DESIGN AND DEVELOPMENT OF SCALABLE ANALYTICS FRAMEWORKS WITH APPLICATIONS IN BLOCKCHAIN SMART CONTRACT SECURITY AND POLITICAL NEWS MINING

by

Maryam Bahojb Imani

THE UNIVERSITY OF TEXAS AT DALLAS
EST. 1969

APPROVED BY SUPERVISORY COMMITTEE:

Dr. Latifur Khan, Chair

Dr. Bhavani Thuraisingham, Co-Chair

Dr. Farokh B. Bastani

Dr. Weili Wu

Copyright © 2020

Maryam Bahojb Imani

All rights reserved

To my beloved family,

who gave me wings to fly, roots to come back, and reasons to stay.

DESIGN AND DEVELOPMENT OF SCALABLE ANALYTICS FRAMEWORKS WITH APPLICATIONS IN BLOCKCHAIN SMART CONTRACT SECURITY AND POLITICAL NEWS MINING

by

MARYAM BAHOJB IMANI, BS, MS

DISSERTATION
Presented to the Faculty of
The University of Texas at Dallas
in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY IN
COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS
May 2020

ACKNOWLEDGMENTS

This journey would not have been possible without the support of my family, professors and friends. To my family, thank you for encouraging me in all of my pursuits and inspiring me to follow my dreams. I am especially grateful to my parents, who supported me in all aspects. I always knew that you believed in me and wanted the best for me.

I would like to thank my advisor, Dr. Latifur Khan for his valuable teaching, wisdom, continuous support and guidance in conducting research throughout my PhD years. His endless trust, patience, and advice have helped me towards my accomplishments.

I would like to thank my co-advisor, Dr. Bhavani Thuraisingham, whose guidance, support and encouragement have been invaluable throughout this study. Dr. Bhavani is not only a great professor, she is also a compassionate friend.

Besides my advisors, I would like to thank my committee members, Dr. Weili Wu and Dr. Farokh Bastani for evaluating my dissertation and my Examining Committee Chair, Dr. Tyler Summers.

I would also like to thank all the administrative assistants of the Department of Computer Science, especially Rhonda Walls, who was nice and helpful.

I would like to take this opportunity to express my thanks to Afsaneh Rafie and her family who have been with me all the time in this journey. Their love and support mean a lot to me.

I am really grateful to my lab mates Bo, Ashrafi, Sayeed, Khaled, Ahmad, Vishal, Swarup, Gbadebo, Solaimani, Aswin, Yang, Yifan, and Wang for being great friends. I really enjoyed their company, collaboration, and depth of knowledge.

To my friends, thank you for listening, offering me advice, and supporting me through this entire process. Special thanks to Sepideh, Marjan, Pouyeh, Mahdieh, Sara, Faezeh, Banafsheh,

Sona, Ceren, Aref, Hafez, Hamed, Reza, Mohsen, Mohammad Ali, Brian, and John and Mary Anderson.

Finally, I would like to thank NSF for sponsoring my research.

February 2020

DESIGN AND DEVELOPMENT OF SCALABLE ANALYTICS FRAMEWORKS WITH APPLICATIONS IN BLOCKCHAIN SMART CONTRACT SECURITY AND POLITICAL NEWS MINING

Maryam Bahojb Imani, PhD
The University of Texas at Dallas, 2020

Supervising Professors: Dr. Latifur Khan, Chair
Dr. Bhavani Thuraisingham, Co-Chair

Nowadays, high amounts of data are continuously generated at unprecedented rate from various domains such as e-commerce, education, health, security, and social networks. This is due to many technological advancements, including Internet of Things (IoT), autonomous driving, the proliferation of Cloud Computing, data center consolidation as well as the growth of smart devices. The term big data was created to demonstrate the meaning of this emerging trend. The high volumes, velocities, and varieties of data pose a great challenge for the data mining community to extract useful knowledge. In response to this, we need scalable analytics frameworks for data acquisition, filtering, and analyzing in a quick time.

Current state-of-the-arts like advanced analytics, Machine Learning (ML), Natural Language Processing (NLP) can be utilized to handle heterogeneous Big Data. Yet, most of these systems suffer scalability issues. In this dissertation, we focus on social science and blockchain areas. More specifically, we focus on location extraction from unstructured political text data, vulnerability detection in Blockchain's smart contracts and fault diagnosis in wind turbine vibration data. With regard to focus location extraction, although various tools exist to identify geolocation, they fail to identify at a granular level; they mostly rely on external knowledge, and they do not support most languages. We propose a novel scalable framework PROFILE to extract the primary focus location from political news articles in different languages. With regard to blockchain, existing solutions to this problem particularly rely on human experts to define features or different rules to detect vulnerabilities, which often lead to missing many vulnerabilities and they are inefficient in detecting new vulnerabilities. We develop a novel scalable framework to detect vulnerabilities in smart contracts. With regard to fault diagnosis in wind turbines, real-time fault diagnosis for streaming vibration data from turbine gearboxes is still an outstanding challenge. Moreover, monitoring gearboxes in a wind farm with thousands of wind turbines requires massive computational power. We address these challenges by developing SAIL, a scalable real-time framework, to capture wind turbine vibration data using a novel feature extraction and predict faults in gearbox. We show empirically that the proposed techniques outperform state-of-the-art techniques in all three areas.

TABLE OF CONTENTS

ACKNOWLEDGMENTS . . . v

ABSTRACT . . . vii

LIST OF FIGURES . . . xii

LIST OF TABLES . . . xiv

CHAPTER 1 INTRODUCTION . . . 1

- 1.1 Big Data Frameworks . . . 1
- 1.2 Contributions . . . 3
  - 1.2.1 Designing a scalable framework for primary focus location in political news articles . . . 3
  - 1.2.2 Designing a scalable framework for vulnerability detection in smart contracts . . . 4
  - 1.2.3 Designing a scalable framework for wind turbine fault diagnosis . . . 6
- 1.3 Organization of Dissertation . . . 8

CHAPTER 2 RELATED WORKS . . . 9

- 2.1 Geolocation Extraction From Unstructured Text . . . 9
  - 2.1.1 Geolocation as Named Entity Recognition . . . 10
  - 2.1.2 Location Named Entity Resolution . . . 14
  - 2.1.3 Geolocation Extraction from Event Data . . . 20
- 2.2 Smart Contract Vulnerability Detection . . . 24
  - 2.2.1 Static Analysis Tools . . . 24
  - 2.2.2 Dynamic Analysis Tools . . . 25
- 2.3 Wind Turbine Fault Diagnosis . . . 25
  - 2.3.1 Model-based and model-free fault diagnosis methods . . . 26

CHAPTER 3 A SCALABLE FRAMEWORK FOR PRIMARY FOCUS LOCATION EXTRACTION . . . 28

- 3.1 Introduction . . . 28
- 3.2 Background . . . 29
  - 3.2.1 Geolocation Extraction . . . 29
  - 3.2.2 Multilingual Word Vectorization . . . 31

3.3 Approach 34
3.3.1 Focus Location Extraction 35
3.4 Experiments 39
3.4.1 Dataset 40
3.4.2 Focus Location Extraction Results 44
3.4.3 Intra-Language Bias Correction Results 46
3.4.4 Inter-Language Bias Correction Results 49
CHAPTER 4 A SCALABLE FRAMEWORK FOR VULNERABILITY DETECTION IN SMART CONTRACTS 52
4.1 Introduction 52
4.2 Background 53
4.2.1 Ethereum Virtual Machine 54
4.2.2 Common Ethereum Smart Contract Vulnerabilities 55
4.3 Proposed Framework 58
4.3.1 Smart Contract Collection 58
4.3.2 Disassembling 61
4.3.3 Feature Engineering 62
4.3.4 Bidirectional Long Short-Term Memory 64
4.4 Evaluation 68
4.4.1 Benchmarks 68
4.4.2 Neural network configuration 68
4.4.3 Experimental Results 70
CHAPTER 5 A SCALABLE FRAMEWORK FOR WIND TURBINE FAULT DIAGNOSIS 73
5.1 Introduction 73
5.2 Background 75
5.2.1 Industrial Applications 75
5.2.2 Big Data Analytics 77
5.3 SAIL Architecture 78
5.3.1 Scalable Monitoring System 78

5.3.2 Bottom-Up Architecture 78

5.4 Fault Diagnosis Framework 78

5.4.1 Sensor Layer 80

5.4.2 Fog Layer 82

5.4.3 Cloud Layer 83

5.5 Experimental Results 84

5.5.1 Algorithm Accuracy Test 85

5.5.2 Platform Scalability Test 94

CHAPTER 6 CONCLUSION AND FUTURE WORK 98

6.1 Primary Focus Location Extraction 98

6.2 Vulnerability Detection in Smart Contracts 99

6.3 SAIL: A Scalable Wind Turbine Fault Diagnosis Framework 100

APPENDIX MATHEMATICAL FRAMEWORK FOR FEATURE EXTRACTION FROM WIND TURBINE VIBRATION 102

A.1 ARMA Representation of Vibration Data 102

A.2 Harmonic Decomposition 103

A.3 Model Order Selection 105

A.4 Dictionary Learning and Compression 106

A.5 Sparse Decomposition 107

A.6 Dictionary Learning Using K-SVD 108

REFERENCES 110

BIOGRAPHICAL SKETCH 119

CURRICULUM VITAE

LIST OF FIGURES

2.1 A sample English news report with different place names from Atrocity dataset (Schrodt and Ulfelder, 2009) 21

2.2 A sample Spanish news report with different place names from Revista Noche y Niebla 22

2.3 A sample Arabic news report with different place names from Alghad news agency 22

3.1 A high level schema of Profile (Primary Focus Location Extraction). 36

3.2 Percentage of documents containing focus location in the initial set of sentences. 42

3.3 Performance of bias-corrected classifier —●— $\text{Profile}_s^{\text{KMM}}$ with Atrocity dataset as training and NYT dataset as test, compared to a biased classifier —●— $\text{Profile}_s^{\text{SVM}}$. 45

3.4 Performance of bias-corrected classifier —●— $\text{Profile}_s^{\text{KMM}}$ with Spanish Protest dataset as training and Nocheyniebla dataset as test, compared to the biased classifier —●— $\text{Profile}_s^{\text{SVM}}$. 47

3.5 Performance of bias-corrected classifier —●— $\text{Profile}_s^{\text{KMM}}$ with Arabic world news dataset as training and Arabic Arabian news dataset as test, compared to the biased classifier —●— $\text{Profile}_s^{\text{SVM}}$. 48

3.6 Performance of bias-corrected classifier —●— $\text{Profile}_s^{\text{KMM}}$ with English Atrocity dataset as training and Spanish Nocheyniebla news dataset as test, compared to the biased classifier —●— $\text{Profile}_s^{\text{SVM}}$. 50

3.7 Performance of bias-corrected classifier —●— $\text{Profile}_s^{\text{KMM}}$ with English Atrocity dataset as training and Arabic news dataset as test, compared to the biased classifier —●— $\text{Profile}_s^{\text{SVM}}$. 51

4.1 Smart contract CALL flow 59

4.2 Smart contract vulnerability detection proposed framework 60

4.3 Compiling code in Algorithm 2 to EVM bytecode by using Solidity 0.4.1 compiler 61

4.4 Compiling code in Algorithm 3 to EVM bytecode by using Solidity 0.4.1 compiler 61

4.5 Control flow graph for the Fibonacci contract (Algorithm 2) by using (Brent et al., 2018) 65

4.6 Control flow graph for the Factorial contract (Algorithm 3) by using (Brent et al., 2018) 66

4.7 Percentage of benign and vulnerabilities in the benchmark 69

4.8 Number of smart contracts in Benchmark with one or more vulnerability 70

4.9 LSTM and BiLSTM models 71

4.10 Implementation of LSTM model for smart contract vulnerability detection in Keras 72
5.1 The proposed three-layer platform (SAIL). 79
5.2 The proposed fault diagnosing framework 81
5.3 The test turbine (image from ((NREL), 2017)) 85
5.4 The gearbox block diagram 85
5.5 Experimental setup of CWRU bearing data (Loparo, Loparo). 87
5.6 Performance of $SAIL_{Imb+RF}$ ▯▯ with different ratios of normal to fault data instances, compared to $SAIL_{U+RF}$ ▯▯ and $SAIL_{O+RF}$ ▯▯ on NREL dataset . . . 92
5.7 Performance of $SAIL_{Imb+RF}$ ▯▯ with different ratios of normal to fault data instances, compared to $SAIL_{U+RF}$ ▯▯ and $SAIL_{O+RF}$ ▯▯ on CWRU dataset . . . 93
5.8 Comparing processing latency during training Random Forest model with Mllib in Spark ($SAIL_{RF}^{Spark}$) —×— and Sklearn library ($SAIL_{RF}$) —●— using NREL and CWRU dataset. 95
5.9 Average diagnosis latency for real-time data from different number of input sources 97

LIST OF TABLES

2.1 Main characteristics of the Named Entity Recognition tools 13

2.2 Output of focus location extraction from existing tools including Cliff-Clavin, Geoparser, Mordecai, Stanford and MITIE for the given example in Figure 2.1 . 23

2.3 Output of location extraction from Stanford and MITIE for the given example in Figure 2.2 24

2.4 Translated outputs of location extraction from Polyglot for the given example in Figure 2.3 24

3.1 Capabilities of different tools in focus country and focus location extraction. Here, ✓ indicates presence and ✗ indicates absence of corresponding capability. . . . 31

3.2 List of Symbols 34

3.3 Dataset Statistics 40

3.4 Primary focus location accuracy comparison between different methods 44

4.1 Disassembling Figure 4.4 bytecode to opcode 62

4.2 Disassembling Figure 4.3 bytecode to opcode 63

4.3 Comparison of the proposed method with KNN, Random Forest, and SVM . . . 71

5.1 Datasets Statistics 88

5.2 Comparison with prior works Using NREL benchmark 89

5.3 Comparing various Fault Diagnosis methods using CWRU benchmark 90

CHAPTER 1

INTRODUCTION

The world today is generating an inconceivable amount of data every minute in different contexts such as online social media, smartphones, sensors, cameras, and news. Traditional technologies have limited storage capacity to store all data for long periods and manage huge datasets efficiently because it is costly. Due to these reasons, some of these data may lose value or be lost forever. Therefore, it is essential to develop scalable systems that can ingest and analyze unstructured data on a continuous basis.

