriate texts, such as language translation by Google Translate (J. Zhang & Zong, 2015). Automatic Speech Recognition (ASR) enables computers to understand and identify human speech (Thakur et al., 2023). This tool is also observable in numerous applications on mobile phones or computers.

Language generation is the process of creating natural language text to achieve specific communicative objectives. (Dong et al., 2021). Language generation, such as Chat-GPT, can be observed in chatbots and/or virtual assistants and creates human-like responses. Information retrieval focuses on locating documents that match a user's query from within an extensive collection (Voorhees, 1999), (Manning et al., 2008).

2.2 Natural Language Processing in Construction Industry

With the advancement of Artificial Intelligence, every industry tends to be more digitized; however, according to the report published by McKinsey Digital Globalization Index (K.R. Chowdhary, 2020), the construction industry is one of the lowest digitalized industries. Nevertheless, AI's booming impact in the construction industry, specifically in natural language applications, is inevitable. While the total number of papers published related to NLP in construction was 11 in 2016, it was 28 in 2020 (Ding et al., 2022). According to Shamshiri et al. (2024), the number of NLP applications in construction research has been significant, with 42 studies published in 2021 and 39 studies published in 2022. Furthermore, they showed that approximately 20% of the publications are dedicated to construction safety and management. Conversely, only 2% of the publications have focused on Cost Management up to August 2023.

Even though this study is related to construction tender documents and bidding, to better understand the use of NLP in the construction industry, other areas, such as safety, project management, and BIM, should be explored shortly.

Shin & Issa (2021) propose a voice-based BIM information retrieval framework using NLP applications. Williams & Gong (2014) present an algorithm that predicts construction cost overruns using text mining and numerical data. Fan & Li (2013) published a study in the construction contract domain that retrieves similar cases for alternative dispute resolution in construction accidents using text mining techniques.

Also, there are studies from a construction project management perspective; for example, Khalef & El-Adaway (2021) present an algorithm that detects substantial changes in a project through machine learning and NLP comparative analysis. Pan et al. (2021) published a text-mining-based patent analysis of BIM applications in construction in the BIM domain.

(Do et al. 2023) proposed a study to quantify the project similarity by using Cost-Weighted TF-IDF. The similarity method is specifically designed to facilitate project classification for data-driven construction cost and schedule estimation, as well as to support contract bundling.

NLP can be used in the construction industry at the bidding stage to process/review the tender documentation and/or contract documents faster. Therefore, the studies in that domain are summarized below:

Moon et al. (2021) propose a Named Entity Recognition (NER) model to review construction specifications automatically. Their model starts with data collection and its conversion to text. After tokenization (breaking the text into individual words) is performed, the labeling (assigning categories to the sentences from the data) and word embedding processes (converting the words into numerical vectors that represent their meanings) stand out. After adjusting the model input, the model is trained to categorize each specification clause into five categories. On the other hand, NER is effective for identifying and categorizing specific entities within a text; it does not provide a comprehensive solution for categorizing sentences based on their content.

Jafari et al. (2021) present an NLP and Machine Learning (ML) model that extracts the reporting requirements from the construction documents. Their classification algorithm relies on the Rule-Based Method to extract n-grams (sequences of co-occurring words). The rule-based method is typically in the form of “if-then” statements, and it involves creating a set of linguistic rules or patterns created manually based on expert knowledge.

Another study that used a rule-based approach is presented by Lee et al. (2020). The study analyzes the unstructured text data through preprocessing, syntactic analysis, and semantic analysis to detect missing contract conditions from FIDIC that may benefit the contractor. Researchers stated that the model aims to minimize human errors by identifying potential contractual risks, and it supports the contractor during the bidding and contracting phase of the project by developing a decision support tool.

Padhy et al. (2021) proposed another rule-based approach to identify exculpatory clauses in construction contracts. The model begins with preprocessing tasks and continues with rule matching and clause extraction using the PhraseMatcher tool, which is available in Spacy. The study also has a validation part with sector professionals, and the study revealed that out of 46 clauses, 37 (80.43%) were correctly identified as exculpatory, while 9 clauses (19.57%) were incorrectly tagged by the model despite containing common exculpatory phrases. Additionally, the experts found seven exculpatory clauses that the model failed to identify due to issues, including spelling and grammatical errors, special character encoding, lack of common phrases, and discontinuous phrasing.

The performance of the Rule-Based method is often compared with other Machine Learning Algorithms (Naive Bayes, Support Vector Machine, Logistic Regression, Feedforward Neural Networks, etc.). One of the studies is the identification of requirements from contract documents (Hassan & Le, 2020). The study uses a rule-based method and ML algorithms to label the sentences as requirement or non-requirement. To work with the rule-based method, the researchers add the words to the algorithm one by one. On the other hand, researchers follow the Data Pre-processing, Sentence Representation, ML Algorithm Implementations and Evaluation Steps to work with the ML algorithms.

Another study that uses Rule-Based NLP is to enable the automatic extraction of poisonous clauses in international construction contracts. Lee et al. (2019) presented a model that detects the clauses in a contract by using pre-defined rules.

Classical machine learning algorithms can classify the sentences into the desired categories. Ul Hassan et al. (2020) carried out a classification process using NB, SVM, LR, K-nearest number, decision tree, and FFNN to classify the sentences as design, construction, and operation and maintenance (O&M). The study showed that the highest accuracy belongs to the logistic regression with 94.12% and showed that the study will reduce reading time.

Hassan & Le (2021) proposed another study for classifying the sentences as design, construction and operation and maintenance (O&M). Different ML algorithms and deep neural network classifiers were compared to each other by using various text representation techniques, including Bag-of-Words, Word2Vec, GloVe, TF-IDF+Word2Vec, and TF-IDF+GloVe. Traditional algorithms using the Bag of Words (BOW) method outperformed deep neural networks (DNNs) with word embeddings. SVM and Logistic Regression achieved the highest F-scores of 92.67% and 91.27%. Word2Vec and GloVe embeddings significantly decreased performance, especially when weighted by TF-IDF. Traditional algorithms with BOW consistently performed best in precision, recall, and F-score.

Candaş & Tokdemir (2022) proposed a multilabel classification model by using REG, DTCs, SVCs, GNB, MNB, and multilayer perceptrons (MLP) to classify the sentences into one or more predefined labels/classes. Accuracy, macro and micro metrics, and hamming loss were used to measure the performance of the algorithms. They observed the data into six different group splitting in accordance to the n-gram (unigram, bigram and trigram) and train-test split ration (70%-30%). They also stated that the highest precision and recall metrics were obtained using support vector classifiers (SVCs) and that the model can be used without further training.

Caldas et al. (2002) published a study that classified documents into 13 sub-groups (general, schedule, building, structure, etc.). A total of 845 documents were classified, and the Rocchio algorithm, Naïve Bayes, k-nearest neighbors, and support vector machines were used in the classification. The analysis showed that support vector machines achieved the highest accuracy at 91.12%, comparable to human

performance in text classification tasks. Other methods had lower accuracies: Rocchio algorithm at 64.71%, IBM Miner for Text at 60.95%, naïve Bayes at 58.82%, and k-nearest neighbors at 49.11%.

Hierarchical information classification for three levels with 121 CSI Master Format items was performed by (Caldas & Soibelman, 2003). Naive Bayes, k-NN, Rochio, and SVM were used in this study, and SVM outperformed the other classification methods.

Salama & El-Gohary (2016) proposed a semantic, ML-based text classification algorithm for classifying clauses of general conditions. The total number of labels identified in this study was 14, including emergency management, security, and environmental. Different ML classifiers have been facilitated in the study, including Naïve Bayes, Maximum Entropy (ME), and Support Vector Machines (SVM). The classification was tested without preprocessing steps and feature selection on the raw data, and SVM outperformed the other classifiers. Therefore, SVM was used for all possible combinations of the feature selection methods. On the other hand, they observed an increase in precision and recall using stemming. The study also compared TF and TF-IDF for feature weighting, finding that TF achieved better precision and similar recall in most combinations despite TF-IDF usually enhancing term discrimination.

Zhou & El-Gohary (2016) studied construction-related regulatory documents utilizing multilabel text classification. The study includes a comparison of ontology-based classification with a non-ontology-based supervised ML-based approach. For the ML-based approach, SVM was selected after testing commonly used ML algorithms, including DT, NB, etc. The results demonstrated that the proposed ontology-based approach effectively leverages the semantics of the text to enhance text classification performance.

Neural Network applications can also be used to classify sentences. Jeon et al. (2021) propose models that use CNN and RNN by applying different word embedding techniques, either Word2Vec or GloVe. The algorithm processes the Standard

Specification and classifies the sentences as Quality Requirement or not. At the end of the study, the algorithm converts the required sentences into check questions via syntactic analysis, which analyses the grammatical structure of sentences in accordance with the formal grammar.

Bidirectional Encoder Representations from Transformers (BERT) is a pre-trained language model and can understand complex structures by performing fine-tuning. Moon et al. (2022) proposed a study that detects contractual risk clauses from specifications using BERT. The study also involves a comparison between the BERT model and Support Vector Machine (SVM) and a Simple Deep Neural Network (SDNN).

The NLP classification models do not have to be single-task models, which is basically a yes or no answer. Pham & Han (2023) proposes a multi-task classification model for semantic prediction of Risk-Handling Actions in Construction Contracts. The study has three classification classes: risk identification, allocation, and response. The study shows that multi-task classification performs better than single-task classification in classifying sentences. The model also shows the best result when the BERT model is used.

In accordance with the studies explained above, there is a clear gap in identifying and classifying the indirect cost-related clauses, specifically in the construction consultancy domain. Therefore, this study addresses the gap in the NLP in the construction domain, including the comparison of different algorithms.

CHAPTER 3

METHODOLOGY

This chapter outlines the methodology steps summarized in Figure 3.1.

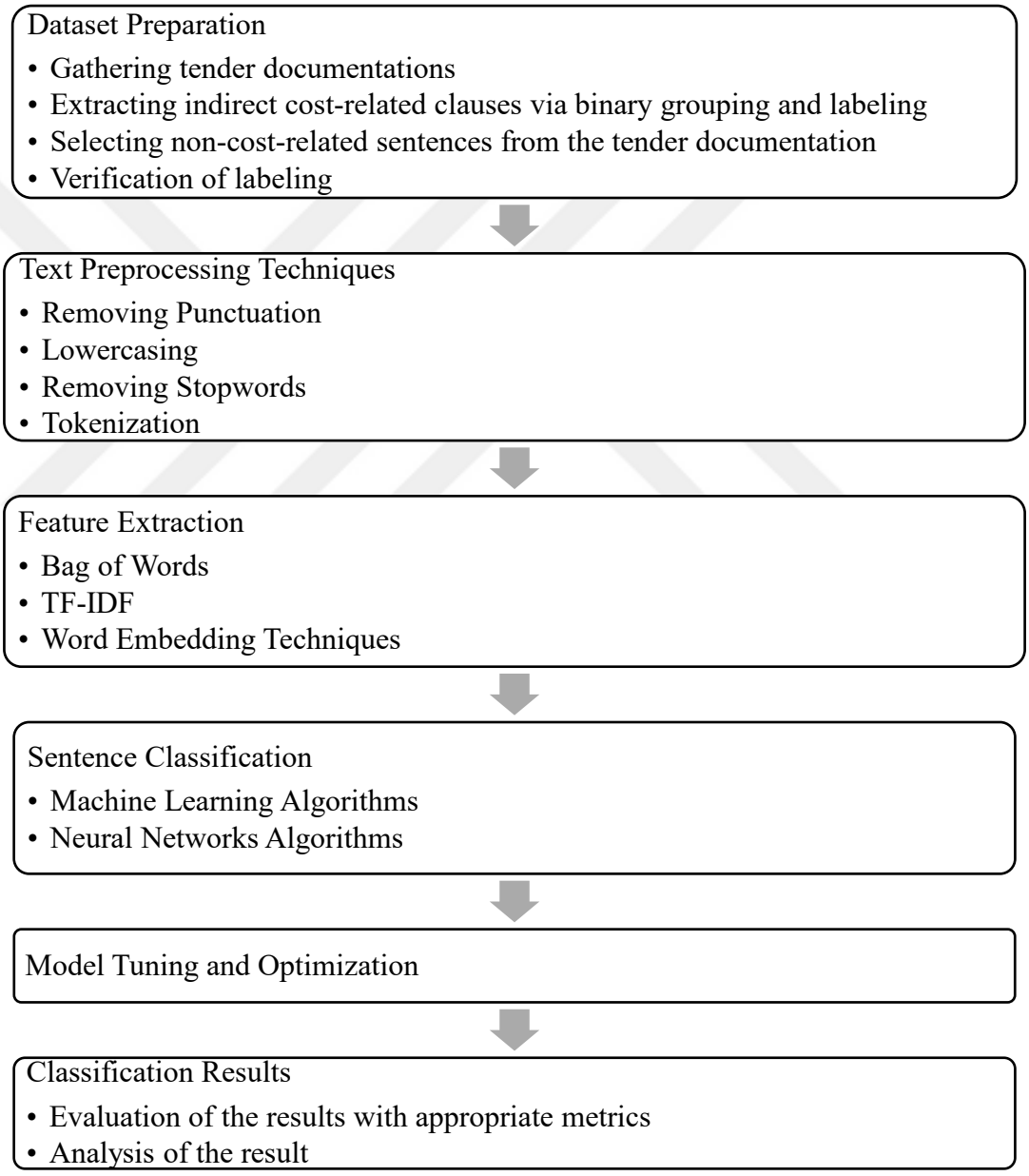


Figure 3-1 Methodology

3.1 Dataset Preparation

Due to the lack of a specific dataset in the literature to extract and classify cost-related clauses in tender documents, a dataset had to be created. To populate a pool of tenders, construction consultancy tenders have been selected carefully, paying attention to their differences in their scope of works, geographical locations, clients and funders. Consequently, 28 construction consultancy tenders have been collected from different geographies, Multi Development Banks (MDBs), and clients, both state and private internationally. Therefore, the selected sentences show different varieties.

This study divides the classification process into two: binary and multi-label. First, the sentences are classified as indirect-cost-related clauses and non-indirect-cost-related clauses, which is a binary classification. After finalizing the binary classification, the further classification process is only applied to the cost-related clauses, which are the outputs of the binary classification. Therefore, the extracting and labeling processes are needed for the next steps.

Cost-related clauses have been selected from the tender documentation for binary and multi-label classification. After that, an equal number of non-cost-related clauses were arbitrarily chosen from the tender documentation to create a balanced dataset. After finalizing the reading and reviewing process, the labeling process is verified. Detailed information regarding the dataset preparation steps is given in Chapter 4, Indirect Cost-Related Clause Selection and Dataset Preparation.

3.2 Text Preprocessing

Text preprocessing is one of the most critical tasks for text classification. In the literature, different preprocessing techniques are applied to the sentences before classification processes, such as tokenization, stop word removal, lowercasing, punctuation removal, stemming, lemmatization, etc. Experiments conducted by Uysal & Gunal (2014) show that choosing the right combinations of preprocessing

techniques might increase the classification accuracy, depending on the dataset and the language. Also, White et al. (2018) indicate that the methods are useful depending on the corpus and the goal of text mining. Therefore, careful consideration should be given when selecting the preprocessing techniques used in the algorithms. Detailed information about the text preprocessing techniques used in this study is presented in Chapter 5.

3.3 Feature Extraction

After applying the text preprocessing methods to the sentences, the words in the sentences must be converted into a numerical format suitable for ML algorithms. Different feature extraction models, including the ones employed in this study, are presented in Chapter 5.

3.4 Sentence Classification

In previous steps, preprocessing techniques were applied, and the words were converted into numerical vectors for the algorithms to use. Hence, the process continues with the application of the classification models.

This study employs classical ML algorithms, such as SVM and NB, and neural network applications, such as LSTM and Bi-LSTM, and BERT, for the binary classification process. Additionally, BERT is used for multi-label classification. The models that are used in this study are presented in Chapter 5.

3.5 Model Tuning and Hyperparameter Optimization

Using the default setting of the algorithms generally does not give the most accurate and precise results. Therefore, hyperparameter optimization is required to find the best parameters to be deployed in each model. However, the suitable hyperparameters that will be optimized should be carefully selected for each model,

and the same optimized results of the hyperparameters must not be used for all the models in the algorithm because the different hyperparameter configurations tend to work with different datasets (Kohavi & John, 1995).

On top of the discussion, it is known that hyperparameter optimization improves the overall algorithm performance compared with the default settings (Mantovani et al., 2017), (Olson et al., 2018), (Sanders & Giraud-Carrier, 2017), (Thornton et al., 2013). The information regarding which hyperparameters are optimized and the optimization results are given in Chapter 5.

3.6 Classification Results

Different combinations of feature extraction techniques and ML algorithms have been employed throughout the study. Chapter 6 summarizes the results of each combination for the binary classification, and BERT results for the multi-label classification. Additionally, Chapter 6 includes the interpretation and the insights of the results.

CHAPTER 4

COST-RELATED CLAUSE SELECTION AND DATASET PREPARATION

To work with algorithms and not cause ambiguity, the clauses that will construct the whole database must be carefully selected. Therefore, this section summarizes the method of choosing the clauses from the construction consultancy tender documentation and the labeling method for the classification processes. Additionally, since the cost-related clauses are classified into four different classification groups, this section details the further classification strategy.

This section explains the differences between direct and indirect costs in a construction project since this study focuses on extracting the cost-related clauses related to indirect costs. After explaining the differences, the selection criteria for cost-related and non-cost-related clauses are stated. The overall dataset information follows further explanation on labeling the selected cost-related clauses.