In the recent era, businesses are more aware that data analysis is increasingly becoming a vital factor to be competitive, to discover new insight, and to personalize services. In order to cope with these massive, exponentially increasing amounts of heterogeneous data that are generated faster and faster, an organization should act swiftly to process data and take necessary actions. In response to these requirements on massive data, agile, analytic, adaptive, and scalable frameworks are required which combine business rules and predictive analytics to make business decisions. Therefore, it is a challenging task given the volume, velocity, and possibly complex nature of the input data. The lack of scalability, flexibility and performance in big data frameworks may lead to catastrophic consequences with a severe impact on business continuity.

1.1 Big Data Frameworks

Because of the large data, new systems and technologies, instead of the single machines, have emerged to scale out computations to multiple nodes. Big Data has three main characteristics, which are also some the challenges of the data in industry: volume, velocity and variety (Zikopoulos et al., 2011).

Volume: Every second, a huge amount of data is generating in different places or by various devices. In 2015, Yin reported that annually data generated by modern industry has reached a total amount of more than 1000 Exabytes (10^{36} byte) and is predicted to grow 20-fold in the next decade (Yin and Kaynak, 2015). As a result, the exponentially increasing volume of data is a serious challenge ahead of today's industrial world.

Velocity: Different devices such as sensors have led to an unprecedented pace of data generation and are driving a growing need for processing this big volume of data in real-time. So, growth in data velocity is another big data challenge which stands in front of industries, since it affects time and efficiency.

Variety: Data is generating in different formats and structures, such as digital and analogous signals. So, the other challenge is finding the correlation among various types of data in real-time and analyzing them.

Many different Big Data analysis tools with different features have been developed so far, such as Hadoop (Shvachko et al., 2010), Apache Storm (Storm, Storm), and Apache Spark (Spark, Spark). In this project, Spark is selected over a traditional distributed framework (e.g., Hadoop and MapReduce, etc.) that is not ideal for stream data processing as described in the next subsection.

The Apache Spark is a fast and general purpose cluster computing engine for large-scale data analytics. Apache Spark was developed at the University of California, Berkeley AMP Lab (Spark, Spark). Many different applications have been developed based on Spark clustering so far, such as anomaly detection (Solaimani et al., 2014), political event coding (Solaimani et al., 2016), processing the traffic data (Sinnott et al., 2015), etc.

Another feature that Spark offers is supporting scalable, high-throughput, fault-tolerant real-time data stream processing (Zaharia et al., 2012). Spark streaming uses a "discretized streams" which is an incremental stream processing model. Data can be fed by many sources like Kafka (Kafka, 2014).

Machine Learning library (MLLIB) is Spark's distributed and scalable machine learning library (Meng et al., 2016). It improves the computational efficiency by using data-parallelism or model-parallelism techniques to store and process data or models. MLLIB includes more than 50 common algorithms for classification, clustering, regression, collaborative filtering and dimensionality reduction.

1.2 Contributions

We have highlighted our contributions in this section.

1.2.1 Designing a scalable framework for primary focus location in political news articles

Political news reports are populated all over the world in various languages. It has a great value to automatically detect the geolocation from these reports for better understanding of the associated events. Although various open-source and commercial tools exist to identify geolocation, they fail to identify at a granular level such as locality or city and they do not support most languages. Most of the techniques view the problem in terms of Named Entity Recognition (NER) and identify geolocation information at the country level for a given text. In this study, we consider English, Spanish and Arabic news articles from different publishers. We define *primary focus location* as the actual location where the event occurred amongst other *focus* locations mentioned in the report. Our aim is to extract the *primary focus location* regardless of the language from articles belonging to different news agencies. We propose a mechanism to identify potential sentences containing *focus* locations using NER. After that, we perform sentence embedding over words from different languages, and then employ a supervised classification mechanism to predict the *primary focus location.* We also perform bias correction over the training data using a suitable adaptation mechanism to reduce the sampling bias in training data. Our method trains a classifier using bias-corrected training data from news articles published by an agency in one language, while testing the model on news articles published by another agency in a different language.

The key contributions of this study are as follows:

- We address the problem of automatically predicting a primary focus location for event-based news reports in different languages by extracting semantic features that aid in capturing patterns of focus location occurrences in various languages, rather than using an external database such as gazetteer[1].
- We propose *Profile* which identifies the primary focus location of political news article in different languages. In particular, we apply supervised learning on a smaller set of biased training data and leverage a well-known bias-correction mechanism to evaluate test data from the same domain to address the label scarcity problem. This approach is language agnostic.
- We empirically evaluate Profile over real-world English, Spanish and Arabic news articles and compare its performance against other available tools.

1.2.2 Designing a scalable framework for vulnerability detection in smart contracts

With the increase in adoption of blockchain technology in providing decentralized solutions to various problems, smart contracts have been becoming more popular to the point that billions of US Dollars are currently exchanged everyday through such technology. Meanwhile, various vulnerabilities in smart contracts have been exploited by attackers to steal cryptocurrencies worth millions of dollars. The automatic detection of smart contract vulnerabilities is an essential research problem. Yet, existing solutions to this problem particularly rely on human

---

[1] http://www.geonames.org/

experts to define features or different rules to detect vulnerabilities; which often lead to missing many vulnerabilities and they are inefficient in detecting new vulnerabilities. In this study, we address these challenges and propose a framework to analyze the data and detect some vulnerabilities in Ethereum smart contracts on the blockchain platform. We apply machine learning-based (i.e. deep learning-based) vulnerability detection to relieve human experts from the tedious and subjective task of manually defining features and rules. In particular, we need to find representations of Ethereum smart contracts that are suitable for deep learning. One of the main challenges in this step is that the smart contracts are translated to low-level bytecode for deployment on the blockchain. It means the high-level source codes of smart contracts are rarely available. To address this challenge, we convert those bytecode to sequence of more readable low-level mnemonics. To represent the smart contracts, we construct the control flow graphs to show the semantic relations between lines of codes which are not necessarily consecutive. Then, we traverse these graphs to transform them into vectors. This leads to extract features to feed into our developed deep learning-based vulnerability detection system for smart contracts.

The key contributions of this study are as follows:

- Since the high-level source code of smart contracts is hardly available, we directly use bytecode as an input in our system. In our feature engineering pipeline, we convert the bytecode to a sequence of more readable low-level opcodes and construct its control flow graph. Then, we traverse these graphs to transform them into vectors. These vectors can keep the semantic relations between lines of codes (operations) which are not necessarily consecutive.
- We use deep learning for vulnerability detection in Ethereum smart contracts. This approach has a great potential because deep learning does not need human experts to manually define features and rules, meaning that vulnerability detection can be automated. Our model utilizes Bidirectional Long Short-Term Memory Networks, which

can automatically capture the most important semantic information in a sequence of program control flow, without using external knowledge.

- We empirically evaluate the proposed framework over real-world large-scaled Ethereum smart contract benchmark and compare its performance against other available machine learning-based approaches.

1.2.3 Designing a scalable framework for wind turbine fault diagnosis

Failure of a wind turbine is largely attributed to faults that occur in its gearbox. Maintenance of this machinery is very expensive, mainly due to large downtime and repair cost. While much attention has been given to detect faults in these mechanical devices, real-time fault diagnosis for streaming vibration data from turbine gearboxes is still an outstanding challenge. Moreover, monitoring gearboxes in a wind farm with thousands of wind turbines require massive computational power. In this study, we propose a three-layer monitoring system: Sensor, Fog, and Cloud layers. Each layer provides a special functionality and runs part of the proposed data processing pipeline.

In the *Sensor* layer, vibration data is collected using accelerometers. Industrial single chip computers are the best candidates for node computation. Since the majority of wind turbines are installed in harsh environments, sensor node computers should be embedded within wind turbines. Therefore, a robust computation platform is necessary for sensor nodes. In this layer, we propose a novel feature extraction method which is applied over a short window of vibration data. Using a time-series model assumption, our method estimates vibration power at high resolution and low cost. *Fog* layer provides Internet connectivity. Fog-server collects data from sensor nodes and sends them to the cloud. Since many wind farms are located in remote locations, providing network connectivity is challenging and expensive. Sometimes a wind farm is offshore and a satellite connection is the only solution. In this regard, we use a compressive sensing algorithm by deploying them on fog-servers to conserve communication bandwidth. *Cloud* layer performs most of the computations. In the online mode, after decompression, fault diagnosis is performed using a trained classifier, while generating reports and logs. Whereas, in the offline mode, model training for classifier, parameters learning for feature extraction in sensor layer and dictionary learning for compression on fog servers and decompression are performed. The proposed architecture monitors the health of turbines in a scalable framework by leveraging the distributed computation techniques.

Our contribution is twofold. First, we propose a scalable cloud-based computation platform denoted by *SAIL* (Scalable wind turbine fAult dIagnosis pLatform). Second, we propose a novel signal processing pipeline to diagnose the health state of a wind turbine. This algorithm runs on different computational elements in SAIL and is scalable.

The key contributions of this study are as follows:

- First, we propose a novel data-driven fault diagnosis algorithm. The proposed algorithm uses super-resolution spectral analysis for feature extraction from the vibration signal. We employ a time series model for vibration signal and use the model parameters for feature extraction. These parameters represent the spectrum of vibration signals in a very compact way and require a reasonable computational cost. For classification, we apply the Random Forest algorithm which is in general suitable for parallel implementation.
- Second, we have developed a novel real-time framework for multi-source stream data to detect the turbine's faults using Apache Spark, with Kafka as a message broker (Spark, Spark). In this framework, different sources can send their data stream through Kafka to a Spark cluster. Then, we apply the trained Random Forest model to analyze the data.
- Third, we address the class imbalance problem prevalent in applications such as fault diagnosis, unlike previous studies. Fault data is typically rare while fault-free data is available in abundance. Applying a traditional machine learning classifier over such biased data yields a largely inaccurate model. We address this challenge by leveraging sampling techniques during data preprocessing.

- Fourth, we compare our fault detection technique by implementing it on traditional and Spark-based platforms. Our empirical results on real-world datasets show high detection accuracy with low latency. In addition, our proposed platform performance analysis indicates a significant reduction in computational time.

1.3 Organization of Dissertation

The rest of my dissertation is organized as follows: Chapter 2 sums up all related works, Chapter 3 describes the primary focus location extraction framework. Chapter 4 presents the smart contract vulnerability detection framework. Chapter 5 discusses a scalable framework for wind turbine fault diagnosis. Finally, Chapter 6 presents the conclusion and future works.

CHAPTER 2

RELATED WORKS

2.1 Geolocation Extraction From Unstructured Text

Geolocation information plays an important role in different applications, such as considering the daily news; methods for searching and organizing news stories can highly take advantage of place information, as it is one of the basic important parts of the Five Ws, i.e. Who, What, When, Where, Why, applied in journalism to characterize events. As another example, consider advertising: by extracting geolocation information from available data, different local service or product providers can offer more personalized and local relevant advertisement content to the user.