4.1 Direct and Indirect Costs

The indirect costs in a construction project refer to expenses that are not directly tied to a specific building or structure. On the other hand, they are directly related to the overall execution of the project. Understanding the indirect cost and the liabilities of the contract parties is crucial because the clauses will influence the prices throughout the project life cycle. They are necessary to calculate the project budgeting and the financial offer at the bidding stage, affecting the project's overall profitability.

4.1.1 Direct Costs

Direct costs in a construction project are directly attributable to the construction of a specific project. It is a better way to imagine the physical creation of the structure or

facility to comprehend the direct costs better. Direct costs can be divided into 3 as follows:

1. **Labor Costs** consist of wages and salaries of the workers, engineers, architects, etc. They also include insurance, retirement contributions, and other accounting terms.
2. **Material Costs** refer to the materials used throughout the construction process, including transportation costs. The direct transportation costs should not be confused with the indirect transportation costs. The indirect transport costs refer to the fuel of the site car, transportation cost of the expert to the work site, etc.
3. **Equipment Costs** include renting and/or purchasing construction machinery or equipment, such as cranes and excavation equipment. The term also includes the maintenance and operation costs of this equipment, such as fuel, oil, and repairs.

4.1.2 Indirect Costs

The indirect costs in a construction project refer to expenses that are not directly tied to a specific building or structure. On the other hand, they are directly related to the overall execution of the project. Understanding the indirect cost and the liabilities of the contract parties is crucial because the clauses will affect the prices throughout the project life cycle. Moreover, they are necessary to perform the project budgeting and the financial offer at the bidding stage. Therefore, they affect the overall profitability of the project.

There are different approaches to what kind of expenses should be considered indirect costs. A study by Saini et al., (2021) summarized the indirect expense items from various approaches for both project indirect costs and company (HQ) indirect costs. The study categorized the project indirect costs into twenty categories and company indirect costs into sixteen categories, synthesizing previous publications.

Since this study focuses on the project's indirect costs, a comparison demonstrating the differences and similarities with Saini et al. (2021) is presented in Table 4.1.

As can be observed from the study by Saini et al. (2021), there are different perspectives on identifying and clearly explaining what indirect costs include. Therefore, various combinations of expenses are used in this study to define the indirect costs. Additionally, it should be emphasized that this study does not aim to extract all the indirect cost-related clauses, but it seeks to extract/identify the clauses that are desired by the user based on sectoral experience and prepare a comprehensive financial proposal from an indirect perspective.

This study focuses on extracting the indirect cost-related clauses from the construction consultancy tender documentation, as discussed earlier. A unique approach has been developed for this specific study to align with the researcher's needs within the tender department. The reason for this unique approach lies in its ability to encompass the major items and/or clauses of the indirect costs, such as transportation and accommodation, while simultaneously preventing and mitigating data imbalance within the dataset.

Considering the researcher's requirements and the sub-items of the indirect costs, four main clauses have been identified to be considered in the study and to be classified in multi-label classification. Two major motivations exist for sentences to be grouped into four different sub-clauses. One of them is that items grouped together tend to be located in similar sections of the tender documentation due to their nature. For instance, the tender package often mentions accommodation and transportation issues in the same paragraph, as are office, equipment, and laboratory-related sentences. Another important consideration is that including every specific indirect cost-related sentence in the construction consultancy tender documentation could lead to data imbalance due to the numerous indirect cost items prevalent in the construction sector. To address this, sentences have been grouped into four optimized categories, ensuring a balanced dataset while encompassing most indirect costs pertinent to consultancy tenders. In practice, these items are typically priced

together in the pricing sheets. For instance, accommodation costs are often grouped with transportation rather than equipment, frequently in the same Excel sheet. Although this approach may vary among practitioners, the underlying logic remains consistent: these are related costs. For example, the costs associated with a project manager's travel and accommodation are added as additional expenses to the project manager's overall cost. In contrast, equipment costs are excluded from this calculation. Consequently, the four identified groups provide an optimized classification scheme that captures the majority of indirect costs in consultancy tender documents.

Before reading and reviewing the tender documents, the following sub-clauses have been defined for indirect cost, which will be searched in the tender documents:

a- Accommodation/Transportation (AT)
b- Administrative Costs (AC)
c- Office/Equipment/Laboratory (OEL)
d- Personnel Related Payments (P)

Accommodation/Transportation (AT) heading refers to the costs associated with site accommodation. This includes determining whether the contractor or consultant will be responsible for accommodation expenses and addressing the client's requirements for on-site accommodation or rental arrangements. Additionally, this sub-clause examines the clauses for transportation costs for experts involved in the project. It assesses the necessity of having experts on-site and identifies who will bear the transportation costs for their deployment.

Administrative Costs (AC) heading refers to the costs associated with the administrative costs in the tender and project duration, such as taxes, insurance, cost adjustments, fees, and duration. Taxes are mandatory charges imposed by the government, affecting the overall project budget during the tender stage and throughout the project duration. Therefore, the clauses related to tax regulations must be investigated. Insurance and Guarantee heading refers to the cost of the brokerage. The brokerage fee depends on the Guarantee percentage and/or amount that is

requested by the client and also depends on the insurance types for the whole project, such as Professional Liability, etc. On the other hand, price adjustment and payment clauses affect the overall project cash-flow, therefore, it should be included to be investigated. Price adjustments can lead to variations in the project budget, thus influencing the indirect costs. Additionally, the duration of the project is also added under this sub-item because the duration impacts the overall project budget. Many overhead items need to be multiplied by the project's duration, affecting the total indirect costs.

Office/Equipment/Laboratory (OEL) sub-clause refers to the costs associated with the office expenses at the site, including the office equipment or survey equipment that will be used at the site. Additionally, this sub-clause addresses laboratory requirements, specifying whether the contractor will provide laboratory facilities or if the consultant must establish a new laboratory for the project. The clause also delineates the responsibility for office setup, specifying whether it will be managed by the contractor or the consultant.

Personnel Related Payments (P) encompass costs associated with the registration of the experts in the country where the project is located, including the licensing fees for those experts that will be deployed in the project. Additionally, the total number of person-month information and the information on whether the support experts are needed or not are also included under this heading. Moreover, it consists of the support Experts' costs, HQ Support, translators & security guards, along with the insurances and fees in the salaries.

Table 4-1 Comparison of Indirect Cost Terminology

This Study	Literature (Saini et al. (2021)
Accommodation / Transportation	• Project-related travel expenses • Transportation/ Conveyance at site • Misc. expenses (Repair and maintenance, fuel etc.) • Rent (equipment, tool, office trailer, etc.)
Administrative Costs	• Bonds, Insurance and Taxes • Consultancy fees
Office / Equipment / Laboratory	• Safety supplies for workers • Site utilities • Site Office expenses • Site temporary structures • Rent (equipment, tools, office trailer, etc.)
Personnel Related Payments	• Jobsite staff salaries* • Site security

*Jobsite staff salaries do not refer to their gross salary; instead, they refer to the existence of support experts, back-up support & experts, translators, and security guards, including the leave, insurance, contributions, etc. Table 4-2 presents the examples of each sub-clause in the dataset.

Table 4-2 Examples of Sentences from Dataset (cont'd)

Sentences	CR (1)	Cat.
The Consultant shall provide vehicles for site transportation for all of his staff, for the entire duration of the Supervision Contract.	1	AT
These vehicles shall be maintained, fueled and insured by the Consultant until handover.	1	AT

All local transport and security costs for the Consultants staff including the site supervisory staff shall be provided by the Consultant and shall be included in the fee proposal submitted.	1	AT
An advance payment in the amount of ten (10) percent of the Contract price shall be made against an advance payment bank guarantee acceptable to the Client.	1	AC
Price adjustment on the remuneration does not apply.	1	AC
The Consultant shall provide data processing equipment for himself (i.e. laptops/notebooks, other day to day office supplies, telephone service, etc.) and other communication facilities for all of Consultant's staff, for the entire duration of the Consulting services contract.	1	OEL
After providing the offices and equipment the consultant shall maintain them for the duration of the Service	1	OEL
Estimated input of Non-Key Experts' time-input: 215,5 person-months.	1	P
The Consultant's remuneration rate for an expert shall be deemed to cover payment of overtime, sick leave, vacation leave, taxes or any other benefit that the Consultant may agree to pay the Expert as part of the remuneration package agreed separately between the Consultant and Expert	1	P
The cost of the support staff is to be covered by the Experts fees.	1	P
The expected outcome is to achieve within the designated contract period and allocated budget high quality roads with sound engineering properties that are safe for the public to travel.	0	-
Footnote: CR: Cost-Related. *AT: Accommodation/ Transportation.* *AC: Administrative Costs.* *OEL: Office/Equipment/Laboratory.* *P: Personnel Related Payments*		

4.2 Dataset

4.2.1 Selection Techniques

The cost-related clauses in the tender documents can be located in Instruction to Bidders (ITB)/Data Sheets, Terms of Reference in Construction Supervision Tenders and the Model Contract, which includes the General Conditions of the Contract and Specific Conditions of the Contract. Therefore, to detect the relevant cost clauses, these three documents of the construction tenders have been read by the author, who worked in the bidding department of an international company, and the cost-related clauses were identified one by one.

Due to the lack of a database for this classification algorithm, it is essential to gather and compile all the sentences/clauses from scratch. On the other hand, it is crucial to have a diverse dataset that consists of tenders with varying scopes from different geographical regions. In addition to the scope and the geography, special attention is given to selecting the tenders funded by different Multi Development Banks (MDBs) such as EBRD, EIB, ADB, WB, etc. This is important because the clients of these projects often use standardized formats prescribed by the MDBs, leading to similarities in sentence structure and content. Consequently, to enhance the robustness and reliability of the dataset, it is imperative to incorporate a wide variety of sentence types. These differences will increase the effectiveness and generalizability of the classification algorithms.

4.2.2 Dataset Information

This study used a total of 28 construction consultancy tenders. Table 4-3 presents the variety in the sector and funding sources of these tenders that span 15 countries across three continents. The tenders vary in their scope of work, covering sectors such as road, tramway, water, building and railway sectors. This diversity in scope, geography, and funding sources ensures a comprehensive dataset for the study.

Table 4-3 Dataset Information

Feature	Frequency
Sectoral Differences	
Road	12
Tramway	1
Water	7
Building	6
Railway	2
Geographical Differences	
Europe	17
Asia	6
Africa	5
Funder Information	
EBRD	5
EIB	5
WB	6
Local Fund	2
AIIB	2
ADB	5
EBRD & EIB	1
WB & AFD	1
AFD	1

4.2.3 Labeling Techniques

Within these 28 construction tenders, 1252 sentences were labeled in the examination and reviewing process. Detailed attention was given to the equality of the cost-related clauses and non-cost-related clauses to have a balanced dataset, which is a crucial matter while dealing with Neural Network Applications. For example, Mirus et al. (2020) state that training the algorithms on a more balanced

data set could enhance the algorithm's performance. The methodology of selecting the clauses is demonstrated in Figure 4-1.

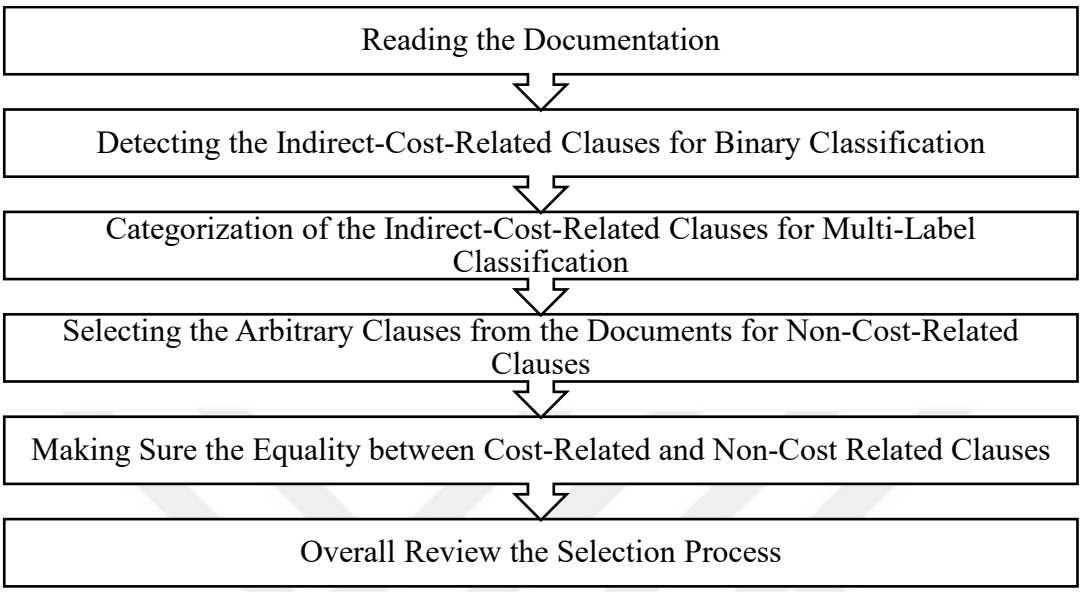


Figure 4-1 Sentence Labeling Steps

After completing the overall process of reviewing tender documentation, the final number of sentences for the binary classification is shown in Table 4-4.

Table 4-4 Number of Clauses in Dataset

Total Tender Documents:	28
Total Cost-Related Clauses:	626
Total Non-Cost Related Clauses:	626

Examples of the Binary Classification are tabulated in Table 4-5 and Table 4-6.

Table 4-5 Example of Indirect Cost-Related Clause

#	**Text**	**CR**
82	The Engineer/ Project Director's position and any other head office support staff will not be paid separately.	1

Table 4-6 Example of Non-Cost Related Clause

#	**Text**	**CR**
94	The expected outcome is to achieve within the designated contract period and allocated budget high quality roads with sound engineering properties that are safe for the public to travel.	0

On the contrary, the binary classification, which is formed by (0)s and (1)s, multi-label classification is the model of assigning sentences/clauses into more than two classes. In other words, an arbitrary CRC does not have to be assigned to only one group among the four different groups. This characteristic feature can identify the algorithm as a multi-label classification. For the multi-label classification, all the indirect-cost-related clauses are filtered to classify the sentences. Because of the different frequencies of the clauses in the tender documents, it is nearly impossible to obtain a balanced dataset for the multi-label classification. Examples of the multi-label classification are presented in Table 4-7 and Table 4-8, illustrating instances with a single label and multiple labels, respectively.

Table 4-7 Example of Multilabeling (1 Label)

#	**Text**	**CR**	**AT**	**AC**	**OEL**	**P**
82	The Engineer/ Project Director's position and any other head office support staff will not be paid separately.	1	0	0	0	1

Table 4-8 Example of Multilabeling (>1 Label)

#	**Text**	**CR**	**AT**	**AC**	**OEL**	**P**
76	The Consultant will provide residential accommodation for their specialists, and local and international transportation, mobile telephones, topographical survey equipment, all necessary local support staff such as secretaries/ interpreters, drivers, office assistants, field assistants, as well as any other goods, equipment or services for successful execution of Consultancy Services.	1	1	0	1	1

Consequently, the total number of the four classification groups is presented in Table 4-9, and the total combination of the sentences is presented in Table 4-10.

Table 4-9 Number of Sentences in Labels

Accommodation / Transportation (AT)	92
Administrative Costs (AC)	279
Office / Equipments / Laboratory (OEL)	167
Personnel Related Payments (P)	145

Table 4-10 Multi-Label Combinations

AT	**AC**	**OEL**	**P**	**Total number**
1	0	0	0	49
0	1	0	0	256
0	0	1	0	131
0	0	0	1	116
1	1	0	0	4
0	1	1	0	3
0	0	1	1	8
1	0	0	1	20
0	1	0	1	10
1	0	1	0	24
1	0	1	1	5
			GRAND TOTAL	**626**

Under the “Verification of labeling” step of our methodology, we engaged sectoral experts from consultancy and construction companies operating internationally. This engagement aimed to explain the labeling and categorization techniques to ensure accuracy and relevance.

The experts evaluated the methods and labels during discussions and provided valuable feedback. Specifically, the experts agreed that including duration information as a factor (designated as 1) effectively identifies clauses related to indirect costs along with other acceptances. Additionally, further user requirements have been established (see Table 4-11). For example, suppose a document requires translation into a language other than English. In that case, the relevant sentence is labeled as an indirect cost-related clause under "Personnel Related Payment," as it indicates the need for a translator in the office. Similarly, if there is a requirement for hardcopy submission, the sentence or clause is labeled as "1" in the "OEL" category, reflecting the need for physical resources such as printers, scanners, and other equipment from the consultant's office.

Table 4-11 Demonstrative Examples of Additional User Requirements

Concept & Reason & Designated Label
Concept: Duration The duration of a project significantly influences most of the indirect costs. As a result, it acts as a multiplier for all indirect costs within a project. Therefore, clauses that reference duration were labeled as "1" under the category of Administrative Cost. *Designated Label: Administrative Cost*
Concept: Hardcopy Submission of Reports / Documents The requirement for the hardcopy submission of documents or reports implies the need for equipment, such as printers and scanners, within the construction project. Consequently, sentences related to this concept were labeled as "1" under the Office/Equipment/Laboratory category. *Designated Label: Office / Equipment / Laboratory*
Concept: Submission of Reports / Documents in a language other than English Submitting documents or reports in a language other than English requires personnel for office translation services. Therefore, sentences related to this concept were labeled as "1" under Personnel-Related Payments. *Designated Label: Personnel-Related Payments*

Additionally, the experts validated our method for identifying non-cost-related clauses. This collaborative verification process not only enhances the credibility of our study but also ensures that our methodology aligns with industry standards and practical applications.