We present here a survey of several years of research in the geolocation extraction field, from 2000 to 2018. This research includes different tasks and analysis that have been done in the field of geolocation extraction from the text. The main three tasks in this field are: 1- Location named entity extraction, 2- Location named entity resolution, 3- the event's location extraction.

Nowadays, the term Named Entity is increasingly used in Natural Language Processing. Different tools have been developed to extract place named entity from text. According to their requirements, they utilized various learning techniques, ranging from supervised learning to unsupervised learning. In this survey, we will review and explain the different techniques to extract named entities, particularly place names.

The other task after geolocation names extraction is disambiguating the geolocation names. Geoparsers are Natural Language Processing (NLP) systems designed to analyze unstructured text in order to extract occurrences of location entities and place names and to resolve their ambiguities. Homographs and toponyms pose well-known problems in the process of geo-tagging news articles. In this article, we will present different approaches to address these issues.

In every moment, several events are happening all over the world, and they are being reported by local or international news agencies. Automatic geolocation information extraction is vital to detect and analyze the impact of these events. We study extracting focus geolocations automatically from unstructured text-based news reports. The focus geolocation for a given news report is a location where an event mentioned in the report has happened. In the next sections, we will study several tools that have been developed to extract geolocation information from text.

Different applications can utilize rich, timely geolocation information to provide better support for decision making. A few examples are determining crime pattern locations, predicting the place of protests and political unrest, or identifying the geolocation of natural disasters from tracking news events.

2.1.1 Geolocation as Named Entity Recognition

The most studied types are three specializations of “proper names”: names of “persons”, “locations” and “organizations”. These types are collectively known as “enamex” since the MUC-6 competition. The type “location” can thusly be partitioned into numerous subtypes of “fine grained locations”: Locality, city, state, country, etc. ((Fleischman, 2001), (Lee and Lee, 2005)).

Presently, there are various open source tools available for named entity recognition (NER). Popular software programs for the NER task include the Stanford Named Entity Recognizer (part of Stanford CoreNLP), Apache Open NLP algorithm, and MIT Information Extraction (MITIE), developed by MIT’s Lincoln Laboratory. Besides these approaches, there are some other methods to solve this problem. In the next subsection, we will discuss them.

Generally, the problems of machine learning are typically classified into three different categories, depending on the nature of the learning and data available to a learning system. These are supervised learning, unsupervised learning, and semi-supervised learning.

The idea of supervised learning (classification) is to study the features of positive and negative instances of named entities over a large collection of annotated documents and design a model to classify and predict the label of unseen named entity instances.

The most popular technique for addressing the named entity recognition problem is supervised learning. Supervised learning techniques include Hidden Markov Models (HMM), Maximum Entropy Models, Support Vector Machines (SVM), and Conditional Random Fields (CRF).

Stanford Named Entity Recognition Stanford NER [1] is a Java implementation of a Named Entity Recognizer to find several different classes of named entities: Person, Organization, Location, Date, Time, Percent, Money. It uses a linear chain Conditional Random Field (CRF) model to label sequences of words in a document into the mentioned entity types.

MITIE The MIT Information Extraction Toolkit [2] (King, King) is a free, open-source software library of state-of-the-art NLP tool developed at MIT Lincoln Laboratory. MITIE can automatically extract the named entities and different binary relations from unstructured text in English and Spanish. MITIE uses distributional word embeddings for dimensionality reduction, structured support vector machines for learning syntactic relationships, and automated hyperparameter optimization to enable user customization. MITIE is built on the Dlib machine learning library (King, 2009), contains interfaces to C, C++, Java, R, and Python, and is easy for a user to integrate it into his/her own applications.

---

[1] http://nlp.stanford.edu/software/CRF-NER.shtml

[2] https://github.com/mit-nlp/MITIE

OpenNLP OpenNLP [3] is a Java-based library for different natural language processing problems, such as tokenization, part-of-speech tagging, and named entity recognition. For named entity recognition, it trains a Maximum Entropy model using the information from the whole text to identify entities in the text.

Yahoo! PlaceMaker Yahoo! PlaceMaker [4] is a geoparsing service that recognizes place names in a given unstructured document.

TwitterNLP TwitterNLP is a specific toolkit developed for processing the natural language on Twitter textual data. It utilizes a supervised topic model, called LabeledLDA, and also Freebase as a source of distant supervision, to classify entity mentions in tweets. TwitterNLP performs three tasks of classification, part-of-speech tagging, and chunking.

The principal characteristics of these systems are outlined in Table 2.1. Stanford NER, MITIE and OpenNLP, three state-of-the-art NER tools, are originally trained on news data which include formal text. They provide retraining and customizing options for other types of text. TwitterNLP, based on topic modeling, can only be customized by changing the dictionary in Freebase.

As we are interested on the location and the geo-location extraction task, LOCATION and ORGANIZATION, tagged by Stanford NER, MITIE and OpenNLP, and GEOLOCATION, COMPANY and FACILITY, by TwitterNLP, are considered as locations. For Yahoo! PlaceMaker we only use the locations found in the text submitted to this service.

---

[3] http://opennlp.apache.org

[4] http://developer.yahoo.com/geo/placemaker/ visited in January 2013. Now it is migrated to Yahoo! Boss at http://developer.yahoo.com/boss/geo/.

Table 2.1: Main characteristics of the Named Entity Recognition tools

Tool	**Entity Type**	**Model**	**Training Data**	**Retrainable**
Stanford NER	PERSON, ORGANIZATION, LOCATION, TIME, DATE, MONEY, PERCENT	CRF	News Data	✓
MITIE	PERSON, ORGANIZATION, LOCATION	Structural SVM	News Data and Wikipedia	✓
OpenNLP	PERSON, ORGANIZATION, LOCATION	Maximum Entropy	News Data	✓
TwitterNLP	PERSON, GEO-LOCATION, COMPANY, PRODUCT, FACILITY, TV-SHOW, SPORTSTEAM, BAND MOVIE, and OTHER.	LabeledLDA	News Data	✗
Yahoo! PlaceMaker	N/A	N/A	N/A	N/A

2.1.2 Location Named Entity Resolution

Geolocation extraction tools use Natural Language Processing techniques to analyze unstructured text in order to extract occurrences of location entities and place names and to resolve their ambiguities. In this section, we are going to present different strategies to handle the location named entity resolution.

Place Name Disambiguation

Homographs are a common problem in the process of geolocation extraction from news reports (Pouliquen et al., 2006). Especially, words referring to place names probably: (a) occur as person names (e.g., Conrad Hilton are the name of towns in the New Zealand, USA and Canada); (b) occur as common words (e.g., And is a village in Iran); or (c) have variants, i.e., well known cities have language variants (e.g., Seul meaning alone/only in French is also the Capital of South Korea in Portuguese and Italian); (d) refer to different locations (e.g., there are 33 places named Clinton). Further complexities are produced by language inflection and capital cities referring to the reporting location.

To resolve such problems, various heuristics including human linguistic knowledge and external resources like WordNet, encyclopedia and Web documents have been applied. One of the common disambiguation approaches is using a gazetteer which is a collection of geographic locations that provides for each location a latitude and longitude, the location's various classifications (country, state, city, park, building, etc.), its population (if applicable), and information on the location's inclusion hierarchy (city, state, country and continent). For a given location entity extracted from text, a gazetteer can be applied to find a list of toponyms that match the entity's name and present candidate locations for that entity. Different strategies have been developed that use such candidate locations to disambiguate an extracted location entity on the map by assigning a specific toponym to it (Abrol and Khan, 2010).

(Erik, 2008) proposed homograph disambiguation at two particular levels, firstly at the level of separate document and secondly at the level of clustered documents. The geo-tagging algorithm works as follows: First, the problem is solved via removing from the text names of known people and organizations. Next, the gazetteer of place, province, region and country names is used to find the matches of candidate locations in the news reports. In order to disambiguate homographs that are common words and location names, the traditional strategy is to use language dependent stop word lists. They have applied a different method based on two characteristics in the gazetteer. The first characteristic classifies places based on their size, in a way that capital cities and major cities have a higher class than small villages. The second characteristic maintains the hierarchical relation of location in its administrative located hierarchy (i.e., town, in province, in region, in country). The disambiguation algorithm lets high class locations pass through as well as locations that have a containment relation with other candidate locations (e.g., Paris, Texas, USA). To resolve different variants, they discard any geo-tags that are incompatible with the language of the article (Wang et al., 2004).

In order to handle name inflection, names are maintained with their variants encoded as a regular expression. Only matches, for which the language of the name and the article are compatible, are maintained in the candidate location list. Next, a Newswire location filtering is applied, where the word position of the candidate location is used to promote locations occurring after an initial location since the Newswire location generally appears earlier in the article (Erik, 2008).

Metacarta [5] is a commercial company that offers geographic information retrieval technology. The company also provides a freely-available Web service that can be used to identify and disambiguate place references over text (Rauch et al., 2003).

---

[5] http://metacarta.com/

Yahoo! Placemaker [6] is a geotagging web service that provides third-party developers the means to enrich their applications or Web sites with geographic information. The service can recognize, extract, and disambiguate location names from unstructured and semi-structured documents. It is also able to use the location references in a document, together with a pre-determined set of rules, to discover the geographic scope that best encompasses its contents. Thus, given a textual document, Yahoo! Placemaker returns unique Where-on-Earth identifiers (WOEIDs) for each of the named places and scopes. Through these identifiers, one can use the Yahoo! GeoPlanet [7] Web service to access hierarchical information (i.e., containing regions) or spatial information (i.e. centroids and bounding boxes). There are two flavors of document scopes in Placemaker, namely the geographic scope and the administrative scope. The geographic scope is the place that best describes the document. The administrative scope is also the place that best describes the document, but is of an administrative type (i.e., Continent, Country, State, County, Local Administrative Area, Town, or Suburb). Placemaker is a commercial product and not many details are available regarding its functioning. However, some information about the service is available in the Web site, together with its documentation. For example, the Website claims that "when the service encounters a structured address, it will not perform street level geocoding but will instead disambiguate the reference to the smallest bounding named place known, frequently a postal code or neighborhood". The Website also claims that besides place names, the service also understands geography-rich tags, such as the W3C Basic Geo Vocabulary and HTML microformats [8]. However, no details about the rules that are used in the scope assignment process are given in the documentation for the service. The Placemaker Web service accepts plain text as input, returning an XML document with the results. The service has an input

---

[6] http://developer.yahoo.com/geo/placemaker/

[7] https://developer.yahoo.com/geo/geoplanet/

[8] http://microformats.org/wiki/geo or http://microformats.org/wiki/adr

parameter that allows users to provide the title of the document separately from the rest of the textual contents, weighting the title text as more representative. In their experiments, they used the Web service as a black-box to assign scopes to the Web documents, using the option that weights the title text as more important than the rest.

In the context of the Web-a-Where project, (Amitay et al., 2004) proposed a method for assigning Web documents to the corresponding geographic scopes. The system gave confidence scores to every location using some predefined rules and propagating scores between locations that are in a containment relationship. It produces a candidate list of geographic names mixing locations regardless of their administrative levels, i.e., there are locations at different levels in the rank-list of geographic names for the web page. Their technique leverages on part-of relations among the recognized place references, provided by a hierarchical gazetteer. The basic idea is that, for example, if several cities from the same country are mentioned, this probably means that this country is the scope, i.e. the algorithm tries to generalize from the disambiguated place references. More specific places are scored higher if they are the only places mentioned.

In another study (Yu, 2014), they also use Web-a-where technique. However, they consider each administrative level (country, state and city level) separately at first and then determine the correct one. This method is based on a probabilistic model capturing the relation between the mentions of named entities and locations, rather than some hand-picked confidence scores that are used by the web-a-where approach.

In the context of the GREASE project, Martins and Silva proposed a scope assignment method based on a graph-ranking approach (Martins and Silva, 2005). The idea was to represent the gazetteer used for place reference disambiguation as a graph, where the nodes correspond to different places and the edges correspond to semantic relationships (part-of, containment or adjacency) between places. Nodes on this graph can be weighted according to the occurrence frequency of place references in a document, and edges can be weighted according to the relative importance of the different types of relationships. A graph-ranking algorithm, PageRank, is then applied to this graph, and finally the highest ranked node is selected as the scope. In case of ties, the node connected to the highest number of edges is selected. By propagating scores across the graph, this algorithm tries, at the same time, to generalize and to specify from the available information, in order to find the region that best reflects the scope of the document.

In another study, they have applied a combination of techniques (i.e. (Amitay et al., 2004) and (Martins and Silva, 2005)) to solve the ambiguity problems. Their approach permits a general region to be chosen if several different places in it are mentioned, with no specific emphasis on any. The algorithm starts by placing the recognized place references in a locational hierarchy. By looping over the disambiguated references, the algorithm aggregates the importance of the various levels in the hierarchy. The levels are then sorted by importance and the highest ranked level is returned as the scope. The multiplying discount parameters correspond to those originally reported in the Web-a-Where paper. Finally, the highest scoring taxonomy node would be selected as the scope to assign. On the other hand, they apply graph-ranking approach (Martins and Silva, 2005). But for computing the PageRank score, they use the open-source weighted PageRank implementation made available by the Laboratory for Web Algorithmics of the University of Milan. Since this implementation does not allow for weighted nodes, they instead use self-edges, one for each occurrence of a given place in the document. Consider the following example. In order to generate the graph, they first find the hierarchical parents of the references that are made in the document, the neighboring places to the document references, and the hierarchical parents for these neighboring places. The places discovered through the above procedure would be the nodes of the graph and the relationships between them would be used to produce directed edges between the nodes. For all the nodes with no outlinks (i.e., the roots and the leaf nodes) they add artificial edges to all other nodes in the graph. The part-of, containment, and adjacency edges would all get a value of 0.4, and artificial edges of 0.01. These weights were tuned empirically. The PageRank algorithm would then be applied and, in the end, the highest scoring node would be selected as the scope (Anastácio et al., 2009).

Geocoding

With the term geocoding, we refer to associating geographical coordinates (latitude and longitude) to data, objects or entities which can be geographically annotated or described. In this section, we will review studies that have developed geocoding systems.

One of the first works on document geolocation is (Ding et al., 1999), who attempt to automatically determine the geographic scope of web pages. They focus on named locations, e.g. cities and states, found in gazetteers. Locations are predicted based on toponym detection and heuristic resolution algorithms. A related, recent effort is (Leidner et al., 2003), who geolocate Twitter users by resolving their profile locations against a gazetteer of U.S. cities and training a classifier to identify geographically local words.

An alternative to using a discrete set of locations from a gazetteer is to use information retrieval (IR) techniques on a set of geolocated training documents. A new test document is compared with each training document and a location chosen based on the location(s) of the most similar training document(s). For text, both (Leidner et al., 2003) and (Wing and Baldridge, 2011) use a similar approach, but compute document similarity based on language models rather than image features. Additionally, they group documents via a uniform geodesic grid rather than a clustered set of locations. This reduces the number of similarity computations and removes the need to perform location clustering altogether, but introduces a new parameter controlling the granularity of the grid. (Kinsella et al., 2011) predict the locations of tweets and users by comparing text in tweets to language models associated with zip codes and broader geopolitical enclosures. (Sadilek et al., 2012) discretize by simply clustering data points within a small distance threshold, but only perform geolocation within fixed city limits (Awad et al., 2008).

While the above approaches discretize the continuous surface of the earth, (Eisenstein et al., 2010) predict locations based on Gaussian distributions over the earth's surface as part of a hierarchical Bayesian model. This model has many advantages (e.g. the ability to compute a complete probability distribution over locations), but it will be difficult to scale up to the large document collections needed for high accuracy.

2.1.3 Geolocation Extraction from Event Data

With numerous news reports generated every day, many applications search and organize these daily generated news stories for analysis. These applications include determining crime pattern locations, predicting the place of protests and political unrest, and identifying the geolocation of natural disasters. Such applications can largely benefit from identifying precise geolocation information in a timely manner to provide better support for decision making (Lavee et al., 2007; Masud et al., 2010).

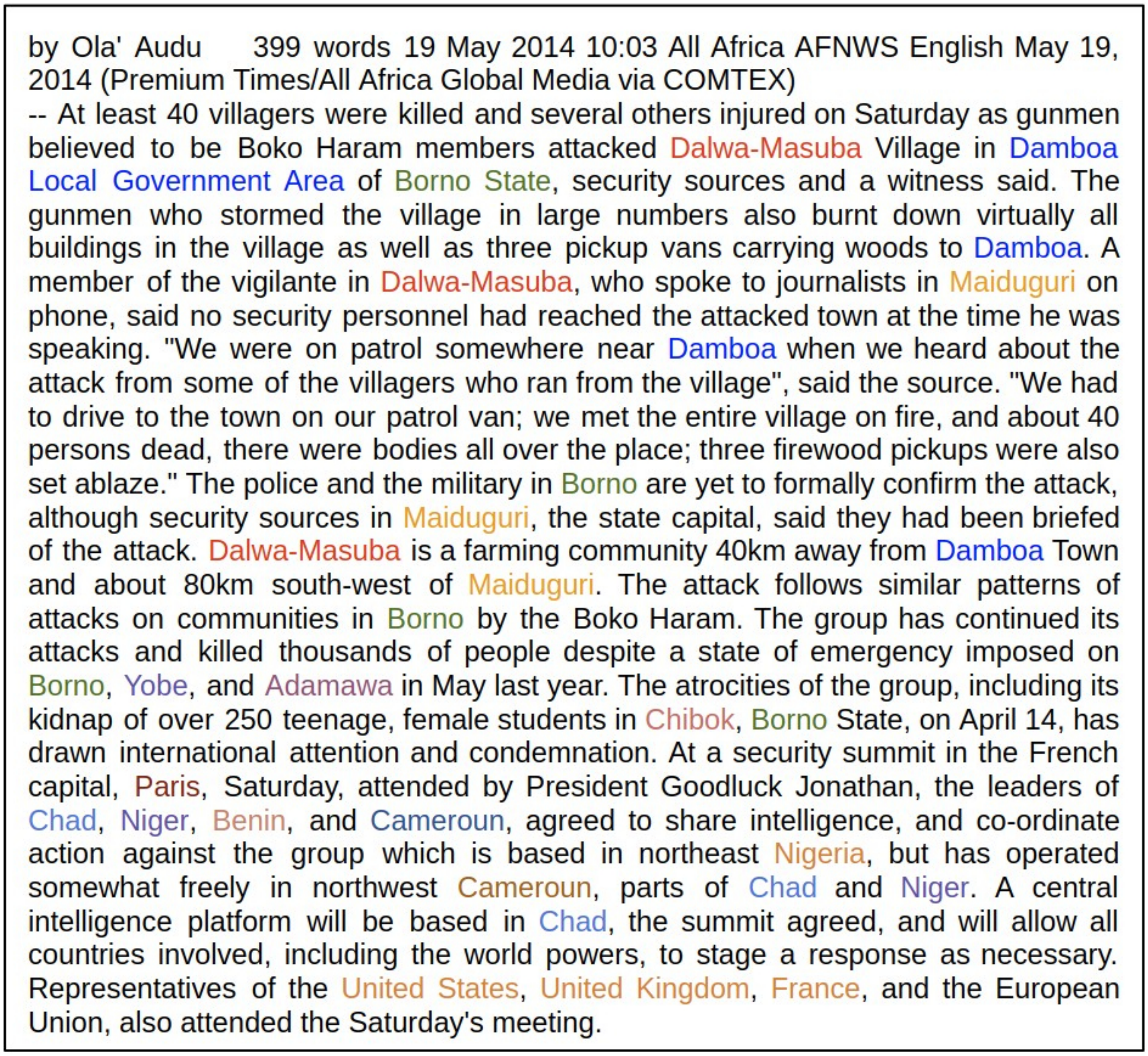
In this study, we focus on identifying the associated locality information of a news article. Typically, the term *Location* is used in a variety of contexts, but the term *Locality* is used to describe a more precise area (D'Ignazio et al., 2014). A news article may contain multiple related localities mentioned in them. These are called *Focus Locations*. However, we aim to identify the place of occurrence of an event. We call this locality *Primary Focus Location*.

For instance, consider the news reports given in Figures 2.1, 2.2[9] and 2.3[10] in English, Spanish and Arabic languages respectively. Figure 2.1 is a report which describes an English atrocity event that occurred in the village of *Dalwa-Masuba*, Nigeria. Moreover, the report also mentions other locations such as Damboa, Maiduguri, Borno, Yobe, Adamawa, Chibok, and Paris. Figure 2.2 is a Spanish atrocity event about killing a villager in *El Caracol*, Colombia. This report also mentions other locations such as Arauca and Venezuela.

---

[9] `http://www.nocheyniebla.org`

[10] `https://alghad.com`



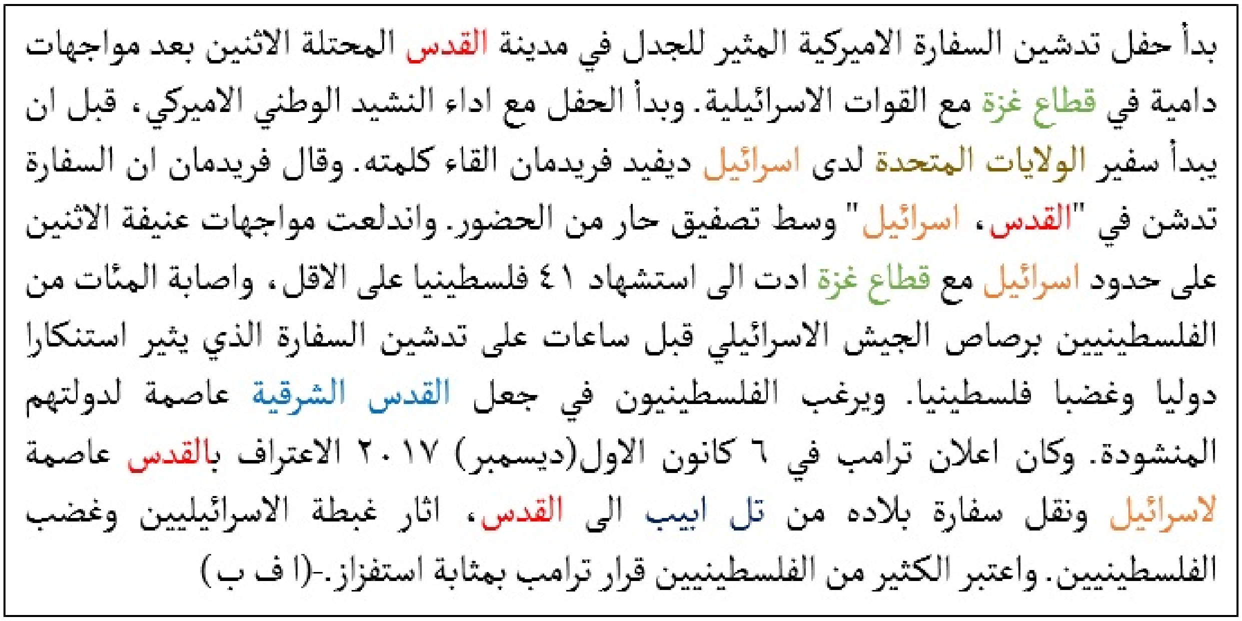
by Ola' Audu 399 words 19 May 2014 10:03 All Africa AFNWS English May 19, 2014 (Premium Times/All Africa Global Media via COMTEX)

-- At least 40 villagers were killed and several others injured on Saturday as gunmen believed to be Boko Haram members attacked Dalwa-Masuba Village in Damboa Local Government Area of Borno State, security sources and a witness said. The gunmen who stormed the village in large numbers also burnt down virtually all buildings in the village as well as three pickup vans carrying woods to Damboa. A member of the vigilante in Dalwa-Masuba, who spoke to journalists in Maiduguri on phone, said no security personnel had reached the attacked town at the time he was speaking. "We were on patrol somewhere near Damboa when we heard about the attack from some of the villagers who ran from the village", said the source. "We had to drive to the town on our patrol van; we met the entire village on fire, and about 40 persons dead, there were bodies all over the place; three firewood pickups were also set ablaze." The police and the military in Borno are yet to formally confirm the attack, although security sources in Maiduguri, the state capital, said they had been briefed of the attack. Dalwa-Masuba is a farming community 40km away from Damboa Town and about 80km south-west of Maiduguri. The attack follows similar patterns of attacks on communities in Borno by the Boko Haram. The group has continued its attacks and killed thousands of people despite a state of emergency imposed on Borno, Yobe, and Adamawa in May last year. The atrocities of the group, including its kidnap of over 250 teenage, female students in Chibok, Borno State, on April 14, has drawn international attention and condemnation. At a security summit in the French capital, Paris, Saturday, attended by President Goodluck Jonathan, the leaders of Chad, Niger, Benin, and Cameroun, agreed to share intelligence, and co-ordinate action against the group which is based in northeast Nigeria, but has operated somewhat freely in northwest Cameroun, parts of Chad and Niger. A central intelligence platform will be based in Chad, the summit agreed, and will allow all countries involved, including the world powers, to stage a response as necessary. Representatives of the United States, United Kingdom, France, and the European Union, also attended the Saturday's meeting.