This step is crucial as it bridges the gap between theoretical research and practical implementation, ensuring that the findings are academically reliable and industry-relevant. The feedback from the experts enhanced the overall accuracy of our classification system.

Consequently, there was a consensus on the importance and contribution of this study, particularly in calculating indirect costs.

CHAPTER 5

CLASSIFICATION PROCESS AND MODEL PREPARATION

Chapter 5 outlines the whole classification process, including binary and multi-label classification. It begins with the preprocessing techniques, which help convert raw data into formatted and usable data in further processes. It continues with the text representation and word embedding techniques, which convert the words into numerical vectors so that the outputs of this step can be used by machine learning (ML) and neural network algorithms. After that, the classical ML algorithms and the neural network algorithms are explained in detail, along with their applications and hyperparameters, which will be optimized in the next section of this chapter.

5.1 Preprocessing the Data

The data must be preprocessed to be ready for use in the algorithms. Preprocessing involves transforming raw data into a clean, organized, and usable format. The preprocessing steps significantly increase the accuracy of the ML models and other analytical processes in choosing the appropriate tasks (Uysal & Gunal, 2014).

Kadhim (2018) states that the experiments show that text preprocessing enhances the performance of English text classification by affecting the feature extraction models. Gomes et al. (2018) used text preprocessing for sentiment analysis of social media data. In their study, text preprocessing improved the algorithm's accuracy by 20%.

However, all the preprocessing techniques do not create a massive or significant impact on the performance of the algorithms. HaCohen-Kerner et al. (2020) indicate that for some of their dataset, removing HTML tags, performing spelling correction or removing punctuation marks, and reducing replicated characters make minimal

improvement in the performance. Even stopword removal was not beneficial in one of their datasets. Therefore, it should be noted that choosing the right combination for the specific dataset is critical. The processing steps that have been applied to the algorithms are presented below.

5.1.1 Data Sampling & Equalization of Data

In many real-world applications of supervised learning algorithms, there is a significant difference in the number of classes, similar to the number of indirect-cost-related clauses, among the construction consultancy tender documentation. This issue is known as the class imbalance problem (Chawla et al., 2004) (He & Garcia, 2009) (Sun et al., 2011), and this problem causes the reduction of the performance of the algorithms. Standard learning algorithms consider a balancing dataset. Hence, they do not achieve the best performance for an imbalanced dataset (López et al., 2013) (Fernández et al., 2010). Additionally, even though it should be noted that data imbalanced problem is generally associated with binary classification (López et al., 2013), it is also a problematic issue in the multi-class and/or multi-label classification since there are more than one minority class (Fernández et al., 2013) (Lin et al., 2013).

On the other hand, since data collection for the minority class is common and costly (Weiss & Tian, 2008), many solutions have been proposed to solve the problem both for standard learning algorithms and for ensemble techniques, such as data sampling, algorithmic modification, and cost-sensitive learning (López et al., 2013) (Galar et al., 2012).

In this research, the cost-related clauses were identified by reviewing and reading the tender documentation one by one. After thoroughly reading the documentation, all the cost-related clauses have been recorded and counted. Upon completion of the counting stage, a selection of arbitrary clauses was made from the same tender

documentation, ensuring the number of these clauses matched the number of cost-related clauses in each document to prevent the effects of the data imbalance.

There are some techniques to increase/decrease the selected class in a database, called data sampling. In this study, data sampling techniques are not applied since the non-cost-related clauses are selected equal to the number of indirect cost-related clauses in the same tender documentation. The reason is that it is aimed that all the clauses in the algorithms must be from the tender documentation. It is not desirable to have clauses that are produced by AI.

5.1.2 Stopword Removal

Stopwords correspond to words that do not differ from one text document to another (Ferilli et al., 2014), and this concept was first developed by (Luhn, 1960). Stopwords are commonly used words in a language that are usually extracted from the algorithms (Rajaraman & Ullman, 2011) because they carry less meaningful information for specific tasks.

Stopword removal reduces the dataset size and, therefore, improves the overall performance of the algorithm (Babar & Patil, 2015) (J. R. Saini et al., 2016). However, extracting the word “no” or “not” would alter the meaning of the context (Chai, 2023). Additionally, it should be noted that stopwords are useful for some of the NLP Applications, such as dependency parsing, in which the stopwords are used to understand the connection between words (Chai, 2023) (Elming et al., 2013) (Poria et al., 2014).

Consequently, stopwords are extracted in this study by using the Natural Language Tool Kit (NLTK) Python package, which consists of 179 stopwords in its library (Bird et al., 2009).

5.1.3 Punctuation Removal

Punctuation Removal is one of the simplest processes of text processing and refers to the removal of the punctuation marks from the whole corpus. Even though this process is seen as a first step of text preprocessing (Kwartler, 2017), punctuation conveys important meanings beyond the words that should always not be ignored (Chai, 2023), especially for the semantic meanings of the sentences and removing punctuation may be detrimental while performing semantic analysis, especially in social media data (Koto & Adriani, 2015). Therefore, the algorithm user should be cautious about removing punctuation because the lost information will not return to the corpus (Chai, 2023).

The significance of splitting the text into sentences in NLP applications varies depending on the NLP tasks (Chai, 2023); however, in this study, the inputs for the algorithms are entire sentences. Therefore, there will be no loss of meaning, as the corpus or paragraph is not being fragmented.

5.1.4 Lowercasing

Another preprocessing technique is lowercasing where all letters in the text are converted to lowercase. The purpose of this preprocessing step is to treat the same words in the sentence in an equal way. Lowercasing is the most common form of letter case normalization (Manning et al., 2008).

Converting all words to lowercase can lead to a loss of semantic information, as some acronyms have different meanings when spelled in lowercase (Chai, 2023). Therefore, lowercasing is not particularly useful in Named Entity Recognition (NER) (Campbell et al., 2016). Moreover, it can be problematic in certain contexts, such as text summarization and sentence boundary disambiguation (Chai, 2023), (Abdolahi & Zahedh, 2017). However, it has shown success in text classification performance (HaCohen-Kerner et al., 2020). Consequently, this study applies the lowercasing process to sentences as a preprocessing technique.

After applying lowercasing, stop-words removal, and lowercasing, a random sentence from the database appears as follows:

"conduct onsite observation work progress determine work proceeding accordance contract schedule completed work conforms contracts technical specifications"

5.1.5 Tokenization

Tokenization refers to the process of breaking down input text, which appears to a computer as a continuous string of characters, into smaller units known as tokens (Grefenstette, 1999). Tokenization, simply, is a text preprocessing method that converts raw text into tokens, which are the basic units of analysis for these models. Tokenization can be in word-level, sentence-level, or sub-word level, and the process serves as the foundation for the NLP tasks (Frank et al., 2024). Within the context of this study, tokens are the words in the sentences.

Tokenization is necessary because it prepares the data for subsequent steps, such as text representation and word embedding techniques. The word token example can be observed below:

Word Tokenization: "Eray had to read all the tender documentation."

Tokens: ["Eray", "had", "to", "read", "all", "the", "tender", "documentation".]

5.1.6 Stemming and Lemmatization:

As these concepts are alike to each other, there is a difference between them. Stemming reduces words to their root by removing prefixes and suffices, while lemmatization maps words to their base or dictionary form. These concepts are not applied as preprocessing steps in this study.

5.2 Text Representations and Word Embeddings

In the previous section, the text preprocessing techniques have been explained, emphasizing that the words must be converted to numerical formats so that the ML algorithms can utilize those numerical formats. These representations allow models to process text data for various Natural Language Processing tasks.

This chapter presents the basics of vectorizing text, covering the simplest methods of text representation and word embedding methods.

5.2.1 Bag-of-Words (BoW)

The Bag-of-Words is a simple and widely used method for text representation (Qader et al., 2019). In this approach, a clause is represented as an unordered collection of words, disregarding grammar and word order. Each word in a clause corresponds to a unique dimension in the vector space, and its value represents the occurrence of that word in the vector space. This model is the simplest model to convert the text into numerical formats; however, due to their consideration only based on the number of appearances of that word, it lacks semantic information of the sentence and disregards the order of the words or the grammar (Qader et al., 2019). Another disadvantage of BoW is that it neglects the frequency of a term and/or in a document, which does not help distinguish its relevance (Egger, 2022).

To carry out the BoW method, the algorithm has to count the number of unique words in all clauses. After obtaining the number of unique words in the database, the number of occurrences of each word can be counted. To get a better understanding of Bag of Words (BoW), the following example can be given:

- **First Sentence:** Eray had to read all the tender documentation in order to extract all the cost-related clauses from the text.
- **Second Sentence:** The children have to read their homework to pass the exam.

First, it should be preprocessed the text. In this example, we will not carry out the preprocessing steps except for tokenization.

Tokens: ["Eray", "had", "to", "read", "all", "the", "tender", "documentation", "in", "order", "to", "extract", "all", "the", "cost", "related", "clauses", "from", "the", "text"]

Tokens: ["The", "children", "have", "to", "read", "their", "homework", "to", "pass", "the", "exam"]

To count the number of words in the sentences, the vocabulary must be created by combining both sentences.

Vocabulary:

["Eray", "had", "to", "read", "all", "the", "tender", "documentation", "in", "order", "extract", "cost", "related", "clauses", "from", "text", "children", "have", "their", "homework", "pass", "exam"]

After obtaining the vocabulary of the sentences, BoW vector can be created based on the frequency of each word in the vocabulary.

Sentence 1: [1, 1, 2, 1, 2, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]

Sentence 2: [0, 0, 2, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1]

These vectors represent the counts of each word in the vocabulary for each sentence. For instance, the third word of the vocabulary is “to”, and the 3rd cell of the sentence 2 matrix is “2”. Meaning that, there are 2 “to” in sentence 2.

5.2.2 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF proposed first by Salton & Yu, (1973) is a statistical method to evaluate the cruciality of a word in a document (C. Z. Liu et al., 2018). It aims to reflect the significance of a word in a document relative to a collection of documents.

The main advantage of TF-IDF over BoW is that TF-IDF weights of terms and a score is calculated for the relevance of each word in a document (Egger, 2022).

TF-IDF employs two measures in its calculation:

- **Term Frequency (TF):** This measure is similar to the BoW method. It is the frequency of a word in a document, simply stating how often a word occurs in a single document.
- **Inverse Document Frequency (IDF):** This is the exceeding part of TF-IDF when it is compared with BoW method. IDF is a measure of how common or rare a word is across all documents in the vocabulary. To calculate the IDF, the number of documents must be divided by the number of documents containing the word.

The TF-IDF is a combination of these two measures; it helps to calculate the number of occurrences of the word; on the other hand, it also helps in identifying words that are important to a document but not too common across all documents. To have a better understanding, below example can be examined:

It is assumed that the above sentences are used to explain TF-IDF and the same vocabulary can be used again.

Term Frequency:

Counting of words in Sentence 1: 20

Counting of words in Sentence 2: 11

Sentence 1: ["Eray": 1/20, "had": 1/20, "to": 2/20, "read": 1/20, "all": 2/20, "the": 3/20, "tender": 1/20, "documentation": 1/20, "in": 1/20, "order": 1/20, "extract": 1/20, "cost": 1/20, "related": 1/20, "clauses": 1/20, "from": 1/20, "text": 1/20, "children": 0, "have": 0, "their": 0, "homework": 0, "pass": 0, "exam": 0]

Sentence 2: ["Eray": 0, "had": 0, "to": 2/11, "read": 1/11, "all": 0, "the": 2/11, "tender": 0, "documentation": 0, "in": 0, "order": 0, "extract": 0, "cost": 0, "related":

0, "clauses": 0, "from": 0, "text": 0, "children": 1/11, "have": 1/11, "their": 1/11, "homework": 1/11, "pass": 1/11, "exam": 1/11]

Inverse Document Frequency (IDF):

IDF is computed as the logarithm of the total number of documents divided by the number of documents that contain the term.

Number of documents (N) = 2

- IDF("Eray") = log(2/1) = log(2)
- IDF("had") = log(2/1) = log(2)
- IDF("to") = log(2/2) = log(1) = 0

TF-IDF is a simple multiplication of TF and IDF value for each term.

Sentence 1: ["Eray": (1/20) * log(2), "had": (1/20) * log(2), "to": (2/20) * log(1) = 0, "read": (1/20) * log(1) = 0, "all": (2/20) * log(2), "the": (3/20) * log(1) = 0, "tender": (1/20) * log(2), "documentation": (1/20) * log(2), "in": (1/20) * log(2), "order": (1/20) * log(2), "extract": (1/20) * log(2), "cost": (1/20) * log(2), "related": (1/20) * log(2), "clauses": (1/20) * log(2), "from": (1/20) * log(2), "text": (1/20) * log(2), "children": 0, "have": 0, "their": 0, "homework": 0, "pass": 0, "exam": 0]

5.2.3 Word Embeddings

Egger, (2022) as cited in (Jurafsky & Martin, 2000), word embeddings are representations of words in the form of real-valued vectors that encode their meanings. Therefore, semantically and similar words are positioned closer in the vector space (Egger, 2022) such as (“hotel” vs “hostel”) and (“restaurant” vs “food”).

Vector space models can be categorized into three different models, such as Count Based Vector Space Model (Non-Semantic) such as TF-IDF, Non-Context-Based Vector Space Model (Semantic) such as Word2Vec and Glove and lastly, Context-Based Vector Space Model (Semantic) such as BERT (Egger, 2022).

(Egger, 2022) states that there are two different approaches for creating word embeddings. One of them is a counting-based approach such as GloVe algorithm, and on the other site, a prediction-based is carried out by the Word2Vec algorithm. It should be noted that both GloVe and Word2Vec have been used in this study.

5.2.3.1 Global Vectors (GloVe)

Global Vectors (GloVe) is a method for generating word embeddings, developed at Stanford. GloVe is based on the global statistical information of a corpus. The GloVe model constructs a matrix where each element represents how often a word appears in the context of another word. The purpose is to find the relationships between the words. Thanks to its ability to find the relationships, (Mikolov et al., 2013) states that GloVe – unsupervised learning of word representations surpasses other models on word analogy, word similarity, and NER tasks.

The goal of training GloVe is to find word vectors such that their dot product matches the logarithm of the co-occurrence probability of the words. The objective is to minimize the discrepancy between the dot product and the logarithm of these co-occurrence probabilities.

Additionally, the researchers publicly published the pre-trained GloVe documents with different dimensions and different numbers of tokens and vocabulary sizes. GloVe 6B document was selected for this specific study, which contains 6B tokens and 400K vocab.

5.2.3.2 Word2Vec

Word2Vec is a group of related models used to produce word embeddings. Mikolov et al. (2013) proposed two state-of-the-art model architectures for computing continuous vector representations of words from massive data sets in 2013. Therefore, the quality of the representations is measured by word similarity. They

also observed a reduction in the computational cost and good performance in catching syntactic and semantic word similarities.

Word2vec offers two different architectures to learn word representation. These two state-of-art model architectures are explained below:

- **Continuous Bag-of-Words (CBOW):** This model tries to predict the current word based on the surrounding words. According to the comparison of models presented by (Mikolov et al., 2013), the training time of CBOW is less than the Skip-Gram on the same data and better for frequently occurring words (Jang et al., 2019).
- **Skip-Gram:** Contrary to CBOW, Skip-gram tries to predict the surrounding words by using the current word. (Egger, 2022) states that skip-gram is more suitable when small data is available, that is because the algorithm can represent rare words and phrases adequately.

Hyperparameter optimization can also be performed while using Word2Vec. (Egger, 2022) summarized the most important hyperparameters in Word2Vec along with their suggested values and the information of who suggested those values.

- **Window_size:** Because both Skip-gram and CBOW work with predicting the surrounding words, the window size must be indicated. Goldberg et al. (2014) state that larger windows represent more topical similarities, and smaller windows have more syntactic similarities. The suggested value of this word2vec hyperparameter is 5 (Goldberg et al., 2014) (Horn et al., 2020).
- **Vector_size:** Vector size indicates the dimension of each word. It is suggested as 300 by (X. Zhang et al., 2017) (Yuan et al., 2018).
- **SG:** If the function is set as 0, the architecture works as CBOW. On the contrary, if it is set as 1, it works as Skip-Gram. (Horn et al., 2020) suggested that it should be set as 1, which is Skip-Gram. However, according to the classification model of (Jang et al., 2019), the CBOW algorithm performed better when the data was news articles; on the other hand, the Skip-gram algorithm performed better when the tweets were used for the data. This

implies that the performance of the CBOW and Skip-gram architectures depends on the type of data to be analyzed.

The output of Word2Vec is a high-dimensional vector for each word in the vocabulary. The vectors capture semantic similarities between words, which can be measured by cosine similarity.