Figure 2.1: A sample English news report with different place names from Atrocity dataset (Schrodt and Ulfelder, 2009)

Dos hombres armados asesinaron en el corregimiento El Caracol, al comerciante y ganadero de 66 años de edad. Según la denuncia la víctima: Era reconocido por la comunidad araucana por sus servicios y apoyo a los habitantes del corregimiento El Caracol, que se encuentra ubicado a 60 kilómetros al oriente de la capital de departamento de Arauca, en la frontera con la república bolivariana de Venezuela. Gómez Daza, había sufrido dos atentados y recientemente había denunciado ante las autoridades competentes, amenaza a su vida por parte de miembros de la guerrilla del Ejército de Liberación Nacional y paramilitares, en contra de quienes declaró recientemente en audiencia púb lica.

Figure 2.2: A sample Spanish news report with different place names from Revista Noche y Niebla



بدأ حفل تدشين السفارة الاميركية المثير للجدل في مدينة القدس المحتلة الاثنين بعد مواجهات دامية في قطاع غزة مع القوات الاسرائيلية. وبدأ الحفل مع اداء النشيد الوطني الاميركي، قبل ان يبدأ سفير الولايات المتحدة لدى اسرائيل ديفيد فريدمان القاء كلمته. وقال فريدمان ان السفارة تدشن في "القدس، اسرائيل" وسط تصفيق حار من الحضور. واندلعت مواجهات عنيفة الاثنين على حدود اسرائيل مع قطاع غزة ادت الى استشهاد ٤١ فلسطينيا على الاقل، واصابة المئات من الفلسطينيين برصاص الجيش الاسرائيلي قبل ساعات على تدشين السفارة الذي يثير استنكارا دوليا وغضبا فلسطينيا. ويرغب الفلسطينيون في جعل القدس الشرقية عاصمة لدولتهم المنشودة. وكان اعلان ترامب في ٦ كانون الاول(ديسمبر) ٢٠١٧ الاعتراف بالقدس عاصمة لاسرائيل ونقل سفارة بلاده من تل ابيب الى القدس، اثار غبطة الاسرائيليين وغضب الفلسطينيين. واعتبر الكثير من الفلسطينيين قرار ترامب بمثابة استفزاز.-(ا ف ب)

Figure 2.3: A sample Arabic news report with different place names from Alghad news agency

Figure 2.3 is about killing several Palestinian protesters during the opening ceremony of the US embassy at *Jerusalem* in Arabic. This news report mentions various locations such as the Gaza strip, Israel, United States, and Tel Aviv. For Figures 2.1, 2.2 and 2.3, we say that "Dalwa-Masuba", "El Caracol" and "Jerusalem" are respectively the primary focus locations

Table 2.2: Output of focus location extraction from existing tools including Cliff-Clavin, Geoparser, Mordecai, Stanford and MITIE for the given example in Figure 2.1

Tool	**Extracted Country**	**Extracted Locations**	**Focus Country**	**Focus Location**
Cliff-Clavin	NG, FR, TD, NE, BJ,CM,US,IT	Adamaoua, Benin, Borno, Cameroun, Chad, Chibok, Damboa, France, Niger, Maiduguri, Paris, United Kingdom, United States, Yobe	NG	Borno, Damboa Maiduguri
Geoparser	NG, NE, FR, USA AE,CM,TD,BJ	Adamawa,Benin,Borno,Cameroon,Chad,Chibok,European Union,Faransā, Maiduguri, Niger, Nigeria, Paris, United States, Yobe	-	-
Mordecai	NG	Borno, Cameroun-Gbene, Chibok, Dalwa, Damboa, Komadugu, Yobe, Maiduguri	NG	-
Edinburgh	-	Adamawa, Benin, Borno, Cameroun, Chad, Chibok, Damboa, France, Maiduguri, Niger, Nigeria, Paris, United Kingdom, United States, Yobe	-	-
Stanford CoreNLP	-	Adamawa, Benin, Boko Haram, Borno, Chad, Chibok, Cameroun, Dalwa-Masuba Damboa, France, Maiduguri, Niger, United Kingdom, United States, Yobe	-	-
MITIE	-	Paris, Borno state, Damboa, Maiduguri, Niger, Benin	-	-
Profile	NG, FR, TD, NE, BJ,CM,US,IT	Adamawa, Benin, Boko Haram, Borno, Chad, Chibok, Cameroun, Dalwa-Masuba Damboa, France, Maiduguri, Niger, United Kingdom, United States, Yobe	NG	Dalwa-Masuba

since the events occurred in these locations. However, other localities associated with the event which are course-grained form the elements of the focus location set.

Even though several geoparsers such as Cliff-Clavin (D'Ignazio et al., 2014), Mordecai (mordecai, mordecai), and Stanford-CoreNLP (Manning et al., 2014) have been developed to automatically extract named locations from unstructured English text, location extraction from a text is still a challenging task due to the complexity, diversity, and ambiguity of location information in different languages. However, these tools cannot extract the *focus location* with good accuracy, and most of them cannot differentiate between different locations in the text—i.e. focus locality versus non-focus locality—and are not language agnostic. Following the example in Figures 2.1, 2.2 and 2.3, Tables 2.2, 2.3 and 2.4 show the output of these different tools for extracting focus location respectively. Clearly, these tools can identify multiple locations mentioned in the news article. Among them, only Cliff-Calvin is able to identify a few focus locations over the English dataset. But it still cannot identify the desired primary focus location. For Spanish and Arabic languages, to the best of our knowledge there is no tool to extract focus location. Furthermore, Stanford NER and MITIE do not support Arabic. Therefore, we utilize *Polyglot* as a NER tool for Arabic news.

Table 2.3: Output of location extraction from Stanford and MITIE for the given example in Figure 2.2

Tool	Extracted Locations
Stanford	Venezuela
MITIE	El Caracol, venezuela, greenarauca

Table 2.4: Translated outputs of location extraction from Polyglot for the given example in Figure 2.3

Tool	Extracted Locations
Polyglot	Jerusalem, Gaza strip, Israel, United States, Tel Aviv

2.2 Smart Contract Vulnerability Detection

2.2.1 Static Analysis Tools

Static analysis tools rely on the analysis of the contract without executing it. These tools mainly focus on Intermediate Representation, abstract syntax tree (AST), Control Flow Graph (CFG) to extract properties of a code or program. The characteristic of the static tool is it usually performs a high coverage rate. Static methods can cover as many as possible execution paths. Most static analysis tools are rule-based. One of the advantages of these methods is rapid detection speed, which guarantees scalability. However, in the rule-based approaches, if the designed rules do not cover some of the vulnerabilities, it may lead to the high False Positive rate and poor accuracy.

Vandal (Kalra et al., 2018a) is a Static framework that uses low level bytecode to make smart contracts secure. In this framework, the bytecode from Ethereum smart contracts is converted to higher level logic relations. Vandal enables users to identify security problems in the contracts by using declarative logic rules to enumerate the problems, resulting in improved security analysis. Vandal also flags potential security vulnerabilities in the smart contract.

2.2.2 Dynamic Analysis Tools

There are a couple of dynamic analysis tools. One is based on the symbolic execution by constraint solvers, while the other one uses some test cases as inputs of the program and analyzes the results. Some tools may have both characteristics. Regardless of which method they use, these tools have scalability problems due to the long time cost in execution procedure.

OYENTE (Luu et al., 2016) creates a Control Flow Graph (CFG) of the smart contract using the bytecode of the contract as well. However, OYENTE detects security vulnerabilities in a different way. OYENTE detects three security vulnerabilities: transaction ordering dependence, time stamp dependence, and the reentrancy vulnerability by executing the contract and tracing the flow of the smart contract. OYENTE flags contracts that could potentially be insecure.

2.3 Wind Turbine Fault Diagnosis

The wind energy industry has experienced major growth over the last decade. Based on Global Wind Energy Council statistics, the total number of wind turbines across the world has increased from 6,100 wind turbines in 1996 to 3,695,789 wind turbines in 2014 (Pullen and Sawyer, 2014), showing an exponential growth in global demands for wind energy. Wind energy is freely available, does not pollute the environment and needs less space for energy production.

Wind turbines are big, complex and expensive machines which are installed in harsh environments. They are vulnerable to different defects and faults. Manual condition monitoring is cumbersome and costly. A failure in these machines can disable them for several weeks and cost hundred of thousands dollars. Costs due to maintaining failed components affect the price of produced energy as a result. Developing automatic condition monitoring systems for these machines is vital for wind industry. Based on a report by the National Renewable Energy Lab (NREL), gearbox failure causes the longest downtime and is the most costly fault in these machines (Sheng and Veers, 2011). Different factors like poor lubrication, bending fatigue, fretting corrosion and mechanical stresses can cause a defect in a wind turbine gearbox. An early fault detection can prevent catastrophic failures. Monitoring vibration signals picked up from gearboxes is an effective way of condition monitoring. However, by increasing the number of turbines, the number of sensors to monitor turbines will dramatically increase. This will lead to a large volume of data that needs to be processed.

Automated fault detection is difficult and heavy in terms of computational cost. Various studies have been done to solve this problem by using traditional approaches (Kusiak and Li, 2011). Unfortunately, their performance is insufficient since they still require several hours of training in order to learn an appropriate model. Furthermore, they cannot take advantage of live stream data to detect faults in real-time. This prevents technicians from promptly repairing or replacing these devices before they are totally broken.

2.3.1 Model-based and model-free fault diagnosis methods

The two categories of fault diagnosis methods are: model-based and model-free (Imani et al., 2017). Model-based approaches use a mathematical model of the system (Chen and Patton, 2012). Such a model is usually obtained by physical modeling. Having a model reduces the uncertainty regarding measurements, however, physical modeling is challenging. On the other hand, the only assumption that model-free methods make is availability of some training data from the system. Then, signal processing and machine learning is applied to classify the health state of the system (Watson et al., 2010; Imani et al., 2017). Such approaches are easier to generalize compared to other approaches since they do not make any assumptions about the system (Masud et al., 2011).

The majority of prior gearbox fault diagnosis works are model-free methods. However, since vibration signal is a non-stationary random signal, extracting a compact informative feature vector is a challenging signal processing task. The key difference from prior works is in the feature extraction step. Authors in (Zhang et al., 2012; Imani et al., 2019) proposed time domain statistical features. Specifically, they showed that high order statistics like kurtosis and skewness can be employed for fault diagnosis of gearbox. However, these features are very sensitive noise and outliers which limit their applications in real industrial systems. The frequency domain features are proposed in (Watson et al., 2010; Imani et al., 2017). Frequency domain methods are robust and can distinguish between different classes of faults. However, for developing an automatic fault diagnosis system, it is needed to reduce the dimensionality of signal representation. Fourier transform and wavelet transform do not necessarily reduce the dimension of spectrum of signal. Authors of (Harmouche et al., 2015) post-processed the spectrum of signal by PCA for dimension reduction. However, PCA requires more calculations at run-time and can corrupt the spectrum of the signal. So, presenting the spectrum of vibration signal in a compact way is required for developing a robust fault diagnosis system.

On the other hand, the majority of research in this field is devoted to developing signal processing algorithms for fault diagnosis. However, this problem has another challenge for data analytics. Scaling a real-time fault diagnosis algorithm for a wind farm with thousands of turbines requires a massive computational power which is currently available on cloud servers. This dimension of research is related to the emerging field of *Industrial Internet of Things* (IIoT). From this perspective, authors in (Chen et al., 2007; Tamura et al., 2015; Qi et al., 2016; Shao et al., 2016) applied the state-of-the-art Big Data analytics methods to the fault diagnosis problem in industrial systems (Awad and Khan, 2007).

CHAPTER 3

A SCALABLE FRAMEWORK FOR PRIMARY FOCUS LOCATION EXTRACTION[1]

3.1 Introduction

One of the main challenges is to identify the primary focus location among the different candidate locations and from news articles in different languages. We address this challenge by using a supervised classification model that leverages contextual patterns in the occurrences of focus locations regardless of the language. Concretely, we first extract candidate locations using a named entity recognition tool and identify the sentences in which they occur. We then extract semantic features from these sentences by using fastText_multilingual model (Samuel L. Smith and Hammerla, 2017) and sentence embedding approaches (Arora et al., 2017; Mikolov et al., 2013). Finally, we train a classifier on labeled training instances of different languages and then predict the primary focus location on unlabeled test sentences of different languages. We denote this approach as Primary Focus Location Extraction or *Profile* (Imani et al., 2017).

One of the major challenges with the discussed approach is the lack of suitable labeled data instances for training. In the real world, these labeled instances are not readily available, or may be available scarcely. Traditionally, it is assumed that the training and test data sets used for supervised learning methods are generated from the same data distribution and are monolingual. In practice, this assumption may not be true. In our scenario, true

---

[1] This chapter contains material previously published as:
©2017 IEEE. Reprinted, with permission, from Imani, M. B., Chandra, S., Ma, Khan, L., and Thuraisingham, B. “Focus location extraction from political news reports with bias correction.” in IEEE International conference on big data, pp. 1956-196, December 2017.
©2019 IEEE. Reprinted, with permission, from Imani, M. B., Khan, L., and Thuraisingham, B., "Where did the political news event happen?" in IEEE International conference on collaboration and internet computing, December 2019.
Lead author, Imani, conducted the majority of the research, including most of the writing, the full design, the full implementation, and the full evaluation in the both papers.

labels of sentences (focus or non-focus) may be only available from monolingual news articles associated with a single news agency or for a small number of news articles. In such cases, these labeled articles may not be a good representative of the population. For example, news articles from different agencies typically have dissimilar linguistic content, vocabularies, writing styles, or type of emotions (e.g., acted, elicited, or naturalistic). Such differences affect classifier performance when employed to predict focus locations in news articles in the wild, and limits the scalability of our approach (Imani et al., 2019).

We address this challenge by manually labeling a small number of sentences which creates a sampling bias between the training and test data sets. We then leverage the approach of sampling bias correction by using Kernel Mean Matching (KMM) (Huang et al., 2006) to estimate the density ratio between the test and training data distributions to appropriately weight each training data instance and then use these instances to train a classifier for prediction of focus location (Awad and Khan, 2007).

3.2 Background

3.2.1 Geolocation Extraction

The field of geolocation extraction collectively involves many different tasks and analyses to be performed over text. The three main tasks among these are: (i) Location named entity extraction (Finkel et al., 2005; Ritter et al., 2011); (ii) Location named entity resolution (Gritta et al., 2017); (iii) Event's location extraction (D'Ignazio et al., 2014; mordecai, mordecai).

This study revolves around the last task, i.e. using geoparsers to extract event location.

Web page geotagging models such as Web-a-where (Amitay et al., 2004) identify all location names using gazetteer, assign a geographic location and a confidence level to each page, and derive the focus location associated with a web page. Another such framework presented by Silva et al. (Silva et al., 2006) is used to automatically identify geographic scopes from Portuguese web pages. To locate the geographical entities in web pages, this framework utilizes different external sources such as WHOIS and DNS registrars, and the Portuguese postal codes database. Geoparsers such as Geoparser.io (geoparser, geoparser) are hosted as web services that identify place names and handle contextual ambiguity among those places. Cliff-Clavin (D'Ignazio et al., 2014) is an open source geoparser which is also hosted as a web service that parses news articles or other documents. It employs context-based geographic disambiguation over organizations and locations extracted using Stanford CoreNLP from the text. It employs a simple frequency-based method to identify the focus places from places mentioned at city, state, and country levels. Mordecai (mordecai, mordecai) is also an open source geoparser which uses MITIE's NER tool to extract place names from text and then uses gazetteer to identify focus country and all other place names from the text. The Edinburgh Geoparser (Alex et al., 2015) is yet another geoparser designed to identify occurrences of locations from unstructured text and map them to exact latitude and longitude. NewsStand (Teitler et al., 2008) is another geoparser and geotagger tool. NewsStand extracts the "interesting" phrases that are most likely to be references to geographic locations and other entities by using NER methods. LOCATION phrases are stored as geographic features of the entity feature vector. Then, it uses a Gazetteer to find those geographic features in the entity feature vector that are names of actual locations. It also employs Gazetteer to identify the hierarchical information for each location (i.e. country and administrative subdivisions). After that, it extracts *geographic focus* (or focus location) based on the frequency of the locations in the news. The empirical results of our approach are compared with these competing methods.

Each of the discussed geoparsers capabilities in extracting focus location and focus country are presented in Table 3.1. It is observed that Mordecai, Cliff-Clavin and NewsStand are the only geoparsers capable in extracting the focus country. Moreover, Cliff-Clavin also extracts

Table 3.1: Capabilities of different tools in focus country and focus location extraction. Here, ✓ indicates presence and ✗ indicates absence of corresponding capability.

Tools	**NER Extraction**	**Location Extraction**	**Focus Country**	**Focus Location**
Cliff-Clavin	Stanford CoreNLP	✓	✓	✓ (City, State)
Mordecai	MITIE	✓	✓	✗
Geoparser	NA	✓	✗	✗
Edinburgh	rule-based	✓	✗	✗
NewsStand	LingPipe[2]	✓	✓	✓ (Region)

focus location on two different levels, i.e. city and state. Almost all of the above mentioned geoparsers are only able to work on English text.

In this study, we are going to extend our previous research (Imani et al., 2017; Gunasekaran et al., 2018) to other languages, such as Arabic and Spanish. Apart from that, due to the lack of label data in languages other than English, we propose an approach to address this challenge in this study.

3.2.2 Multilingual Word Vectorization

Monolingual word vectors are represented in a high-dimensional vector space, such that two contextually similar words are closer in this space (Mikolov et al., 2013). Since these vectors are monolingual, similar words within a language share similar vectors while translated words from different languages do not have similar vectors. To solve this particular problem, Smith et al. (Samuel L. Smith and Hammerla, 2017) presented a framework using Singular Value Decomposition (SVD) to learn a linear transformation (a matrix), which aligns monolingual vectors from two languages in a single vector space without changing any of the monolingual similarity relationships. This model uses Facebook's fasttext word vectors and then uses the Google translate API to translate these words into the 78 languages available. To place all

78 languages in single space, this model aligned every language to the English vectors (the English matrix is the identity).

An element belonging to a training set is indicated by a subscript tr, while that of a test set is indicated by a subscript te. Each element is indexed by a superscript integer, since a set may contain multiple elements. For instance, $\mathbf{x}^{(i)} \in \mathbf{X}_{tr}$ indicates the i^{th} data instance (array of d covariates) of a training dataset $\mathbf{X}_{tr}$ (a set containing arrays). Also, a hat indicates an estimated value. In general, we use a capital-bold letter to indicate a set of arrays, and a bold letter to indicate an array.

In the case of data classification — a binary focus or non-focus classification in this study — inequality between the probability distributions of the training and test data sets can be represented in the form of the joint probability distributions $p_{tr}(\mathbf{x}, y) \neq p_{te}(\mathbf{x}, y)$, where $\mathbf{x} \in \mathbb{R}^d$ is the d-dimensional covariate of a data instance with class label y. p_{tr} and p_{te} are the training and test probability distribution respectively. According to Ben-David et al. (Ben-David et al., 2010), learning is not possible with bounded error if the two distributions are arbitrarily different. However, this challenge can be addressed by a method that transfers knowledge (model) from training data to test data using instances or feature representation under several assumptions (Pan and Yang, 2010).

One such assumption is the equality in class conditional distribution. Concretely, $p_{tr}(y|\mathbf{x}) = p_{te}(y|\mathbf{x})$. Therefore, the inequality in joint probability distribution is attributed to the covariate distribution, i.e., $p_{tr}(\mathbf{x}) \neq p_{te}(\mathbf{x})$. This is known as *covariate shift*. Overall, a correction to the inequality between $p_{tr}(\mathbf{x})$ and $p_{te}(\mathbf{x})$ is provided by computing an importance weight $\beta(\mathbf{x}) = \frac{p_{te}(\mathbf{x})}{p_{tr}(\mathbf{x})}$ for each instance $\mathbf{x}$. This weighted training data set whose data distribution is equivalent to the test data distribution can be used to train a supervised classifier. Various studies have focused on directly estimating the importance weighting function (or density ratio) rather than computing $p_{te}(\mathbf{x})$ and $p_{tr}(\mathbf{x})$ separately. These include Kernel Mean Matching (KMM) (Huang et al., 2006), unconstrained Least Square Importance Fitting (uLSIF) (Kanamori et al., 2009), and Kullback-Leibler Importance Estimation Procedure (KLIEP) (Sugiyama et al., 2008).

Kernel Mean Matching

The main idea in KMM is to decrease the mean distance between weighted training data distribution $\beta(\mathbf{x})p_{tr}(\mathbf{x})$ and the observed test data distribution $p_{te}(\mathbf{x})$ in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{F}$ with feature map $\phi : \mathcal{D} \rightarrow \mathcal{F}$. The mean distance is determined by computing the *Maximum Mean Discrepancy* (MMD), given by

$$\left\| E_{\mathbf{x} \sim p_{tr}(\mathbf{x})}[\beta(\mathbf{x})\phi(\mathbf{x})] - E_{\mathbf{x} \sim p_{te}(\mathbf{x})}[\phi(\mathbf{x})] \right\| \tag{3.1}$$

where $\|\cdot\|$ is the l_2 norm, and $\mathbf{x} \in \mathbf{X} \subseteq \mathcal{D}$ is a data instance in a dataset $\mathbf{X}$. Here, it is assumed that p_{te} is absolutely continuous with respect to p_{tr}, i.e. $p_{te}(\mathbf{x}) = 0$ whenever $p_{tr}(\mathbf{x}) = 0$. Furthermore, the RKHS kernel h is universal in the domain. It has been proven that under these conditions, minimizing MMD in Equation 3.1 converges to $p_{te}(\mathbf{x}) = \beta(\mathbf{x})p_{tr}(\mathbf{x})$ (Yu and Szepesvári, 2012).

In general, finding desired importance weights by minimizing MMD is equivalent to minimizing the corresponding quadratic program that estimates the population expectation with an empirical expectation. The empirical approximation of MMD (Equation 3.1) to get the optimal solution for $\hat{\beta}(\mathbf{x})$ is given by

$$\hat{\boldsymbol{\beta}} \approx \arg\min_{\beta} \left\| \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \beta(\mathbf{x}_{tr}^{(i)})\phi(\mathbf{x}_{tr}^{(i)}) - \frac{1}{n_{te}} \sum_{j=1}^{n_{te}} \phi(\mathbf{x}_{te}^{(j)}) \right\|^2 \tag{3.2}$$

where $\hat{\beta}(\mathbf{x}) \in \hat{\boldsymbol{\beta}}$. The equivalent quadratic program is as follows.

$$\begin{gathered} \hat{\boldsymbol{\beta}} \approx \underset{\beta}{\text{minimize}} \, \frac{1}{2}\boldsymbol{\beta}^T \mathbf{K} \boldsymbol{\beta} - \boldsymbol{\kappa}^T \boldsymbol{\beta} \qquad (3.3) \\ \text{subject to } \beta(\mathbf{x}^{(i)}) \in [0, B], \forall i \in \{1 \ldots n_{tr}\} \\ \& \, \left| \sum_{i=1}^{n_{tr}} \beta(\mathbf{x}^{(i)}) - n_{tr} \right| \leq n_{tr}\epsilon \end{gathered}$$

Table 3.2: List of Symbols

Symbols	**Description**
$\mathbf{c_s}$	Discourse vector
α	Sentence embedding parameter
γ	Number of the sentences
loc	Location
w	Word
s	Sentence
$\mathbf{v_w}$	Word embedding
$\mathbf{v_s}$	Sentence ebedding
$\mathbf{u}$	First principal component of $\mathbf{v_s}$
$\mathcal{D}$	Domain
d	Number of dimensions
$\mathbf{x}$	Array of covariates (data instance)
y	Class label
p_{tr}, p_{te}	Probability distribution of train/test
$\beta(\mathbf{x})$	Instance weight of $\mathbf{x}$
n_{tr}, n_{te}	Total number of train/test instances
ϕ	RKHS Map
h	RKHS Kernel
ϵ, B, σ	KMM parameters
K, κ	KMM kernel functions

where $\mathbf{K}$ and $\boldsymbol{\kappa}$ are matrices of a RKHS kernel $h(\cdot)$ with $K^{(ij)} = h(\mathbf{x}_{tr}^{(i)}, \mathbf{x}_{tr}^{(j)}) \in \mathbf{K}$, and $\kappa^{(i)} = \frac{n_{tr}}{n_{te}} \sum_{j=1}^{n_{te}} h(\mathbf{x}_{tr}^{(i)}, \mathbf{x}_{te}^{(j)}) \in \boldsymbol{\kappa}$. $B > 0$ is an upper bound on the solution search space, and ϵ is the normalization error. In this study, we utilize the KMM algorithm for bias correction on training data.

3.3 Approach

Table 3.2 lists frequently used symbols in this study. In general, we use a capital letter to denote a set of elements, and a small letter to denote an element of a set. A bold letter indicates an array.

3.3.1 Focus Location Extraction

Figure 3.1 shows an overview of Profile for primary focus location extraction. We first extract candidate focus location by pre-processing the given news reports, and then utilize a supervised classifier to identify a *primary* focus location among them. In the pre-processing step, since focus locations are mostly mentioned in the first few sentences, we choose a user defined number (denoted by γ) of sentences in each news report. Then, we identify the location named entities in the training news report among these first few sentences using Stanford CoreNLP. Next, we extract the sentence features from select sentences that contain locations. If the sentence includes a focus location, we assign a *Focus* label to it; otherwise, we assign a *Non-Focus* label to it. Finally, we train a binary classifier in a supervised manner using this labeled training data (Imani et al., 2019; Al-Khateeb et al., 2012).