The hyperparameters that have been used in this study are shown below:

"word2vec = Word2Vec(train_sentences_split, vector_size=200, window=5, min_count=1, sg=0)"

The below example is taken from the study's algorithm:

Sentence: *phase1 lumpsum services task 1 3 estimated time frame design review preparation bidding documents technical assistance bidding evaluation phase seven 7 months*

Vectorized Sentence:

[-0.00272712 0.00333275 0.00358276 -0.00106268 0.00369229 -0.00691273 0.00321666 0.01035627 -0.0046888 -0.0041376 -0.00521937 -0.00872111 -0.00267293 0.00337658 0.00116015 -0.00718355 -0.00046563 -0.00634795 -0.00136821 -0.00931838 0.0060808 -0.00016733 0.00231252 -0.00142751 -0.00026323 -0.00090338 -0.00316266 -0.00472783 -0.00531531 0.00186504 0.00468011 0.00051506 -0.00231697 -0.00427884 -0.00349652 0.00832851 0.00224783 -0.00309362 -0.00405975 -0.00866095 0.00327426 -0.00530495 0.0025382 -0.00046286 0.00629639 -0.00395509 -0.00512482 0.00091194 0.00502378 0.00433919 0.00577811 -0.00320982 -0.0009292 -0.00174951 -0.00834268 0.0036093 0.00487177 0.00026336 -0.00907379 0.00295361 0.00244684 0.00077235 -0.00051945 -0.00055664 -0.00692463 0.00230501 0.00319903 0.00270952 -0.00531632 0.00467598 -0.00387087 0.00409047 0.00767331 0.00011842 0.00726114 0.00456008 0.00013296 0.00067812 -0.00578265 0.00338571 0.00022223 -0.00019124 -0.00428851 0.00569853 0.00038126 -0.00186226 0.00416664 0.00729752 0.00603983 0.00550091

0.0061209 0.00556902 -0.00194245 0.00029817 0.00813077 0.00728207 0.00211608 -0.00590005 0.00425827 -0.00151226]

The dataset is split into train_sentences_split and val_sentences_split. While train_sentences_split stands for only the training dataset, which is evaluated as 80% of the whole dataset, val_sentences_split stands for the validation dataset to calculate the algorithm's overall performance. It is desired to vectorize the words in the training dataset. Therefore, this code is applied only to the training dataset.

```
"# Build embedding matrix

vocab_size = len(tokenizer.word_index) + 1

embedding_matrix = np.zeros((vocab_size, 200))  # Assuming embedding dimension is 200

for word, i in tokenizer.word_index.items():

    try:

        embedding_vector = word2vec.wv[word]

        embedding_matrix[i] = embedding_vector

    except KeyError:  # word not in word2vec model vocabulary

        pass"
```

From the above code, the embedding matrix is constructed where each row corresponds to a word in the vocabulary and contains the word's embedding vector. Therefore, this matrix can be used as weights for embedding layers in neural network models.

5.2.3.3 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a recent language representation, and the pre-trained BERT model can be fine-tuned with just

one additional output layer in order to prepare the model for a wide range of tasks in NLP (Devlin et al., 2018). BERT can handle sequence-to-sequence tasks; therefore, it considers the extensive dependencies between words along with their meanings (Egger, 2022).

BERT's significance lies in its use of masked language models, which allow for pre-trained deep bidirectional representations, highlighting the value of bidirectional pre-training for language representations. Additionally, BERT's pre-trained nature minimizes the need for heavily engineered task-specific architectures. Lastly, it advances the state of the art for eleven NLP tasks (Devlin et al., 2018).

For example, (Eken, 2022) shows that BERT was the best model for both sentence type classification and related party classification. Additionally, Fine-tuned BERT outperforms the other ML models with different text vectorization methods in single-task classification (Pham & Han, 2023).

BERT has 8 different model variations, including BERT-base, which consists of 110 M parameters and 12 transformer layers, and BERT-large, which consists of 340 M parameters and 24 transformer layers. The pre-trained models are based on massive datasets and contain over 3 billion English words (Egger, 2022).

BERT also has its own tokenization, which is called BERT Tokenizer. Therefore, this option is used in this study. The BERT model used in this study is BERT-Base-Uncased.

“tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

model = BertForSequenceClassification.from_pretrained(

"bert-base-uncased", #12 layer bert - with an uncased vocab.

num_labels=2,

)”

5.3 Machine Learning and Deep Learning Algorithms

This section summarizes the ML and DL algorithms that have been developed for this specific classification task. A classification task is a task where the goal is to assign a label to a given input based on its features. Various algorithms can be used for classification and each algorithm has distinct advantages and disadvantages.

5.3.1 Naïve Bayes (NB)

Naive Bayes methods are a group of supervised learning algorithms that apply Bayes' theorem (Webb & Bayes, 2010), assuming conditional independence between every pair of features given the class variable. By simplifying the relationship using this assumption, classification becomes straightforward. These classifiers differ based on their rules about the distribution of features. Despite their simplified nature, Naive Bayes classifiers have proven effective in various real-world applications like sentence classification and spam filtering, requiring minimal training data. The classification process is particularly fast due to their ability to independently estimate each feature's distribution, which helps mitigate issues related to high-dimensional data.

Naive Bayes is fundamentally a conditional probability model that computes probabilities for each possible outcome or class given a set of features. The conditional probability can be decomposed as:

$$"p(C_k|x) = p(C_k) * \frac{p(x|C_k)}{p(x)}" \qquad \text{Eq (1)}$$

Naïve Bayes is performed by using the Bag of Word (BoW) and Term Frequency Inverse Document Frequency (TF-IDF) text representation techniques.

“nb_model_bow = MultinomialNB()

nb_model_bow.fit(train_bow, train_labels)

nb_pred_bow = nb_model_bow.predict(val_bow)

nb_accuracy_bow = accuracy_score(val_labels, nb_pred_bow)

nb_model_tfidf = MultinomialNB()

nb_model_tfidf.fit(train_tfidf, train_labels)

nb_pred_tfidf = nb_model_tfidf.predict(val_tfidf)

nb_accuracy_tfidf = accuracy_score(val_labels, nb_pred_tfidf)

Naïve Bayes was used by default settings in this study. Parameters used in Naïve Bayes are presented below:

[alpha=1.0, fit_prior=True, class_prior=None]”

5.3.2 Support Vector Machine (SVM)

Support Vector Machines (SVMs), also known as support vector networks, are supervised learning models used for classification and regression tasks in ML. These models create hyperplanes in a high-dimensional space to distinguish between different data classes. The objective is to identify the hyperplane that maximizes the distance to the nearest data point of any class, referred to as the functional margin. This separation is crucial for reducing the generalization error of the classifier, thus minimizing the risk of overfitting. The Support Vector Machine models are also known for their high accuracy in text classification problems (Yuk Ying Chung et al., 2007).

Support Vector Machine is performed by using the Bag of Word (BoW) and Term Frequency Inverse Document Frequency (TF-IDF) text representation techniques.

“svm_model_bow = SVC()

svm_model_bow.fit(train_bow, train_labels)

svm_pred_bow = svm_model_bow.predict(val_bow)

svm_accuracy_bow = accuracy_score(val_labels, svm_pred_bow)

svm_model_tfidf = SVC()

svm_model_tfidf.fit(train_tfidf, train_labels)

svm_pred_tfidf = svm_model_tfidf.predict(val_tfidf)

svm_accuracy_tfidf = accuracy_score(val_labels, svm_pred_tfidf)

Support Vector Machine was used by default settings in this study. Parameters used in the Support Vector Machine are presented below:

[C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=1e-3, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None]”

5.3.3 Logistic Regression (LR)

Logistic regression is a statistical method and supervised ML algorithm used to analyze datasets in which there is one or more than one independent variable that determines an outcome. The outcome is typically a 0 or 1 variable (i.e., binary), meaning that it has two possible outcomes. The purpose of this model is to determine the probability that a given input point belongs to a certain class which means that aiming the classification. It should be pointed out that logistic regression is one of

the most useful analytic tools, thanks to its ability to study the importance of individual features.

Logistic Regression is performed by using the Bag of Word (BoW) and Term Frequency Inverse Document Frequency (TF-IDF) text representation techniques.

“log_model_bow = LogisticRegression()

log_model_bow.fit(train_bow, train_labels)

log_pred_bow = log_model_bow.predict(val_bow)

log_accuracy_bow = accuracy_score(val_labels, log_pred_bow)

log_model_tfidf = LogisticRegression()

log_model_tfidf.fit(train_tfidf, train_labels)

log_pred_tfidf = log_model_tfidf.predict(val_tfidf)

log_accuracy_tfidf = accuracy_score(val_labels, log_pred_tfidf)

Logistic Regression was used by default settings in this study. Parameters used in Logistic Regression are presented below:

[penalty='l2', dual=False, tol=1e-4, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None]”

5.3.4 Random Forest (RF)

Random Forest is a widely used machine learning algorithm in the ensemble learning family. It operates by constructing a large number of decision trees during training and outputs the class that is the mode of the classes (for classification tasks as in this study) or mean prediction (for regression tasks) of the individual trees.

Random Forest is performed by using the Bag of Word (BoW) and Term Frequency Inverse Document Frequency (TF-IDF) text representation techniques.

“rf_model_bow = RandomForestClassifier()

rf_model_bow.fit(train_bow, train_labels)

rf_pred_bow = rf_model_bow.predict(val_bow)

rf_accuracy_bow = accuracy_score(val_labels, rf_pred_bow)

rf_model_tfidf = RandomForestClassifier()

rf_model_tfidf.fit(train_tfidf, train_labels)

rf_pred_tfidf = rf_model_tfidf.predict(val_tfidf)

rf_accuracy_tfidf = accuracy_score(val_labels, rf_pred_tfidf)”

Random Forest was used by default settings in this study.

5.3.5 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a deep learning algorithm primarily used for processing structured grid data. CNNs are designed to automatically and adaptively learn spatial hierarchies of features from input data. The main difference between traditional Artificial Neural Networks (ANNs) and Convolutional Neural Networks (CNNs) is that CNNs can be used in the field of pattern recognition within images (O’Shea & Nash, 2015). CNNs comprise three types of layers: convolutional layers, pooling layers and fully connected layers. By forming these layers, a CNN architecture has been created (O’Shea & Nash, 2015).

Detailed information about the parameters of CNN is given in Chapter 6.

5.3.6 Recurrent Neural Network (RNN)

The recurrent neural network (RNN) (Elman,1990) is one of the deep neural networks, where the networks are connected in a series of structures (Nallapati et al., 2016). RNN is used to recognize the patterns in sequences of data such as text and

more. RNNs have connections that form directed cycles, allowing them to maintain a hidden state that can capture information about previous inputs.

Detailed information about the parameters of RNN is given in Chapter 6.

5.3.7 Long Short-Term Memory (LSTM)

Long short-term memory network (LSTM) was proposed by (Hochreiter & Schmidhuber, 1997) to specifically address the issue of learning long-term dependencies. LSTM is also a type of RNN architecture specifically designed to model sequential data and address the limitations of traditional RNNs. There is a cell state and three gates in the key elements of LSTM. Cell State is a key feature of LSTMs. It goes through the entire chain, allowing information to flow unchanged. On the other hand, the forget gate determines what kind of information from the cell state should be discarded. Here, 1 means keeping all information, and 0 means discarding all the information. The input gate determines what new information should be added to the cell state. It also consists of two parts: a sigmoid layer and a tanh layer. Last, the output layer determines what the next hidden state should be.

Detailed information about the parameters of LSTM is given in Chapter 6.

5.3.8 Bi-Directional Long Short-Term Memory (bi-LSTM)

Bi-Directional Long Short-Term Memory (Bi-LSTM) proposed by (Graves & Schmidhuber, 2005) network is an extension of the standard Long-Short-Term Memory (LSTM) network that improves the model's ability to capture information from both past and future contexts in sequential data. The basic idea of forward and backward concept was presented by (Schuster & Paliwal, 1997). It comprises two LSTM layers: one processes the sequence in the forward direction (left to right), and the other processes the sequence in the backward direction (right to left). The outputs

of these two layers are then combined, allowing the model to have a comprehensive understanding of the sequence.

Detailed information about the parameters of bi-LSTM is given in Chapter 6.

5.3.9 Bidirectional Encoder Representations from Transformers (BERT)

BERT, Bidirectional Encoder Representations from Transformers, is a transformer-based ML model for natural language processing (NLP) tasks. Google developed it and introduced it in 2018 (Devlin et al., 2018). BERT has significantly advanced the state of the art in NLP, achieving superior performance on a wide range of tasks.

There are some major advantages of using BERT such as it can read text in both directions simultaneously. This bidirectional approach allows BERT to capture context from both sides of a word, leading to a deeper understanding of the text. BERT is based on the transformer architecture, which uses self-attention mechanisms to weigh the importance of different words in a sentence relative to each other. Most importantly, the BERT model is pre-trained on a large corpus, and after this pre-training, BERT can be fine-tuned on specific tasks such as text classification, which will be carried out in this paper. In conclusion, its ability to be pre-trained on vast amounts of text and fine-tuned for specific applications makes it a highly effective and versatile model for understanding and processing natural language.

In this task, the Bert-base-uncased model was used to carry out tokenization and classification tasks.

“tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')”

Padding is the process of making sequences, which are sentences uniform in length. This is necessary because many algorithms, particularly those involving neural networks, require input data to have a consistent shape. In this study, each sentence was padded to 128 by using the following code:

“sentences_train_tokenized = tokenizer(sentences_train, padding=True, truncation=True, max_length=128)”

“sentences_val_tokenized = tokenizer(sentences_val, padding=True, truncation=True, max_length=128)”

Table 5-1 provides a summary of the hyperparameters used in the BERT classifier for this study.

Table 5-1 BERT Hyperparameters

Name	**Value**
Batch Size	16
Epoch	10
Warmup Steps	500
Weight Decay	0,01
Learning Rate	2x10^-5

“args = TrainingArguments(

output_dir="output",

num_train_epochs=10,

per_device_train_batch_size=16,

warmup_steps=500,

weight_decay=0.01,

learning_rate=2e-5,

logging_dir="logs",)”

Up to this section of the thesis, the various text representation techniques and algorithms applicable to the classification task have been thoroughly explained. To provide a comprehensive overview, Table 5-2 presents a combined table that

illustrates the model information, text representation techniques (vectorizers), and the purpose of each model-technique combination.

It is important to emphasize that all combinations have been compared in the context of binary classification, whereas only the BERT model has been applied for multi-label classification. A summary of these combinations is provided in Table 5-2.

Table 5-2 Classification Types and Model&Vectorizer Information

Model	**Vectorizer**	**Purpose**
Naïve Bayes	TF-IDF BoW	Binary Classification
Support Vector Machine	TF-IDF BoW	Binary Classification
Logistic Regression	TF-IDF BoW	Binary Classification
Random Forest	TF-IDF BoW	Binary Classification
LSTM	Word2Vec	Binary Classification
LSTM	GloVe	Binary Classification
bi-LSTM	Word2Vec	Binary Classification
bi-LSTM	GloVe	Binary Classification
RNN	Word2Vec	Binary Classification
RNN	GloVe	Binary Classification
CNN	Word2Vec	Binary Classification
CNN	GloVe	Binary Classification
BERT	BERT Tokenizer	Binary Classification & Multilabel Classification

5.4 Hyper-parameter Optimization

Fine-tuned and optimized hyperparameters can yield better results in terms of accuracy and precision if we compare the results with the default settings. Additionally, the hyperparameter optimization is so dataset dependent, therefore, the same configuration might not give the best results by using the same optimized parameters. Hence, it is necessary to use the same dataset in each iteration of optimization and conduct optimization for each neural network algorithm.

The process of optimization can be performed manually or automatically. In order to pursue manual optimization, the user should be able to review the previous results, and it requires expertise. Therefore, it is a time-consuming process. On the other hand, automated hyperparameter optimization can be classified into two categories: black-box optimization techniques and multi-fidelity optimization techniques (Elshawi et al., 2019). Each technique is summarized below:

- **Black-box Optimization Techniques:**
    - **Grid Search:** The working principle of Grid Search is to try all the options identified in the search space configuration and select the most accurate result. However, as Grid Search tries all the options in the configuration, the optimization would be expensive (Bergstra et al., 2012). Also, the next configuration cannot use and be facilitated by the previous configuration to find the best option (Yaloveha et al., 2022). As an overall result, Grid Search can be used when the search space is small.
    - **Random Search:** Another technique along with Grid Search is Random Search technique which finds the best result by selecting hyperparameters randomly. Bergstra et al. (2012) show that using Random Search is more efficient than using Grid Search technique.
    - **Bayesian Optimization:** Bayesian optimization is designed for optimized functions that require lengthy evaluations and it is particularly effective for optimizing continuous domains with fewer

than 20 dimensions (Frazier, 2018). This method constructs a surrogate model of the objective function using Bayesian ML, specifically Gaussian process regression. It leverages this surrogate to estimate uncertainty and employs an acquisition function derived from this model to determine the most promising locations to sample next.

- **Multi-fidelity Optimization:** Increasing the size and the number of the data and the number of the parameters must be optimized results in the difficulty of usage of black-box optimization techniques especially in terms of time. Multi-fidelity methods formalize heuristic approaches by employing low-fidelity approximations of the actual loss function for minimization purposes (Hutter et al., 2019). The multi-fidelity methods are split into three methods (Hutter et al., 2019):
    - **The Learning Curve-Based Prediction** models evaluate the model learning curves during carrying out the hyperparameter optimization and the model stops itself if adding further resources does not increase the performance.
    - **Bandit-Based Algorithm Selection Methods** aim to identify the best algorithm from a finite set by using low-fidelity approximations of their performance. These methods can also have variations such as *successive halving* and *Hyperband*. Successive halving is the process of removing the half that performed worst and doubling the budget, where it repeats until only a single algorithm remains (Hutter et al., 2019). On the other hand, HyperBand, introduced by L. Li et al. (2018), allocates the total budget across various combinations of configuration counts and budgets for each. Then, successive halving is applied as a subroutine on each set of randomly selected settings.
    - **Adaptive Choices of Fidelities** actively choose which fidelities to evaluate based on previous observations, aiming to avoid misspecification of the schedule.