On the other hand, after the pre-processing step, in the test phase, we assign a label to each sentence in each report using this model. The labels consists of either “Focus" or “Non-Focus". Labeled sentences are called *focus sentences*. Note that more than one focus sentence is included with each news report. Among candidate location names in the collection of focus sentences for each report, we identify the primary focus location using a frequency-based approach to select the focus location. In particular, the frequency-based approach is as follows. We form a histogram of each location detected by NER tools. The location having the highest count is selected as the focus location.

In the next two subsections, we present the features extracted from a text-based dataset. Then we use our learning method to identify the focus locations from an unstructured text.

Word Embedding

Our feature extraction algorithm is based on using pre-trained word embedding model from raw text. We utilized the publicly available fastText_multilingual (Samuel L. Smith and Hammerla, 2017) which was built with fastText from Facebook and Google Translate API to

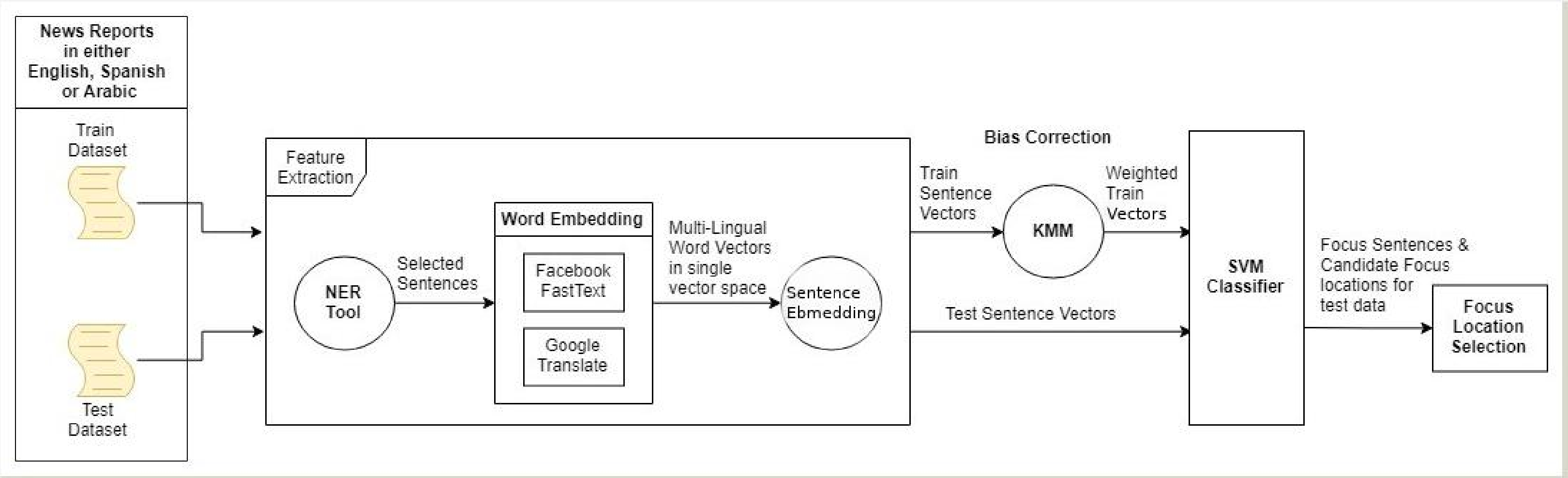


Figure 3.1: A high level schema of Profile (Primary Focus Location Extraction).

align monolingual vectors from two languages in a single vector space. The length of these vectors is 300. We initialize the words that are not present in the set of pre-trained words as zeros. An interesting property of the word embedding is that these vectors effectively encode the semantic meanings of the words in the context. In other words, they are able to represent meaningful syntactic and semantic regularities in a very simple way (Mikolov et al., 2013).

Sentence Embedding

Our basic sentence feature extraction method follows the Sentence Embedding (Arora et al., 2017). We employed this approach because uncommon words are given more weight in the corpus. In other words, common words become less important in the dataset. An alternative approach to find the sentence vector is by computing the mean of the words' vectors in the sentence. We will compare the effectiveness of the Sentence Embedding approach with assigning different weight to each word and the alternative approach empirically in Section 3.4.

Let c_s be a discourse vector, s be a given sentence, S be a set of sentences and α is a scalar. The discourse vector represents "what is being talked about". Assume that $p(w)$ is the unigram probability of a word in a corpus. Given the discourse vector c_s, the probability of a word w in the sentence s is $p(w|c_s)$.

$$p(w|c_s) = \alpha p(w) + (1-\alpha)\frac{exp(< v_w, \tilde{c_s} >)}{Z_{\tilde{c_s}}} \tag{3.4}$$

where

$$\tilde{c_s} = \beta c_0 + (1-\beta)c_s, c_0 \perp c_s$$

$c_0 \in \mathbb{R}^d$ is a common discourse vector which serves as a correction term for the most frequent discourse that is often related to syntax, and $Z_{\tilde{c_s}}$ is a normalizing constant given as follows.

$$Z = \sum_{w \in \mathcal{V}} exp(< v_w, \tilde{c_s} >)$$

So, the likelihood for the sentence s is:

$$\begin{aligned} p(s|c_s) &= \prod_{w \in s} p(w|c_s) \\ &= \prod_{w \in s} \left(\alpha p(w) + (1-\alpha) \frac{exp(< v_w, \tilde{c_s} >)}{Z} \right) \end{aligned} \tag{3.5}$$

where Z is roughly the same as $Z_{\tilde{c_s}}$.

The maximum likelihood estimator (MLE) for $f_w(c_s) = log(p[s|c_s])$ is approximately,

$$\arg\max f_w(\tilde{c_s}) \propto \sum_{w \in s} \frac{a}{p(w) + a} v_w \tag{3.6}$$

where

$$a = \frac{1-\alpha}{\alpha Z}$$

The MLE is approximately a weighted average of the vectors of the words in the sentence. To estimate c_s, we estimate the direction c_0 by computing the first principal component of $\tilde{c_s}$ for a set of sentences. The final sentence embedding is computed by subtracting the first principal component from $\tilde{c_s}$, since we have to omit the effect of a common discourse vector which is often related to the syntax. More details of this method are described in (Arora et al., 2017).

The process of feature extraction by using sentence embedding is summarized in Algorithm 1. The inputs of the algorithm are News_Reports, focus_locations, Word_Embedding, and Parameters a and γ. In the first For-loop of the Algorithm (line 1 to 11), we extract set of the locations (loc) by using Stanford CoreNLP as a named entity recognizer (NER) for each news report (line 3), and exclude countries' name from them in line 4. Then, we select the first γ sentences for each news report which contain at least one location name (line 6 to 10). In the next for-loop, we compute the sentence embedding vector ($\mathbf{v_s}$) for each sentence, based on equation 3.6 (line 12 to 14). For more frequent words w, the weight $\frac{a}{a+p(w)}$ is smaller, so this leads to smaller weights for frequent words. Finally, we compute the first principal component u and decrease it from sentence vector $\mathbf{v_s}$ (line 16 to 18). We trained the SVM classification model using the extracted feature vectors.

We apply the same algorithm during the test process,. However, in the first for-loop, we just use the locations extracted by using Stanford CoreNLP ($loc \leftarrow \mathrm{NER}(\mathrm{News}_i)$). Then, we classify the feature vectors by using the model. Since there may be more than one Focus sentence per report (i.e., sentence containing potential focus location), we extract the locations from Focus sentences. Next, we use the frequent-based approach to extract the Focus locality. In frequency-based approach, we select the most frequent item in the list. In other words, if we find several sentences from one article with a focus label, the most frequent location name will be a candidate for focus location.

As mentioned earlier, in Section 3.1, we may not have sufficient labeled data to train an unbiased classifier. In such a case, we employ the following approach for bias correction over training data. From the given biased training data, we first perform pre-processing steps by extracting feature vectors. Then over these feature vectors, we apply the bias correction method. Particularly, using KMM we compute instance weight for each training data. This estimates density ratios with the given test data instances. We then train a suitable classifier using the weighted training data in RKHS. This classifier is used to predict focus location over test focus sentences.

3.4 Experiments

In this section, we first explain the dataset used to evaluate the proposed method to extract focus locations, and then present the evaluation results while comparing it with the other competing methods.

```
Data:
News_Reports $\text{News}_1, ..., \text{News}_N$ , focus_locations $\text{Floc}_1, ..., \text{Floc}_N$, Word_Embedding $\{\mathbf{v_w} : w \in \mathcal{V}\}$, Parameter $a$ and $\gamma$
Result: Sentence_Embedding $\mathbf{v_s}$
for each News_Report ($\text{News}_i$) do
    /* Extract the Location Named Entities for $\text{News}_i$ by using a NER tool */
    $loc \leftarrow \text{NER}(News_i) \cup Floc_i$
    $loc \leftarrow \text{loc} \setminus \text{Country\_Names}$
    /* Select the first $\gamma$ sentences which contain a location in loc. $S$ is a list of these sentences ($s$). */
    for each sentence ($s$) do
        if ($\#s <= \gamma$ and $\exists i : \text{loc}_i \in s$) then
            $S \leftarrow S \cup s$
        end
    end
end
for each sentence $s_i$ in $S$ do
    $\mathbf{v_{s_i}} \leftarrow \frac{1}{|s_i|} \sum_{w \in S} \frac{a}{a+p(w)} \mathbf{v_w}$
end
/* Compute the first principal component $u$ of $v_{s_i}$ */
for each Sentences $s_i$ in $S$ do
    $\mathbf{v_{s_i}} \leftarrow \mathbf{v_{s_i}} - \mathbf{u}\mathbf{u}^T \mathbf{v_{s_i}}$
end
```

Algorithm 1: Feature Extraction in Profile using Sentence Embedding

Table 3.3: Dataset Statistics

Lang	**Dataset**	**# News Reports**	**# Sentences**
English	Atrocity Event Data	3.6K	40K
	New York Times	1K	103K
Spanish	nocheyniebla	1.5K	8K
	Protest	200	1.5K
Arabic	Alghad News	8.2K	75K

3.4.1 Dataset

The Atrocities Event Data (Schrodt and Ulfelder, 2009) is a collection of recent English news reports on atrocities and mass killings in several locations. Human coders have read the reports and extracted metadata about the events reported. The annotated reports include victims, focus location, and the reports that reported the event. For the training and testing dataset, we excluded the reports that contain multiple events. Moreover, we only select the reports whose locations were correctly extracted by different NER's such as Stanford and MITIE since the performance of NER is beyond the scope of this study. The original size of Atrocity dataset is about $15K$ reports, and almost $5K$ of them are annotated.

Another English dataset that we used is the New York Times (NYT)[3] news reports dataset. The New York Times Annotated Corpus includes more than 1.8 million articles composed and published by the New York Times between January 1, 1987 and June 19, 2007 with article metadata. Similar to the Atrocity Event dataset, we only select political news articles that contain special keywords such as kill, die, injure, dead, death, wounded and massacre in their title. Although the NYT corpus includes location annotations, all of them are not focus locations. Accordingly, we randomly selected 1000 news reports and manually tagged them.

Noche y Niebla[4] is a Spanish dataset with location label (Municipio) for each news report in a PDF format. We developed a PDF extractor to extract news from year 2000 to 2017 in a more structured format[5]. After this process, we removed news reports whose focus locations are not explicitly mentioned in the text. To evaluate the bias correction method, we also asked Spanish-speaking coders to annotate around 200 Spanish protest news articles.