This study uses a bandit-based algorithm selection method; however, it is neither Successive Halving nor HyperBand. Despite HyperBand providing speedup over its competitors in deep-learning and kernel-based learning problems (L. Li et al., 2018), it is very restrictive not to adjust the configuration proposal strategy based on the function evaluations (Hutter et al., 2019). To overcome this restriction, a new approach, BOHB, was introduced by Falkner et al. (2018). BOHB combines Bayesian optimization and HyperBand to achieve the best results. Therefore, it is used in Deep Learning Algorithms with Word2Vec and GloVe word embedding techniques.

In order to increase the reliability of the results, a few techniques were facilitated such as early stopping and clear backend session.

Validation and training curves can be observed while training the model in order to prevent overfitting. Overfitting is a fundamental problem in supervised ML that prevents algorithms from creating well-generalized models (Ying, 2019). The reason is that it fits the training data too closely and performs well on the training dataset, but it shows worse performance on unseen data in the testing/validation dataset.

Early stopping is one of the most popular techniques to prevent overfitting on the models. The main idea behind the concept of early stopping is that it stops after some identified step that shows no improvement in the model's accuracy. Figure 5-1 (Ying, 2019) explains the early stopping concept.

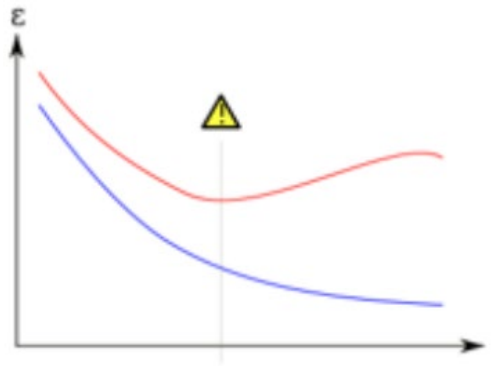


Figure 5-1 Validation Loss vs Training Loss (Ying, 2019)

In the figure, the horizontal axis represents the epoch, which is a single complete pass through the entire training dataset. During an epoch, the model processes each training example and updates its parameters based on the computed loss. On the other hand, the vertical axis refers to loss. The blue line and red line refer to training loss and validation loss, respectively. As can be seen from the figure, after some point, while training loss continues to decrease, validation loss starts to increase. It shows that the model begins to fit only on the training dataset and will not show decent performance on unseen data. Therefore, early stopping should be used in order to prevent overfitting problem as in this study.

The coding of early stopping concept:

“early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)”

The meaning of the parameters is explained below:

monitor='val_loss' shows that the early stopping triggers by monitoring the validation loss values.

patience=5 indicates the how many epochs to wait after the last improvement before stopping the training.

restore_best_weights=True ensures that the model retains the best weights seen during training.

As the nature of the optimization process is loop iteration, it should be beneficial to clear the backend information of the model in order to prevent any potential memory leaks and to ensure that each new model training starts with a clean state. Therefore, the BOHB framework iterates over multiple configurations, and each iteration involves training a new model with a different set of hyperparameters. The coding of the concept of clearing backend information is as follows:

“ tf.keras.backend.clear_session()”

However, it should be included at the beginning of the iteration so that it ensures that each iteration begins with a clean state.

The optimized hyperparameters are listed below by displaying their codings:

"config_space.add(CSH.UniformFloatHyperparameter('dropout_rate', lower=0.2, upper=0.6))

config_space.add(CSH.CategoricalHyperparameter('lstm_units', choices=[32, 64, 128]))

config_space.add(CSH.UniformFloatHyperparameter('learning_rate', lower=0.0001, upper=0.01))

config_space.add(CSH.UniformFloatHyperparameter('regularizer', lower=0.001, upper=0.05))

config_space.add(CSH.CategoricalHyperparameter('batch_size', choices=[16, 32, 64]))"

Dropout is basically a technique that prevents overfitting in the models. The key concept is to randomly omit units, along with their connections, from the neural network during the training process (Srivastava et al., 2014). Using Dropout in models significantly reduces overfitting and improves the performance of supervised learning tasks such as document classification and speech recognition (Srivastava et al., 2014). Increasing the dropout rate leads to more neurons being dropped during training, which can lead to better regularization and potentially prevent overfitting; however, on the other hand, if the dropout rate is too high, the model may not learn the necessary patterns, which is called underfitting.

LSTM units refer to the number of neurons in the LSTM/Bi-LSTM/CNN/RNN. It helps to capture long-term dependencies in sequence data. By increasing the unit, the model is able to learn more complex patterns and capture more information. However, this increases computational cost and the risk of overfitting. Therefore, there is a trade-off between increasing or decreasing the LSTM units.

Learning Rate (LR) is a hyperparameter that controls how much the model's weights are adjusted with respect to the loss gradient during each update. If the learning rate (LR) is too low, it can lead to overfitting. Higher learning rates can help regularize the training, but if set too high, the training process may diverge (Smith, 2018).

Regularizer is a technique used to prevent overfitting by adding a penalty, which is calculated by the model's parameters to the loss function. Increasing regularizer imposes a stronger penalty and helps to reduce the overfitting chance. On the other hand, decreasing the regularizer can lead to underfitting, and the model is too simple to capture the underlying patterns in the data.

Batch size is the number of training samples used to update the model's weights in one iteration. Small batch sizes have been suggested for regularization effects (Wilson & Martinez, 2003). Kandel & Castelli (2020) shows that higher batch size does not guarantee higher accuracy and lowering the learning rate and decreasing the batch size allow the model to train better in fine-tuning cases.

There is an additional hyperparameter that was optimized in this study, kernel size, however, this hyperparameter only exists in CNN model.

config_space.add(CSH.CategoricalHyperparameter('kernel_size', choices=[3, 5]))

Kernel size in a CNN refers to the dimensions of the filter that slides over the input data to perform convolution operations. It determines how many input pixels are considered at each step. Kernel size of 3 focuses on capturing small and detailed features, however, kernel size of 5 captures larger features. (Ozturk et al., 2018) shows that increasing kernel size also increases the computational time. Additionally, they also showed that CNN architect with 3x3 and 7x7 seems to be more successful, and 5x5 and 9x9 kernels memorize the data and do not learn. Therefore, it should be selected in accordance with the dataset available.

The following tables (Tables 5-3 to 5-10) present the results of hyperparameter optimization for various combinations of word embeddings and deep learning models in the binary text classification task.

Table 5-3 Word2Vec & LSTM Hyperparameter Optimization Result

LSTM Results	
Hyperparameters	**Results**
Dropout	0.5032 – *(rounded)*
LSTM Unit	32
Learning Rate	0.0045 – *(rounded)*
Regularizer	0.0054 – *(rounded)*
Batch Size	64

Table 5-4 Word2Vec & Bi-LSTM Hyperparameter Optimization Result

Bi-LSTM Results	
Hyperparameters	**Results**
Dropout	0.5355 – *(rounded)*
Bi-LSTM Unit	32
Learning Rate	0.0016 – *(rounded)*
Regularizer	0.0447 – *(rounded)*
Batch Size	16

Table 5-5 Word2Vec & CNN Hyperparameter Optimization Result

CNN Results	
Hyperparameters	**Results**
Dropout	0.2884 – *(rounded)*
Filter Size	64
Learning Rate	0.0043 – *(rounded)*
Regularizer	0.0014 – *(rounded)*
Batch Size	32
Kernel Size	3

Table 5-6 Word2Vec & RNN Hyperparameter Optimization Result

RNN Results	
Hyperparameters	**Results**
Dropout	0.2911 – *(rounded)*
RNN Units	32
Learning Rate	0.0009 – *(rounded)*
Regularizer	0.0198 – *(rounded)*
Batch Size	32

Table 5-7 GloVe & LSTM Hyperparameter Optimization Result

LSTM Results	
Hyperparameters	**Results**
Dropout	0.3002 – *(rounded)*
LSTM Unit	32
Learning Rate	0.0002 – *(rounded)*
Regularizer	0.0154 – *(rounded)*
Batch Size	32

Table 5-8 GloVe & Bi-LSTM Hyperparameter Optimization Result

Bi-LSTM Results	
Hyperparameters	**Results**
Dropout	0.4085 – *(rounded)*
Bi-LSTM Unit	128
Learning Rate	0.0012 – *(rounded)*
Regularizer	0.0380 – *(rounded)*
Batch Size	16

Table 5-9 GloVe & CNN Hyperparameter Optimization Result

CNN Results	
Hyperparameters	**Results**
Dropout	0.58264 – *(rounded)*
Filter Size	128
Learning Rate	0.0067 – *(rounded)*
Regularizer	0.0035 – *(rounded)*
Batch Size	32
Kernel Size	5

Table 5-10 GloVe & RNN Hyperparameter Optimization Result

RNN Results	
Hyperparameters	**Results**
Dropout	0.3774 – *(rounded)*
RNN Units	64
Learning Rate	0.0004 – *(rounded)*
Regularizer	0.0299 – *(rounded)*
Batch Size	16

CHAPTER 6

CLASSIFICATION RESULTS

Chapter 6, Classification Results, discusses the evaluation methods used to assess the performance of the ML algorithms. Selecting the appropriate evaluation metrics is crucial to accurately evaluate the models. Therefore, two different approaches are provided to evaluate the binary and multi-label classification.

The last section of this chapter provides the details of the classification results, summarizing the binary classification results of all ML algorithms and providing a comparison of the results of each classification algorithm. Following this, the results of multi-label classification using the BERT classifier are presented.

6.1 Evaluation Methods

Before going into the details of the evaluation methods for different classification tasks, it is necessary to give a short detail for each task:

- **Binary Classification:** The input for this classification task is to be classified into only one class. Classes should be different from each other, obviously. This classification task is the most popular task.
- **Multi-class Classification:** The input for this classification task is to be classified into only one class. However, as its name implies, there are more than two classes, rather than binary classification.
- **Multi-label Classification:** The input for this classification task is to be classified into more than one class.
- **Hierarchical Classification:** The input is to be classified into one class, which is divided into subclasses.

Therefore, this study can be categorized as a binary classification for determining the clauses as cost-related and non-cost-related and multi-label classification for classifying the cost-related sentences into sub-groups which are provided in the previous sections. The reason the classification task is multi-label rather than multi-class is that one cost-related clause can be classified as more than one class, as shown in Table 4-8.

Binary Classification performance can be measured from the Table 6-1 and by using four different terms (Sokolova & Lapalme, 2009):

Table 6-1 Confusion Matrix

Data Class	**Predicted as Positive**	**Predicted as Negative**
Actual Positives	True Positive (TP)	False Negative (FN)
Actual Negatives	False Positive (FP)	True Negative (TN)

The explanation of the terms in Table 6-1 is summarized below by using the terminology of this study:

True Positive (TP) refers to sentences that are correctly labeled. It means that the sentence is cost-related, and the algorithm correctly classified this sentence as cost-related.

False Negative (FN) refers to sentences that are wrongly labeled. It means that the sentence is cost-related, and the algorithm wrongly classified this sentence as non-cost-related.

False Positive (FP) refers to sentences that are wrongly labeled. It means that the sentence is non-cost-related, and the algorithm wrongly classified this sentence as cost-related.

True Negative (TN) refers to sentences that are correctly labeled. It means that the sentence is non-cost-related, and the algorithm correctly classified this sentence as non-cost-related.

By using these parameters, the following metrics can be calculated (Sokolova & Lapalme, 2009):

Accuracy	$\frac{TP + TN}{TP + FN + FP + TN}$	Eq (2)
Precision	$\frac{TP}{TP + FP}$	Eq (3)
Recall	$\frac{TP}{TP + FN}$	Eq (4)
Fscore	$2 * \frac{\mathrm{TP}}{2 * \mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$	Eq (5)
Specificity	$\frac{TN}{FP + TN}$	Eq (6)
AUC	$\frac{1}{2} * (\frac{TP}{TP + FN} + \frac{TN}{FP + TN})$	Eq (7)

Accuracy refers to the ratio of all correct sentences, whether it is cost-related or non-cost-related, to the total number of sentences.

Precision is the ratio of correctly classified cost-related sentences to all sentences that were classified as cost-related, whether correctly or incorrectly.

Recall is the ratio of correctly classified cost-related sentences to the summation of actual cost-related sentences, which includes both correctly and incorrectly classified cost-related sentences.

Fscore is the harmonic mean of Precision and Recall, providing a balance between them.

Specificity is a measure of a model's ability to correctly identify negative cases. It reflects the accuracy of the model in excluding non-relevant instances.

AUC (Area Under the Curve) represents the measure of a model's ability to distinguish between classes. It combines both sensitivity and specificity, giving an overall performance metric of the classifier.

On the other hand, there are more evaluation criteria for multilabel classification. (Bogatinovski et al., 2022) gathered evaluation criteria into 3 different groups, namely Bipartition-based, Ranking-based, and Efficiency. Bipartition-based also separates into two, as shown is Figure 6-1:

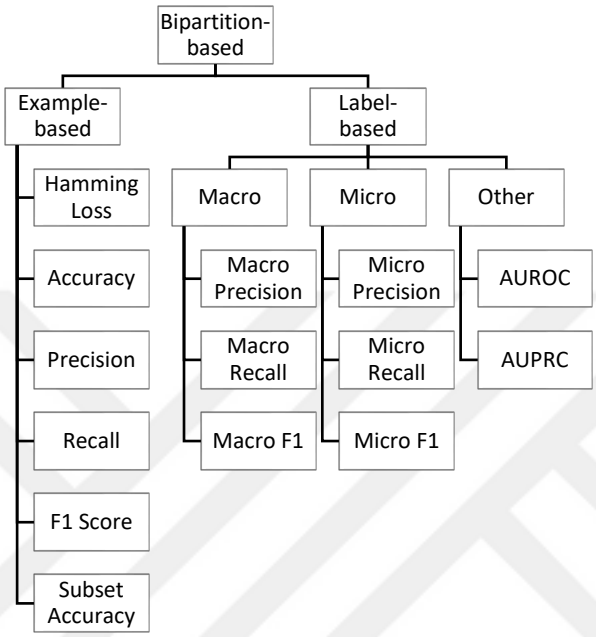


Figure 6-1 Multi-label Classification Metrics

A short summary of the evaluation criteria of example-based, macro and micro.

- **Hamming Loss** is one of the most common performance measures in Multi Label Classification, which is calculated as the symmetric difference between the predicted and true labels and divided by the total number of labels in the dataset.
- **Subset Accuracy** is the percentage of correctly predicted labels among all predicted and true labels.
- **Accuracy, Precision, Recall a**nd **F1 Score** are explained in the previous section.

Label-based measures assess each label individually and then average the results across all labels, resembling binary classification (Tarekegn et al., 2021). These

measures for all labels can be computed using two averaging methods: macro-averaging and micro-averaging (Yang, 1999).

In this study, accuracy, precision, recall, and F1 score are chosen for binary classification for the following reasons:

- Accuracy provides a straightforward measure of overall performance, and this metric is intuitive and easy to understand.
- Precision is crucial when the proportion of false positives is high, therefore, it ensures the model's predictions are reliable.
- Recall is important when the proportion of false negatives is high, thus, it ensures the model captures all relevant sentences.
- F1 score is useful because it combines precision and recall into a single metric, providing a balanced measure of performance.

On the other hand, accuracy, F1 score (micro), and F1 score (macro) are used for multi-label classification for the following reasons:

- Accuracy considers the exact match ratio in multi-label classification, providing how often the model's predicted labels match the true labels. This is a stringent and informative measure.
- F1 score (micro) sums the contributions of all labels and treats each prediction equally, making it useful for handling imbalanced datasets by giving a balanced view of the model's performance across all classes. This is also important for this study because all classes are not the same number-wise as in binary classification.
- F1 score (macro) calculates the F1 score for each label and averages them. It gives a more comprehensive evaluation of the model.

By focusing on these chosen metrics, it aims to provide a clear, balanced and comprehensive evaluation of the model's performance in both binary and multi-label classification tasks.

6.2 Classification Results and Interpretation of the Results

Since this study includes a comparison between different ML and deep learning algorithms, different codes have been used to work with all the algorithms. Consequently, it is essential to provide the same training and validation datasets to all algorithms to get reliable and consistent results. To do so, a random seed was set to guarantee the reproducibility of the dataset. This ensures that the dataset is split into training and validation sets identically in each algorithm. The following code snippet demonstrates this process:

"# Set random seed for reproducibility

SEED = 42

 # Split dataset into training and validation sets

train_sentences, val_sentences, train_labels, val_labels = train_test_split(

df['Text'], df['CostRelated'], test_size=0.2, random_state=SEED

)"

By doing that and the non-deterministic approach of classical ML algorithms, the results will always be same with the same coding approach.

6.2.1 Performance of Machine Learning Algorithms

As explained in Chapter 5, Support Vector Machine (SVM), Naive Bayes (NB), Logistic Regression (LR) and Random Forest (RF) were used with Bag-of-Words (BoW) and Term Frequency Inverse Document Frequency (TF-IDF) text representation techniques.

The steps for all machine learning algorithms are identical, as illustrated in Figure 6-2.

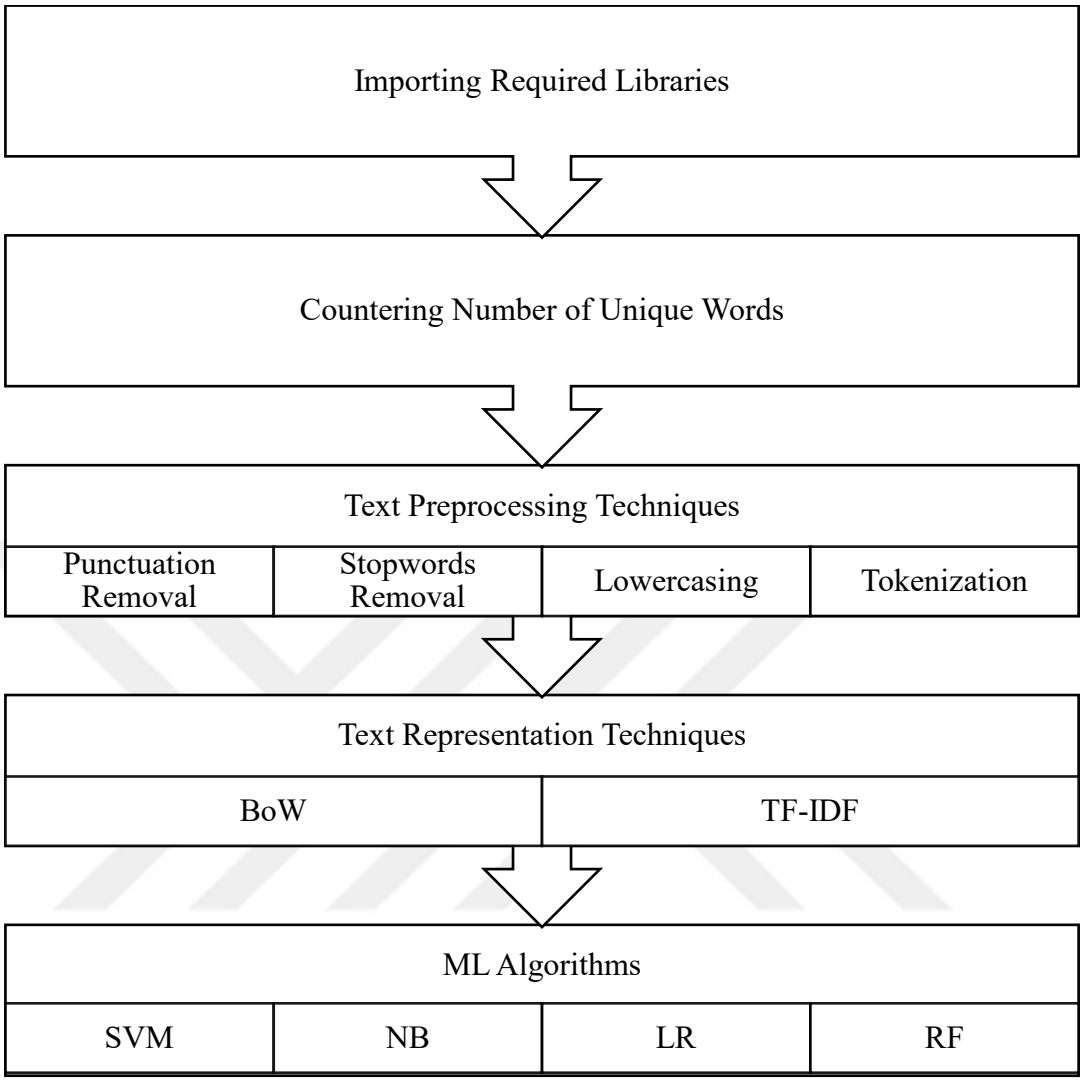


Figure 6-2 Machine Learning Classification Steps

The performance of the ML algorithms with different text representation techniques by displaying all the metrics such as accuracy, precision, recall F1 score is tabulated in Table 6-2:

Table 6-2 Performance of Machine Learning Algorithms

Method	Accuracy	Precision	Recall	F1-Score
SVM with BoW	0.8924	0.8992	0.8771	0.8880
SVM with TF-IDF	**0.9044**	**0.9298**	0.8689	0.8983
Naive Bayes with BoW	0.8885	0.8561	0.9262	0.8898
Naive Bayes with TF-IDF	0.8964	0.8582	**0.9426**	**0.8984**
Logistic Regression with BoW	0.8964	0.8810	0.9098	0.8952
Logistic Regression with TF-IDF	0.8845	0.8790	0.8934	0.8862
Random Forest with BoW	0.8765	0.8765	0.8765	0.8765
Random Forest with TF-IDF	0.8645	0.8647	0.8645	0.8646

The accuracy comparison of algorithms can be investigated in Figure 6-3.

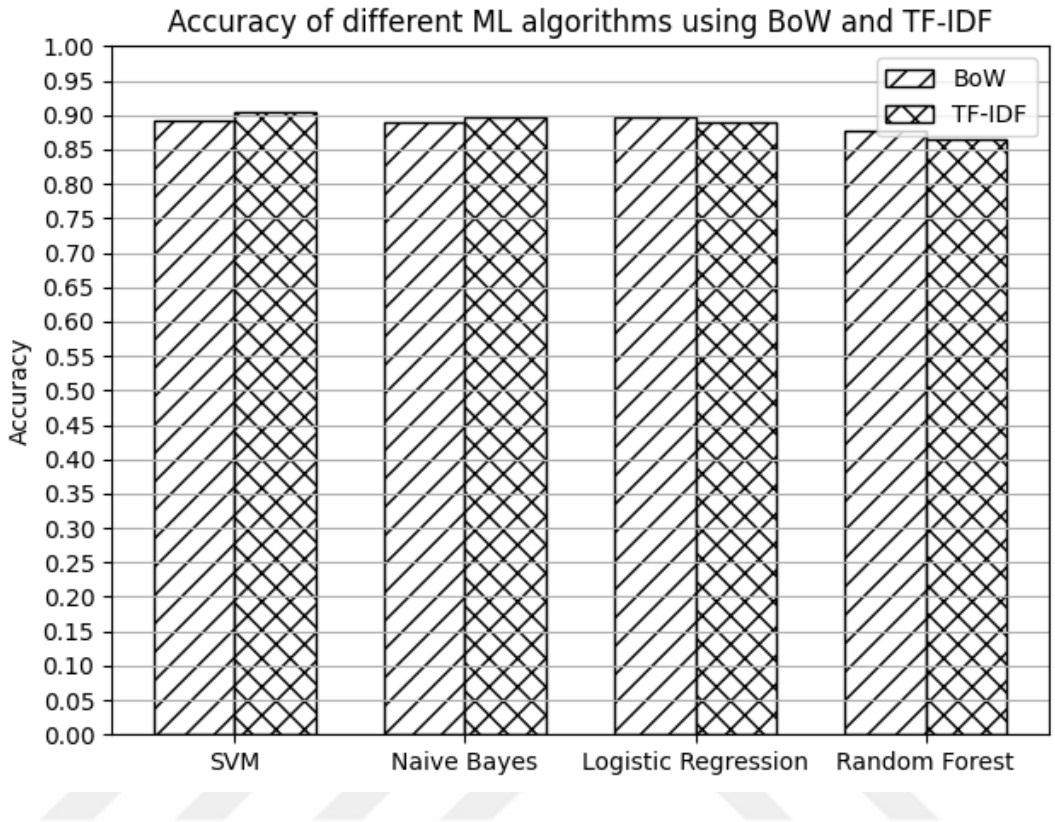


Figure 6-3 Accuracy of Different ML Algorithms using BoW and TF-IDF

The results obtained from the various machine learning algorithms using BoW and TF-IDF exhibit both similarities and differences. While the overall performance trends of the algorithms are comparable across these text representation techniques, distinct variations are also evident. The interpretation of these results and the general trends observed are explained in detail below:

- SVM performs well with BoW, showing high accuracy and precision. Therefore, the F1 score is also quite strong, indicating a good balance between precision and recall.
- SVM with TF-IDF performs slightly better than the BoW in terms of accuracy, precision and F1 score.
- NB with BoW has decent accuracy and a good F1 score. However, the precision is a bit lower compared to other models.

- NB with TF-IDF shows better performance compared to BoW, achieving higher accuracy and F1 score.
- LR with BoW achieves good accuracy and F1 score.
- LR with TF-IDF has slightly lower accuracy and F1 score compared to the BoW variant, but still performs well overall.
- RF with BoW has good precision but a slightly lower F1 score and accuracy compared to other models.
- RF with TF-IDF performs the worst among all models in terms of accuracy, precision, and F1 score, indicating that Random Forest does not benefit much from TF-IDF in this context.

The best-performing model is SVM with TF-IDF, with the highest accuracy and precision and a strong F1 score.

For the best performance on this dataset, SVM with TF-IDF is recommended. Logistic Regression and Naive Bayes are also strong contenders, with Logistic Regression performing well with BoW and Naive Bayes with TF-IDF. Random Forest, while still reasonably good, is less effective in this dataset.

6.2.2 Performance of Deep Learning Algorithms – Binary Classification

Deep Learning Algorithms such as CNN, RNN, LSTM, and Bi-LSTM can be processed using word embedding techniques such as Word2Vec and Glove. It should be emphasized that optimized hyperparameters were used in order to perform the classification tasks.

6.2.2.1 Word2Vec Results

Table 6-3 presents the performance metrics of machine learning algorithms utilizing Word2Vec embeddings.

Table 6-3 Performance of Algorithms with Word2Vec

RESULTS				
ALGORITHM	**Accuracy**	**Precision**	**Recall**	**F1 Score**
LSTM	0.9084	0.9091	0.9016	0.9053
Bi-LSTM	**0.9163**	**0.9106**	**0.9180**	**0.9143**
CNN	0.9084	0.8960	**0.9180**	0.9069
RNN	0.9084	0.9024	0.9098	0.9061

- Bi-LSTM is the best-performing model overall, with the highest accuracy, precision, and F1 score. It also ties in with CNN for the highest recall.
- CNN performs well in terms of recall but has lower precision and F1 score compared to Bi-LSTM.
- LSTM and RNN have similar performance, with slightly lower metrics compared to Bi-LSTM and CNN.

Given these results, Bi-LSTM appears to be the most effective model for this binary classification task using Word2Vec embeddings.

6.2.2.2 GloVe Results

Table 6-4 presents the performance metrics of machine learning algorithms utilizing GloVe embeddings.

Table 6-4 Performance of Algorithms with GloVe

RESULTS				
ALGORITHM	**Accuracy**	**Precision**	**Recall**	**F1 Score**
LSTM	0.9124	0.9310	0.8852	0.9076
BI-LSTM	**0.9203**	**0.9322**	**0.9016**	**0.9167**
CNN	0.9124	0.9237	0.8934	0.9083
RNN	0.9004	0.9008	0.8934	0.8971

- Bi-LSTM is the best-performing model overall, with the highest accuracy (0.9203), precision (0.9322), and F1 score (0.9167). It also has the highest recall (0.9016).
- CNN performs well in terms of precision (0.9237) and F1 score (0.9083), but it has a lower recall (0.8934) compared to Bi-LSTM.
- LSTM also shows strong performance, with high precision (0.9310) and a competitive F1 score (0.9076), but its recall (0.8852) is lower than Bi-LSTM and CNN.
- RNN has the lowest accuracy (0.9004) among the models and lower precision (0.9008), recall (0.8934), and F1 score (0.8971) compared to the other models.

Given these results, Bi-LSTM appears to be the most effective model for this binary classification task using GloVe embeddings.

6.2.2.3 Comparison of Word Embedding Techniques

Different types of word embedding techniques may give different results based on the data available, hyperparameters, and the environment (computer, etc.). It should be noted that different types of parameters have been employed for different algorithms by optimizing the hyperparameters with BOHB and that same dataset has been used in order to obtain reliable results from the algorithms.

Moreover, it should also be indicated that although hyperparameters vary through algorithms and word embedding techniques, the dimension was set as 200d for both Word2Vec and GloVe.

For the LSTM model, both GloVe and Word2Vec embeddings produce nearly identical performance metrics. Hence, it demonstrates that the LSTM model's performance is not significantly affected by the choice of word embeddings in this dataset and case. However, GloVe is better than Word2Vec in terms of all metrics except recall.

The Bi-LSTM model performs slightly better with GloVe embeddings in terms of accuracy, precision, and F1 score, while Word2Vec provides better recall. This indicates that GloVe embeddings might be capturing more relevant context for precision, while Word2Vec slightly enhances the model's ability to identify true positives.

The CNN model performs better with GloVe embeddings in terms of accuracy, precision, and F1 score, while Word2Vec provides better recall. This suggests that GloVe embeddings help the model to better differentiate between classes, whereas Word2Vec helps in detecting more true positives.

The RNN model performs better with Word2Vec embeddings across most metrics, particularly in terms of accuracy, recall, and F1 score. This indicates that Word2Vec embeddings may be better suited for RNNs in capturing the necessary context for classification.

Overall, GloVe provides slightly better performance with Bi-LSTM and CNN models, particularly in terms of precision and F1 score. This suggests that GloVe embeddings might be more effective in capturing context that improves the differentiation between classes. Word2Vec generally provides better recall across models and better overall performance with the RNN model. This indicates that Word2Vec embeddings may help in identifying more true positives, which is particularly beneficial for models like RNNs.

6.2.3 Performance of BERT-Binary Classification

The performance of BERT is measured by using the same metrics for binary classification. The results of the performance are tabulated in Table 6-5.

Table 6-5 Performance Metrics of BERT

Accuracy	**0.9243** – *(rounded)*
Precision	0.9328 – *(rounded)*
Recall	0.9098 – *(rounded)*
F1-Score	0.9212 – *(rounded)*

An accuracy of 0.924 means that the model correctly classified approximately 92% of the total sentences (both true positives and true negatives). On the other side, that value of precision shows that out of all the sentences the model predicted as positive, approximately 93% were actually positive. It also demonstrates the model's ability to avoid false positives.

Recall is the proportion of true positive predictions out of all actual positive instances. The significance of recall in this context lies in its ability to ensure that all indirect-cost-related clauses are extracted from the tender documentation, regardless of their correct labeling. If a clause is correctly labeled, it is beneficial; however, even if it is incorrectly labeled, the tender engineer can still review and potentially disregard it. Thus, achieving a high recall is essential to minimize the risk of missing relevant clauses.

An F1-score of 0.92 indicates that the model has a good balance between the other metrics, meaning it effectively identifies positives while minimizing false positives and false negatives.

The confusion matrix illustrating the performance of the BERT model is presented in Figure 6-4.

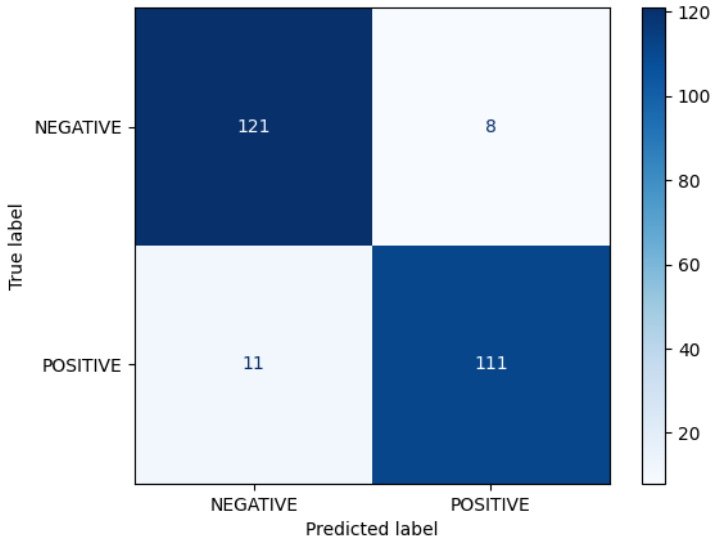


Figure 6-4 Confusion Matrix - BERT Performance

When comparing classical ML algorithms, deep learning algorithms, and BERT, it is evident that BERT outperforms the other approaches due to its state-of-the-art architecture and pre-trained model structure in terms of accuracy and F1 score. Notably, BERT achieved this performance utilizing only 10 epochs, whereas the other deep learning algorithms were trained for 100 epochs. Consequently, the BERT classifier has been selected for the multi-label classification task.

6.2.4 Performance of BERT-Multilabel Classification

This sub-section presents the performance of the BERT on multilabel classification based on the indirect-cost-related clauses in the dataset. As explained in the Dataset Chapter, there are a total of 1252 sentences divided equally between cost-related and non-cost-related. Therefore, the dataset that is used in the classification task consists of 626 sentences and 504 sentences and 122 sentences have been classified as train sentences and validation sentences, respectively.

The algorithm was trained with 5 epochs.

The results of the multi-label classification are presented in Table 6-6.

Table 6-6 Multi-Lable Classification Performance - BERT

Metric	Result
Accuracy	0.8934 – *(rounded)*
F1 Score (Micro)	0.9358 – *(rounded)*
F1 Score (Macro)	0.9265 – *(rounded)*

Accuracy in the context of multi-label classification is the subset accuracy, which means that it considers a prediction to be correct only if all the predicted labels for a given instance match the true labels. An accuracy of approximately 0.893 means that 89.3% of the predictions made by BERT are exactly correct for all labels.

The micro F1 score is calculated by considering the total true positives, false positives, and false negatives across all labels. It is a measure that gives equal weight to each class, treating all label predictions as equally important. A micro F1 score of approximately 0.936 indicates a high overall performance in terms of precision and recall across all labels.

The macro F1 score is the arithmetic mean of the F1 scores for each label. It treats each label equally, regardless of how many instances of that label appear in the dataset. A macro F1 score of approximately 0.926 indicates that, on average, the model performs well across all labels, though it may be affected by labels with fewer instances.

The multi-label classification model shows strong performance based on these metrics. The high accuracy suggests that the model often predicts the correct set of labels for each instance. The high micro and macro F1 scores indicate that the model balances precision and recall effectively across all labels, with the macro F1 score demonstrating good average performance across individual labels.

The slight difference between the micro and macro F1 scores suggests variability in performance across different labels, indicating potential data imbalance. This

difference arises because the macro F1 score treats each label equally, whereas the micro F1 score aggregates the contributions of all labels, potentially giving more weight to more frequent labels.

To better understand and address this variability, further study may be performed to analyze the impact of class balancing on model performance. Addressing data imbalance can help improve the model's performance across all labels, leading to a more robust and reliable classification system.

6.2.5 Analysis of Wrongly-Labeled Sentences: Binary

This sub-section presents a detailed analysis of the misclassified sentences produced by deep learning algorithms when using Word2Vec, GloVe embeddings, and BERT.

The objective is to identify patterns and categorize the types of frequently misclassified sentences, providing insights into the limitations and challenges these models face. The main reasons for the wrong classification are as follows:

- It has been observed that the algorithms may not always perfectly align with the user's specific requirements. For instance, during sentence classification, clauses related to any hard copy report submission and any language other than English should be categorized and classified as indirect-cost-related clauses because they will affect personal needs such as translators and office equipment such as readers and printers. However, the algorithms did not consistently classify all sentences within this category as expected, with the rate of misclassification reaching 19% for Word2Vec word embedding and 26.25% for GloVe embedding.
- It has been observed that words with a strong potential to indicate indirect-cost-related clauses can significantly influence sentence classification results. For instance, the presence of the word "insurance" in a sentence tends to increase the likelihood of it being classified as an indirect-cost-related clause, more so than other sentences within the Terms of Reference and/or

Scope of Work. Consequently, the inclusion of such words can sometimes alter the classification outcome. Additionally, the duration of services is also recognized as impacting indirect costs; therefore, the presence of the word "duration" similarly affects the classification results.

- It has been observed that proper nouns and formulas affect the classification result, including cost adjustment formulas and location names.
- Certain concepts do not frequently recur in the tender documentation, making it challenging to ascertain whether they pertain to indirect costs. An example of this is the requirement for licenses. License requirements in tender documentation are generally not prominently featured and often pertain to specific laws of the country where the services are to be provided. Consequently, the algorithms may struggle to extract usable information from these sentences.

Upon careful analysis of the reasons for mislabeling, it can be concluded that larger datasets can significantly enhance algorithm performance. Comparing algorithm performance with larger datasets would provide valuable insights.

Additionally, the likelihood of a sentence falling into different concepts varies from one tender document to another and is influenced by factors such as the specific tender, geography, and other contextual elements. Therefore, ensuring that the dataset has sufficient diversity to encompass all potential scenarios in consultancy tenders is crucial.

As outlined in the Dataset Chapter, this study's dataset demonstrates considerable diversity regarding project scope, client, geography, funders, and other relevant factors. However, expanding the dataset further could potentially improve performance.

Additionally, certain contexts may be extracted or considered based on the user's preference. However, it is essential to ensure that there are enough sentences to adequately train the algorithms, ensuring reliable and consistent results.

Another piece of information that can be extracted from the misclassified sentences is the type of sentences mislabeled when using different algorithms and word embedding techniques. It is more accurate to say that similar and recurring misclassified sentences are present across all combinations. However, the word embedding technique significantly affects the similarity ratio of misclassified sentences. When the deep learning technique changes, the likelihood of similar sentences being misclassified increases. On the other hand, although some of the sentences that have been classified wrongly are similar, there is no obvious similarity pattern between the BERT classifier and deep learning algorithms.

CHAPTER 7

CONCLUSIONS

In this study, a comprehensive approach was undertaken to extract and classify indirect cost-related clauses from construction consultancy tender documents using both classical ML and deep learning algorithms. The primary objective was to identify and categorize specific clauses and/or sentences related to indirect costs, often critical in construction projects.

To perform the classification task, 28 construction consultancy tender documents were read, and indirect cost-related clauses were manually extracted. On the other hand, non-cost-related clauses were also selected arbitrarily from the tender documents, ensuring an equal number of cost-related and non-cost-related sentences. Notably, the 28 tenders varied in terms of sector, geography, funders, and clients, enhancing the study's robustness. Lastly, the labeling process for multi-label classification and verification was conducted with precision to ensure data integrity.

Several text representation techniques were employed to transform the raw text data into formats suitable for sentence classification. These techniques included traditional methods like BoW and TF-IDF, as well as more advanced word embedding methods such as Word2Vec and GloVe. The variety in text representation ensured a robust analysis of comparison by capturing different aspects of the textual data.

The study employed a range of classical ML and deep learning algorithms from Support Vector Machines to BERT.

For binary classification, classical ML algorithms were first tested. For this specific task, Support Vector Machine (SVM) with TF-IDF outperformed other method-text representation techniques regarding accuracy and precision, similar to the findings of Hassan & Le (2020). Additionally, the best precision and recall were obtained

with support vector classifiers, as demonstrated by Candaş & Tokdemir (2022). On the other hand, Naïve Bayes with TF-IDF showed superior performance than the different combinations in terms of recall and F1-score in this study.

For Deep Learning algorithms, the best results were identical across both Word2Vec and GloVe word embedding techniques. In both techniques, bi-LSTM demonstrated better performance across all metrics used in the classification. Only CNN showed the same performance in recall metric in Word2vec embedding. Jeon et al. (2021) demonstrated that the CNN+GloVe combination led to the highest accuracy in their study, which combined CNN and RNN with GloVe and Word2Vec. Similarly, in this study, CNN with GloVe yielded better performance than other combinations.

Lastly, BERT was used to perform the binary classification task, and it showed the best result compared to other techniques and combinations, achieving an accuracy of 92,4% due to its pre-trained corpus. Pham & Han (2023) also found that fine-tuned BERT demonstrated the best accuracy among classical machine learning algorithms and GloVe, Word2Vec, and Classic BoW combinations.

Therefore, the BERT model was selected to perform multilabel classification. BERT, being a state-of-the-art transformer model, is highly effective in handling the intricacies of language and context, making it well-suited for multilabel classification tasks where multiple labels may be assigned to a single text instance. The labels in the multi-label classification are divided into four groups consisting of different indirect cost items, from the accommodation of the experts to the taxes and price adjustments. Therefore, the study aimed to label the sentences correctly into one or more than one group.

BERT successfully performed this task, demonstrating strong performance with 89.3% accuracy, a 93.6% F1 score (micro), and a 92.6% F1 score (macro).

The performance of these algorithms was meticulously compared using various evaluation metrics. The results of the comparisons were carefully interpreted to provide insights into the strengths and weaknesses of each approach. This

comprehensive analysis not only highlighted the efficacy of different models in handling the task but also provided practical recommendations for future applications in similar domains.

This study offers several practical implications for the construction consultancy tender process, emphasizing methodological enhancements and operational efficiencies. Firstly, this study demonstrates a robust methodology for extracting and classifying indirect cost-related clauses by employing advanced text representation techniques and a range of machine learning and deep learning algorithms. Such a method ensures that relevant clauses are accurately identified, which is critical for cost management and budgeting in construction consultancy tenders. However, while this methodology may not completely replace the expert's perspective, especially without achieving 100% accuracy, it can serve as a valuable supporting tool. Moreover, in tight deadlines, this approach offers a practical solution by speeding up the review process, and it can be used with a higher degree of contingency.

From a methodological perspective, the application of BERT for both binary and multi-label classification tasks has proven to be highly effective. BERT's superior performance, particularly in handling complex language and context, suggests that it can be an invaluable tool for tender departments. Specifically, using BERT can streamline the review process of tender documents, ensuring that indirect cost-related clauses are accurately classified, thereby meeting client demands with greater precision.

The findings from this study can be directly applied to enhance the operations of tender departments in several key ways:

Meeting Client Demands: By leveraging advanced classification models like BERT, tender departments can ensure that their proposals are aligned with client requirements and can ensure that they do not miss any indirect cost-related clause and/or item. The precise extraction and classification of cost-related clauses enable departments to address specific client needs more effectively.

Accurate Financial Proposals: The accurate identification of indirect cost-related clauses ensures that all relevant costs are accounted for in financial proposals. This accuracy leads to more reliable and realistic financial planning, which can enhance the credibility and competitiveness of the consultancy.

Reduction in Time and Effort: The automation of clause extraction and classification significantly reduces the time required to review tender documents. This efficiency not only speeds up the tender preparation process but also allows human resources to be allocated to more strategic tasks, thereby improving overall productivity.

In summary, this study demonstrates the substantial benefits of integrating advanced machine learning and deep learning techniques into the tender document review process. By adopting these methods, tender departments can enhance their accuracy, efficiency, and responsiveness to client demands, thereby gaining a competitive edge in the construction consultancy industry.

Although 28 different construction consultancy tenders were analyzed, the inclusion of more tenders could increase the reliability and accuracy of the binary and multi-label classification tasks. More data could also allow for an increased number of classes in the multi-label classification, as more sentences would be available for each label. Furthermore, with an increased number of classes, user requirements could be defined and identified in a more detailed and comprehensive manner. Hence, future research can use a larger dataset.

Additionally, tables from the tender documents were excluded, and only sentences were included in the dataset. Future models could be developed to process information in tables as well.

Furthermore, this study was conducted exclusively using tender documents in English. Incorporating non-English tender documents in future research would be beneficial to enhance the generalizability of the models and improve their accuracy on unseen data.

Even though this study focuses on construction consultancy tenders, the same codes and models can be used in construction tenders if there is enough data to train and test the algorithms. Therefore, a different approach may be developed in order to extract and classify the sentences and/or clauses from the construction tenders.

Overall, the study demonstrated a thorough exploration of text representation and classification techniques, contributing valuable knowledge to the field of construction consultancy document analysis.

REFERENCES

Abdolahi, M., & Zahedh, M. (2017). Sentence matrix normalization using most likely n-grams vector. *2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation, KBEI 2017*, *2018-January*, 0040–0045. https://doi.org/10.1109/KBEI.2017.8325018

Babar, S. A., & Patil, P. D. (2015). Improving Performance of Text Summarization. *Procedia Computer Science*, *46*, 354–363. https://doi.org/10.1016/J.PROCS.2015.02.031

Bergstra, J., Ca, J. B., & Ca, Y. B. (2012). Random Search for Hyper-Parameter Optimization Yoshua Bengio. *Journal of Machine Learning Research*, *13*, 281–305. http://scikit-learn.sourceforge.net.

Bird, S., Klein, e., & Loper, E. (2009). *Natural Language Processing with Python: Analyzing Text with the natural language toolkit*.

Bogatinovski, J., Todorovski, L., Džeroski, S., & Kocev, D. (2022). Comprehensive comparative study of multi-label classification methods. *Expert Systems with Applications*, *203*, 117215. https://doi.org/10.1016/J.ESWA.2022.117215

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

Caldas, C. H., Soibelman, L., & Han, J. (2002). Automated Classification of Construction Project Documents. *Journal of Computing in Civil Engineering*, *16*(4), 234–243. https://doi.org/10.1061/(ASCE)0887-3801(2002)16:4(234)

Campbell, W. M., Li, L., Dagli, C., Acevedo-Aviles, J., Geyer, K., Campbell, J. P., & Priebe, C. (2016). *Cross-Domain Entity Resolution in Social Media*.

Candaş, A. B., & Tokdemir, O. B. (2022). Automating Coordination Efforts for Reviewing Construction Contracts with Multilabel Text Classification. *Journal of Construction Engineering and Management*, *148*(6), 04022027. https://doi.org/https://doi.org/10.1061/(ASCE)CO.1943-7862.0002275

Chai, C. P. (2023). Comparison of text preprocessing methods. *Natural Language Engineering*, *29*(3), 509–553. https://doi.org/10.1017/S1351324922000213

Chawla, N. V., Japkowicz, N., & Kotcz, A. (2004). Editorial. *ACM SIGKDD Explorations Newsletter*, *6*(1), 1–6. https://doi.org/10.1145/1007730.1007733

Dauphin, Y. N., Fan, A., Auli, M., & Grangier, D. (2017). *Language Modeling with Gated Convolutional Networks*.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). 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*, *1*, 4171–4186. https://arxiv.org/abs/1810.04805v2

Dharma, E. M., Lumban Gaol, F., Leslie, H., Warnars, H. S., & Soewito, B. (2022). The accuracy comparison among word2vec, glove, and fasttext towards convolution neural network (cnn) text classification. *Journal of Theoretical and Applied Information Technology*, *31*(2). www.jatit.org

Ding, Y., Ma, J., & Luo, X. (2022). Applications of natural language processing in construction. *Automation in Construction*, *136*, 104169. https://doi.org/10.1016/J.AUTCON.2022.104169

Do, Q., Moriyani, M. A., Le, C., & Le, T. (2023). Cost-Weighted TF-IDF: A Novel Approach for Measuring Highway Project Similarity Based on Pay Items' Cost Composition and Term Frequency. *Journal of Construction Engineering and Management*, *149*(8), 04023069. https://doi.org/10.1061/JCEMD4.COENG-13023/ASSET/FCC4F27B-D6DC-4386-B5F7-071F740945E5/ASSETS/IMAGES/LARGE/FIGURE9.JPG

Dong, C., Li, Y., Gong, H., Chen, M., Li, J., Shen, Y., & Yang, M. (2021). A Survey of Natural Language Generation. *ACM Computing Surveys*, *55*(8), 38. https://doi.org/10.1145/3554727

Eastman, C., Lee, J. min, Jeong, Y. suk, & Lee, J. kook. (2009). Automatic rule-based checking of building designs. *Automation in Construction*, *18*(8), 1011–1033. https://doi.org/10.1016/J.AUTCON.2009.07.002

Egger, R. (2022). Text Representations and Word Embeddings. *Tourism on the Verge*, *Part F1051*, 335–361. https://doi.org/10.1007/978-3-030-88389-8_16

Eken, G. (2022). *Using natural language processing for automated construction contract review during risk assessment at the bidding stage*. https://open.metu.edu.tr/handle/11511/101793

Elming, J., Johannsen, A., Klerke, S., Lapponi, E., Martinez, H., & Søgaard, A. (2013). *Down-stream effects of tree-to-dependency conversions*. 9–14. http://nlp.cs.lth.se/software/treebank

Elshawi, R., Maher, M., & Sakr, S. (2019). *Automated Machine Learning: State-of-The-Art and Open Challenges*. http://arxiv.org/abs/1906.02287

F. Boukamp. (2006). *Modeling of and reasoning about construction specifications to support automated defect detection*. https://www.proquest.com/docview/305353632?pq-origsite=gscholar&fromopenview=true

Falkner, S., Klein, A., & Hutter, F. (2018). *BOHB: Robust and Efficient Hyperparameter Optimization at Scale* (pp. 1437–1446). PMLR. https://proceedings.mlr.press/v80/falkner18a.html

Fan, H., & Li, H. (2013). Retrieving similar cases for alternative dispute resolution in construction accidents using text mining techniques. *Automation in Construction*, *34*, 85–91. https://doi.org/10.1016/J.AUTCON.2012.10.014

Ferilli, S., Esposito, F., & Grieco, D. (2014). Automatic Learning of Linguistic Resources for Stopword Removal and Stemming from Text. *Procedia Computer Science*, *38*(C), 116–123. https://doi.org/10.1016/J.PROCS.2014.10.019

Fernández, A., García, S., Luengo, J., Bernadó-Mansilla, E., & Herrera, F. (2010). Genetics-based machine learning for rule induction: State of the art, taxonomy, and comparative study. *IEEE Transactions on Evolutionary Computation*, *14*(6), 913–941. https://doi.org/10.1109/TEVC.2009.2039140

Fernández, A., López, V., Galar, M., Del Jesus, M. J., & Herrera, F. (2013). Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-Based Systems*, *42*, 97–110. https://doi.org/10.1016/J.KNOSYS.2013.01.018

Frank, E., Oluwaseyi, J., Olaoye, G. O., & Olaoye, G. (2024). *Data preprocessing techniques for NLP in BI*. https://www.researchgate.net/publication/379652291

Frazier, P. I. (2018). *A Tutorial on Bayesian Optimization*. https://arxiv.org/abs/1807.02811v1

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F. (2012). A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, *42*(4), 463–484. https://doi.org/10.1109/TSMCC.2011.2161285

George, S., & Joseph, S. (2014). *Text Classification by Augmenting Bag of Words (BOW) Representation with Co-occurrence Feature*. *1*, 34–38. www.iosrjournals.org

Goldberg, Y., Levy, O., Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2014). *word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method*. https://arxiv.org/abs/1402.3722v1

Gomes, F. B., Adán-Coello, J. M., & Kintschner, F. E. (2018). Studying the effects of text preprocessing and ensemble methods on sentiment analysis of brazilian portuguese tweets. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11171 LNAI*, 167–177. https://doi.org/https://doi.org/10.1007/978-3-030-00810-9_15

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5–6), 602–610. https://doi.org/10.1016/J.NEUNET.2005.06.042

Grefenstette, G. (1999). *Tokenization*. 117–133. https://doi.org/10.1007/978-94-015-9273-4_9

HaCohen-Kerner, Y., Miller, D., & Yigal, Y. (2020). The influence of preprocessing on text classification using a bag-of-words representation. *PLOS ONE*, *15*(5), e0232525. https://doi.org/10.1371/JOURNAL.PONE.0232525

Hassan, F. ul, & 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), 04520009. https://doi.org/https://doi.org/10.1061/(ASCE)LA.1943-4170.0000379

Hassan, F. ul, & Le, T. (2021). Computer-assisted separation of design-build contract requirements to support subcontract drafting. *Automation in Construction*, *122*, 103479. https://doi.org/10.1016/J.AUTCON.2020.103479

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263–1284. https://doi.org/10.1109/TKDE.2008.239

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/NECO.1997.9.8.1735

Horn, N., Erhardt, M. S., Di Stefano, M., Bosten, F., & Buchkremer, R. (2020). *Vergleichende Analyse der Word-Embedding-Verfahren Word2Vec und GloVe am Beispiel von Kundenbewertungen eines Online-Versandhändlers*. 559–581. https://doi.org/10.1007/978-3-658-29550-9_29

Huang, Z., Research, B., Xu, W., & Baidu, K. Y. (2015). *Bidirectional LSTM-CRF Models for Sequence Tagging*. https://arxiv.org/abs/1508.01991v1

Hutter, F., Kotthoff, L., & Vanschoren, J. (Eds.). (2019). *Automated Machine Learning*. https://doi.org/10.1007/978-3-030-05318-5

Jafari, P., Al Hattab, M., Mohamed, E., & Abourizk, S. (2021). Automated extraction and time-cost prediction of contractual reporting requirements in construction using natural language processing and simulation. *Applied Sciences (Switzerland)*, *11*(13). https://doi.org/10.3390/APP11136188

Jang, B., Kim, I., & Kim, J. W. (2019). Word2vec convolutional neural networks for classification of news articles and tweets. *PLOS ONE*, *14*(8), e0220976. https://doi.org/10.1371/JOURNAL.PONE.0220976

Jeon, J., Xu, X., Zhang, Y., Yang, L., & Cai, H. (2021). Extraction of construction quality requirements from textual specifications via natural language processing. *Transportation Research Record*, *2675*(9), 222–237. https://doi.org/10.1177/03611981211001385

Jianan, G., Kehao, R., Research, G. B.-J. of E., & 2024, undefined. (2024). Deep learning-based text knowledge classification for whole-process engineering consulting standards. *Journal of Engineering Research*, *12*(2), 61–71. https://www.sciencedirect.com/science/article/pii/S2307187723001724

Johnson, S. J., Murty, M. R., & Navakanth, I. (2024). A detailed review on word embedding techniques with emphasis on word2vec. *Multimedia Tools and Applications*, *83*(13), 37979–38007. https://doi.org/10.1007/S11042-023-17007-Z

Jurafsky, D., & Martin, J. H. (2000). *Speech and language processing*. https://www.thuvienso.net/upload/library/204335/preview/ebook-speech-and-language-processing-an-introduction-to-natural-language-processing-computational-linguistics-and-speech-recognition-part-1-daniel-jur.pdf

Kadhim, A. (2018). An Evaluation of Preprocessing Techniques for Text Classification. *Article in International Journal of Computer Science and Information Security*. https://sites.google.com/site/ijcsis/

Kandel, I., & Castelli, M. (2020). The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, *6*(4), 312–315. https://doi.org/10.1016/J.ICTE.2020.04.010

Khalef, R., & El-adaway, I. H. (2021). Automated Identification of Substantial Changes in Construction Projects of Airport Improvement Program: Machine Learning and Natural Language Processing Comparative Analysis. *Journal of Management in Engineering*, *37*(6). https://doi.org/10.1061/(ASCE)ME.1943-5479.0000959

Kohavi, R., & John, G. H. (1995). Automatic Parameter Selection by Minimizing Estimated Error. *Proceedings of the 12th International Conference on Machine Learning, ICML 1995*, 304–312. https://doi.org/10.1016/B978-1-55860-377-6.50045-1

Koto, F., & Adriani, M. (2015). A comparative study on twitter sentiment analysis: Which features are good? *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9103*, 453–457. https://doi.org/10.1007/978-3-319-19581-0_46/TABLES/4

K.R. Chowdhary. (2020). *Fundamentals of Artificial Intelligence*. https://doi.org/10.1007/978-81-322-3972-7

Kwartler, T. (2017). *Text mining in practice with R*. https://books.google.com/books/about/Text_Mining_in_Practice_with_R.html?hl=tr&id=2KxaDwAAQBAJ

Lee, J., Ham, Y., Yi, J.-S., & Son, J. (2020). Effective Risk Positioning through Automated Identification of Missing Contract Conditions from the Contractor's Perspective Based on FIDIC Contract Cases. *Journal of Management in Engineering*, *36*(3), 05020003. https://doi.org/https://doi.org/10.1061/(ASCE)ME.1943-5479.0000757

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), 04019003. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000807

Li, J., Sun, A., Han, J., & Li, C. (2022). A Survey on Deep Learning for Named Entity Recognition. *IEEE Transactions on Knowledge and Data Engineering*, *34*(1), 50–70. https://doi.org/10.1109/TKDE.2020.2981314

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2018). Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research*, *18*(185), 1–52. http://jmlr.org/papers/v18/16-558.html

Lin, M., Tang, K., & Yao, X. (2013). Dynamic sampling approach to training neural networks for multiclass imbalance classification. *IEEE Transactions on Neural Networks and Learning Systems*, *24*(4), 647–660. https://doi.org/10.1109/TNNLS.2012.2228231

Liu, C. Z., Sheng, Y. X., Wei, Z. Q., & Yang, Y. Q. (2018). Research of Text Classification Based on Improved TF-IDF Algorithm. *2018 IEEE International Conference of Intelligent Robotic and Control Engineering, IRCE 2018*, 69–73. https://doi.org/10.1109/IRCE.2018.8492945

Liu, J., Luo, H., & Liu, H. (2022). Deep learning-based data analytics for safety in construction. *Automation in Construction*, *140*, 104302. https://doi.org/10.1016/J.AUTCON.2022.104302

López, V., Fernández, A., García, S., Palade, V., & Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, *250*, 113–141. https://doi.org/10.1016/J.INS.2013.07.007

Luhn, H. P. (1960). Key word-in-context index for technical literature (kwic index). *American Documentation*, *11*(4), 288–295. https://doi.org/10.1002/ASI.5090110403

Manning, C. D., Schütze, H., & Raghavan, P. (2008). *Introduction to Information Retrieval* (Vol. 39). www.cambridge.org/9780521865715

Mantovani, R. G., Horvath, T., Cerri, R., Vanschoren, J., & De Carvalho, A. C. P. L. F. (2017). Hyper-Parameter Tuning of a Decision Tree Induction Algorithm. *Proceedings - 2016 5th Brazilian Conference on Intelligent Systems, BRACIS 2016*, 37–42. https://doi.org/10.1109/BRACIS.2016.018

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, *5*(4), 1093–1113. https://doi.org/10.1016/J.ASEJ.2014.04.011

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

Mirus, F., Stewart, T. C., & Conradt, J. (2020). The Importance of Balanced Data Sets: Analyzing a Vehicle Trajectory Prediction Model based on Neural Networks and Distributed Representations. *Proceedings of the International Joint Conference on Neural Networks*. https://doi.org/10.1109/IJCNN48605.2020.9206627

Moon, S., Chi, S., & Im, S. B. (2022). Automated detection of contractual risk clauses from construction specifications using bidirectional encoder representations from transformers (BERT). *Automation in Construction*, *142*, 104465. https://doi.org/10.1016/J.AUTCON.2022.104465

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), 04020147. https://doi.org/https://doi.org/10.1061/(ASCE)CO.1943-7862.0001953

Nallapati, R., Zhou, B., dos Santos, C., Gulçehre, Ç., & Xiang, B. (2016). Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. *CoNLL 2016 - 20th SIGNLL Conference on Computational Natural Language Learning, Proceedings*, 280–290. https://doi.org/10.18653/v1/k16-1028

Olson, R. S., La Cava, W., Mustahsan, Z., Varik, A., & Moore, J. H. (2018). Data-driven advice for applying machine learning to bioinformatics problems. *Pacific Symposium on Biocomputing*, *0*(212669), 192–203. https://doi.org/10.1142/9789813235533_0018

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. *International Journal for Research in Applied Science and Engineering Technology*, *10*(12), 943–947. https://doi.org/10.22214/ijraset.2022.47789

Ozturk, S., Ozkaya, U., Akdemir, B., & Seyfi, L. (2018). Convolution kernel size effect on convolutional neural network in histopathological image processing applications. *2018 International Symposium on Fundamentals of Electrical Engineering, ISFEE 2018*. https://doi.org/10.1109/ISFEE.2018.8742484

Padhy, J., Jagannathan, M., & Kumar Delhi, V. S. (2021). Application of Natural Language Processing to Automatically Identify Exculpatory Clauses in Construction Contracts. *Journal of Legal Affairs and Dispute Resolution in Engineering and Construction*, *13*(4), 04521035. https://doi.org/https://doi.org/10.1061/(ASCE)LA.1943-4170.0000505

Pan, X., Zhong, B., Wang, X., & Xiang, R. (2021). Text mining-based patent analysis of bim application in construction. *Journal of Civil Engineering and Management*, *27*(5), 303–315. https://doi.org/10.3846/JCEM.2021.14907

Pham, H. T. T. L., & Han, S. (2023). Natural Language Processing with Multitask Classification for Semantic Prediction of Risk-Handling Actions in Construction Contracts. *Journal of Computing in Civil Engineering*, *37*(6), 04023027. https://doi.org/10.1061/JCCEE5.CPENG-5218

Poria, S., Agarwal, B., Gelbukh, A., Hussain, A., & Howard, N. (2014). Dependency-Based Semantic Parsing for Concept-Level Text Analysis. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8403 LNCS*(PART 1), 113–127. https://doi.org/10.1007/978-3-642-54906-9_10

Qader, W. A., Ameen, M. M., & Ahmed, B. I. (2019). An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges. *Proceedings of the 5th International Engineering Conference, IEC 2019*, 200–204. https://doi.org/10.1109/IEC47844.2019.8950616

Rajaraman, A., & Ullman, J. D. (2011). *Mining of Massive Datasets*. http://biblioteca.unisced.edu.mz/handle/123456789/2674

Saini, A., Khursheed, S., Paul, V. K., & Kumar, K. (2021). Analysis of Indirect cost of Construction Projects in India. *International Journal*. https://doi.org/10.37628/IJSBT

Saini, J. R., Raulji, J. K., Director, C., Supervisor, R., & Ambedkar, B. (2016). Stop-Word Removal Algorithm and its Implementation for Sanskrit Language. *Article in International Journal of Computer Applications*, *150*(2), 975–8887. https://doi.org/10.5120/ijca2016911462

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), 04014106. https://doi.org/https://doi.org/10.1061/(ASCE)CP.1943-5487.0000301

Salton, G., & Yu, C. T. (1973). On the construction of effective vocabularies for information retrieval. *ACM SIGPLAN Notices*, *10*(1), 48–60. https://doi.org/10.1145/951787.951766

Sanders, S., & Giraud-Carrier, C. (2017). Informing the use of hyperparameter optimization through metalearning. *Proceedings - IEEE International Conference on Data Mining, ICDM*, *2017-November*, 1051–1056. https://doi.org/10.1109/ICDM.2017.137

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681. https://doi.org/10.1109/78.650093

Shamshiri, A., Ryu, K. R., & Park, J. Y. (2024). Text mining and natural language processing in construction. *Automation in Construction*, *158*, 105200. https://doi.org/10.1016/J.AUTCON.2023.105200

Shin, S., & Issa, R. R. A. (2021). BIMASR: Framework for Voice-Based BIM Information Retrieval. *Journal of Construction Engineering and Management*, *147*(10), 04021124. https://doi.org/10.1061/(ASCE)CO.1943-7862.0002138

Smith, L. N. (2018). *A disciplined approach to neural network hyper-parameters: Part 1 -- learning rate, batch size, momentum, and weight decay*. https://arxiv.org/abs/1803.09820v2

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, *45*(4), 427–437. https://doi.org/10.1016/J.IPM.2009.03.002

Srivastava, N., Hinton, G., Krizhevsky, A., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.

Sun, Y., Wong, A. K. C., & Kamel, M. S. (2011). CLASSIFICATION OF IMBALANCED DATA: A REVIEW. *Https://Doi.Org/10.1142/S0218001409007326*, *23*(4), 687–719. https://doi.org/10.1142/S0218001409007326

Tarekegn, A. N., Giacobini, M., & Michalak, K. (2021). A review of methods for imbalanced multi-label classification. *Pattern Recognition*, *118*, 107965. https://doi.org/10.1016/J.PATCOG.2021.107965

Thakur, A., Ahuja, L., Vashisth, R., & Simon R. (2023, March). *NLP & AI Speech Recognition: An Analytical Review | IEEE Conference Publication | IEEE Xplore*. NLP & AI Speech Recognition: An Analytical Review. https://ieeexplore.ieee.org/document/10112511

Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2013). Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *Part F128815*, 847–855. https://doi.org/10.1145/2487575.2487629

Ul Hassan, F., Le, T., & Tran, D. H. (2020). Multi-Class Categorization of Design-Build Contract Requirements Using Text Mining and Natural Language Processing Techniques. *Construction Research Congress 2020: Project Management and Controls, Materials, and Contracts - Selected Papers from the Construction Research Congress 2020*, 1266–1274. https://doi.org/10.1061/9780784482889.135

Uysal, A. K., & Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing & Management*, *50*(1), 104–112. https://doi.org/10.1016/J.IPM.2013.08.006

Voorhees, E. M. (1999). *Natural Language Processing and Information Retrieval*. 32–48. https://doi.org/10.1007/3-540-48089-7_3

Walsh, K. P. (2017). Identifying and Mitigating the Risks Created by Problematic Clauses in Construction Contracts. *Journal of Legal Affairs and Dispute Resolution in Engineering and Construction*, *9*(3), 03717001. https://doi.org/10.1061/(ASCE)LA.1943-4170.0000225

Webb, G. I., & Bayes, N. (2010). Naïve Bayes. *Encyclopedia of Machine Learning*, *15*(1), 713–714. https://doi.org/10.1007/978-1-4899-7502-7_581-1

Weiss, G. M., & Tian, Y. (2008). Maximizing classifier utility when there are data acquisition and modeling costs. *Data Mining and Knowledge Discovery*, *17*(2), 253–282. https://doi.org/https://doi.org/10.1007/s10618-007-0082-x

Wendland, A., Zenere, M., & Niemann, J. (2021). Introduction to Text Classification: Impact of Stemming and Comparing TF-IDF and Count Vectorization as Feature Extraction Technique. *Communications in Computer and Information Science*, *1442*, 289–300. https://doi.org/https://doi.org/10.1007/978-3-030-85521-5_19

White, L., Togneri, R., Liu, W., & Bennamoun, M. (2018). *Neural Representations of Natural Language*. *783*. https://doi.org/10.1007/978-981-13-0062-2

Williams, T. P., & Gong, J. (2014). Predicting construction cost overruns using text mining, numerical data and ensemble classifiers. *Automation in Construction*, *43*, 23–29. https://doi.org/10.1016/J.AUTCON.2014.02.014

Wilson, D. R., & Martinez, T. R. (2003). The general inefficiency of batch training for gradient descent learning. *Neural Networks*, *16*(10), 1429–1451. https://doi.org/10.1016/S0893-6080(03)00138-2

Yaloveha, V., Podorozhniak, A., & Kuchuk, H. (2022). Convolutional neural network hyperparameter optimization applied to land cover classification. *Radioelectronic and Computer Systems*, *0*(1), 115–128. https://doi.org/10.32620/reks.2022.1.09

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval*, *1*(1–2), 69–90. https://doi.org/10.1023/A:1009982220290/METRICS

Yin, W., Kann, K., Yu, M., SchützeSch, H., & Munich, L. (2017). *Comparative Study of CNN and RNN for Natural Language Processing*. https://arxiv.org/abs/1702.01923v1

Ying, X. (2019). *An Overview of Overfitting and its Solutions*. 22022. https://doi.org/10.1088/1742-6596/1168/2/022022

Yu, W., Li, X., & Gonzalez, J. (2019). Fast Training of Deep LSTM Networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11554 LNCS*, 3–10. https://doi.org/https://doi.org/10.1007/978-3-030-22796-8_1

Yuan, C., Wu, J., Li, H., & Wang, L. (2018). Personality Recognition Based on User Generated Content. *2018 15th International Conference on Service Systems and Service Management, ICSSSM 2018*. https://doi.org/10.1109/ICSSSM.2018.8465006

Yuk Ying Chung, V., Afizi bin Mohd Shukran, M., Chen, F., Ying Chung, Y., Hcchoi, E., Liu, L., Afizi Mohd Shukran, M., & Shi, D. Y. (2007). *A new hybrid audio classification algorithm based on SVM weight factor and Euclidean distance*. https://www.researchgate.net/publication/234797090

Zhang, J., & Zong, C. (2015). Deep Neural Networks in Machine Translation: An Overview. *IEEE Intelligent Systems*, *30*(5), 16–25. https://doi.org/10.1109/MIS.2015.69

Zhang, X., Lin, P., Chen, S., Cen, H., Wang, J., Huang, Q., Xu, Y., Tang, J., & Huang, P. (2017). Valence-Arousal prediction of Chinese Words with multi-layer corpora. *Proceedings of the 2016 International Conference on Asian Language Processing, IALP 2016*, 304–307. https://doi.org/10.1109/IALP.2016.7875992

Zhou, P., & El-Gohary, N. (2016). Ontology-Based Multilabel Text Classification of Construction Regulatory Documents. *Journal of Computing in Civil Engineering*, *30*(4). https://doi.org/10.1061/(ASCE)CP.1943-5487.0000530