To create an Arabic dataset, we used *Alghad*[6] which is an Arabic news agency. Each news report was initialized with a focus location. We crawled this website to extract news and their focus locations. We used two different categories of news reports, i.e. news of

---

[3] https://catalog.ldc.upenn.edu/ldc2008t19

[4] `http://www.nocheyniebla.org/?page_id=399`

[5] https://launchpad.net/pdf2xml/+download

[6] https://alghad.com/

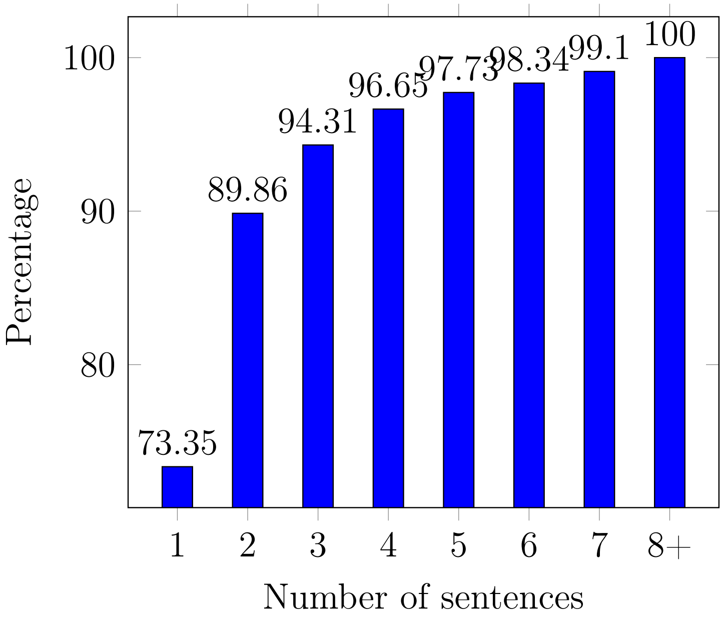


Figure 3.2: Percentage of documents containing focus location in the initial set of sentences.

Arabian countries (3K) and World news (5.2K), to test the bias correction method. The overall number of news reports and sentences for English, Spanish and Arabic corpus are given in Table 3.3.

The experiments were conducted on an Intel machine having Core-i7 3.40GHz CPU with 64 GB of RAM, running a standard Ubuntu Linux version 16.04 LTS. We also set $a = 0.1$ and $\gamma = 7$ as inputs for Algorithm 1 as default. We choose $\gamma = 7$ (first seven sentences of any news reports are selected as in input to the algorithm) since we observed that primary focus locations were present in the first 7 sentences of the training set in more that 99% of news reports (Imani et al., 2017; Gunasekaran et al., 2018). This is illustrated in Figure 3.2. The chart demonstrates that focus location in 73% of articles can be found in the first sentence, and less than 1% of articles contain location information in sentences that occur after the 8^{th} sentence. We denote this sentence filtering by Profile$_s$.

Profile uses a support vector machine (SVM) with a Radial Basis Function (RBF) kernel as a base classifier since it supports weighted training data in RKHS. Here, SVM parameter values are $c_{svm} = 1000$ and $\gamma_{svm} = 0.1$.

We compare the performance of Profile with other tools for each of the languages. For English, we used Cliff-Clavin and the frequency-based approach to extract focus locations from Stanford-CoreNLP. Since Stanford-CoreNLP was only developed to identify named entities such as person and location names, it does not distinguish between different levels of location, such as locality and country. Therefore, we modified the Stanford-CoreNLP output and excluded country names from the resulting location names, and then used a frequency-based approach to obtain the most frequent location name as a surrogate for primary focus location. For Spanish, we employed a similar modified Stanford-CoreNLP and MITIE with the frequency-based approach. We utilized Polyglat with the same frequency-based approach for the Arabic dataset. To train the model for each dataset, we randomly picked 60% of articles as training data, and used the remaining 40% of news reports as the test dataset. As shown in Figure 3.1, Profile first performs the pre-processing steps on both training and test datasets to extract sentences containing potential primary focus location. Then, it extracts related features from the text.

The above experiments assume that the training and test data occur from the same agency or topic. However, a more practical scenario for focus location identification is to study a setting where biased training data is available. We generate a training bias by selecting training data which contains articles from one agency/topic, while the test data contains data from another agency/topic. We utilize the KMM method to obtain an instance weight for each of the training data and build an SVM classifier using the weighted training data. We denote this by $\text{Profile}_s^{\text{KMM}}$. For comparison, we also train another SVM classifier without any weight correction. We denote this by $\text{Profile}_s^{\text{SVM}}$. We then evaluate these classifiers on the same test dataset (Awad et al., 2004).

Table 3.4: Primary focus location accuracy comparison between different methods

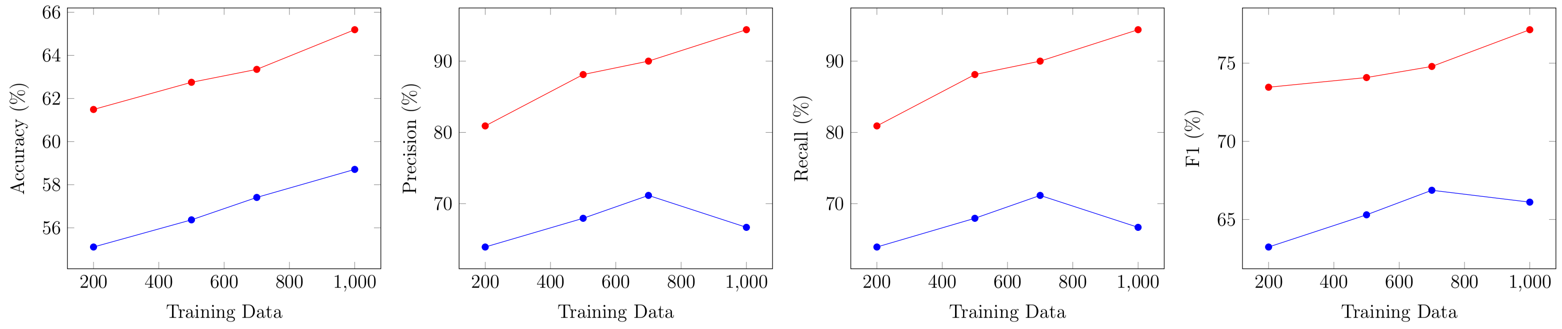
Dataset	**Method**	**Accuracy** (%)
English (Atrocity)	Profile$_s$	**71.27**
	Cliff-Clavin	63.75
	Modified Stanford-CoreNLP	60.83
English (NYT)	Profile$_s$	**64.21**
	Cliff-Clavin	53.65
	Modified Stanford-CoreNLP	36.25
Spanish (Nocheyniebla)	Profile$_s$	**66.63**
	Modified MITIE	38.44
	Modified Stanford-CoreNLP	29.75
Arabic (Alghad)	Profile$_s$	**62.41**
	Modified Polyglot	34.43

3.4.2 Focus Location Extraction Results

We now present the results of Profile where the training and test data occur from the same agency. On average, there are more than five different location names per news report. Note that our goal is to determine the primary focus location while the rest are non-focus locations.

Next, we compare the classification performance of Profile$_s$ with other existing location estimation approaches. The results are presented in Table 3.4 for English (Atrocity and NYT), Spanish and Arabic datasets. In this table, Cliff-Clavin has better accuracy than modified Stanford CoreNLP for English since it is able to extract the focus country and exclude place names which are not in the focus country. The Profile$_s$ significantly outperforms modified Stanford CoreNLP and MITIE in Spanish dataset. The proposed approach also surpasses the modified Polyglot approach for an Arabic dataset.

The proposed approach worked better than the existing methods, since we utilize FastText and the sentence embedding model which encoded word semantics and relationships between words in a sentence. Cliff-Clavin can extract locations at a more coarse-grained level based on the dictionary, and it uses the frequency-based approach to identify the focus locations. As a result, Profile$_s$ outperforms the other methods with 71.27% for Atrocity and 64.21% for NYT. In addition, NER tools are not able to extract focus location at the locality level. To the



(a) Accuracy (b) Precision (c) Recall (d) F1

Figure 3.3: Performance of bias-corrected classifier —●— Profile$_s^{\mathrm{KMM}}$ with Atrocity dataset as training and NYT dataset as test, compared to a biased classifier —●— Profile$_s^{\mathrm{SVM}}$.

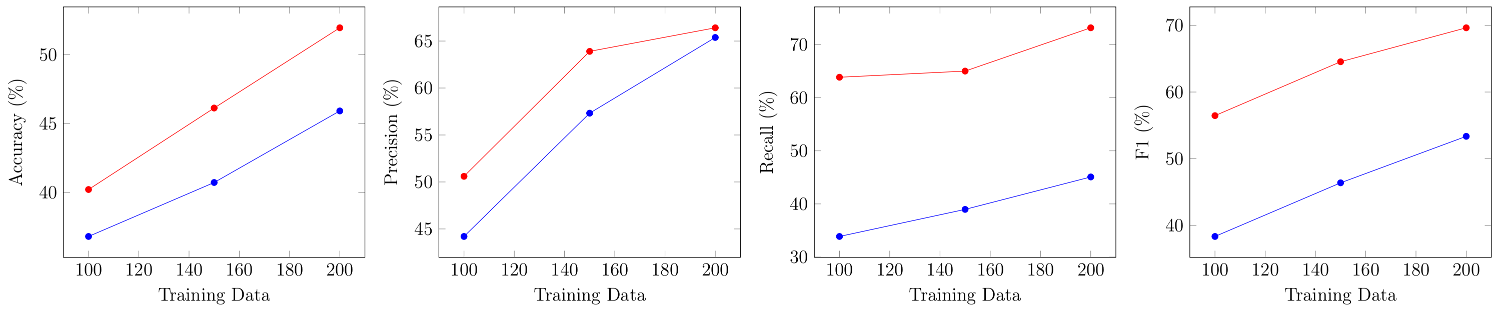
best of our knowledge, there is not any geoparser that is able to extract focus location from Spanish and Arabic text. Furthermore, most of the popular NER tools are not applicable for Arabic language.

3.4.3 Intra-Language Bias Correction Results

Here, we assume the training and test data are from two different publishers/sections in the same language and are related to atrocity news. Figure 3.3 presents the performance of the $\text{Profile}_s^{\text{KMM}}$ model for focus location extraction in English with Atrocity Event data as the training set and NYT as the test set. Similarly, we also considered Spanish Protest as the training set and *Noche y Niebla* Event Data as the test set. The result is shown in Figure 3.4 with different sets of randomly selected training data size, following (Huang et al., 2006). For Arabic language, we use World news as the training and Arabian news reports as the test dataset. The Arabic result is illustrated in Figure 3.5.

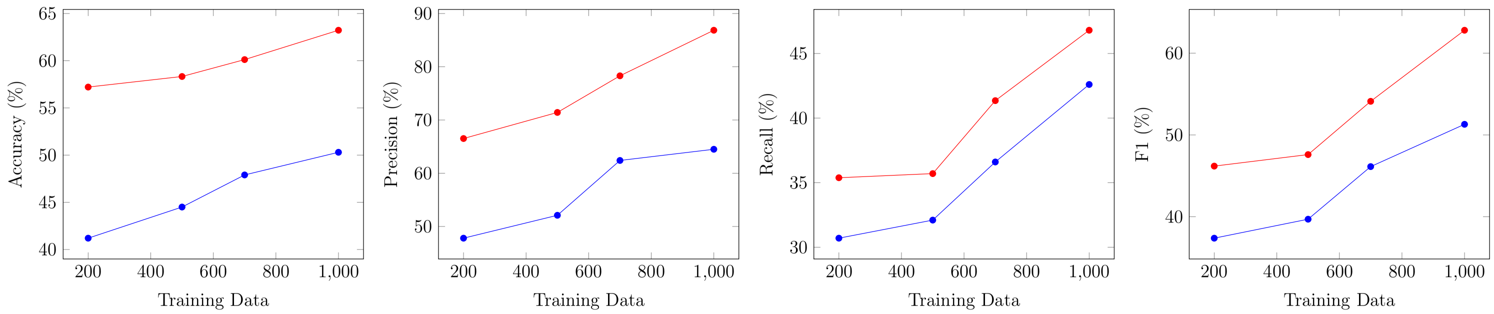
The main conclusions from these three figures are as follows.

- We consistently achieved the best adaptation performance for different training sizes from all experiments on $\text{Profile}_s^{\text{KMM}}$.
- Based on accuracy and precision, we see that $\text{Profile}_s^{\text{KMM}}$ performs similar to the baseline systems. However, $\text{Profile}_s^{\text{KMM}}$ achieves considerably better recall and F1-measure.
- Overall, the $\text{Profile}_s^{\text{KMM}}$ method achieved higher performance than $\text{Profile}_s^{\text{SVM}}$ in all of the experiments.
- Selecting fewer training instances introduced more bias for these domains. As a result, $\text{Profile}_s^{\text{KMM}}$ significantly outperformed the baseline method when the bias is more.
- Finally, the proposed approach works better for English datasets. One of the main reasons is that those datasets are manually selected and labeled. However, this assumption is not true for the Arabic dataset. Since we automatically extracted the news, the data can be noisy or the labels may not have been verified. In addition to this, since



(a) Accuracy (b) Precision (c) Recall (d) F1

Figure 3.4: Performance of bias-corrected classifier —●— $\text{Profile}_s^{\text{KMM}}$ with Spanish Protest dataset as training and Nocheyniebla dataset as test, compared to the biased classifier —●— $\text{Profile}_s^{\text{SVM}}$.



(a) Accuracy (b) Precision (c) Recall (d) F1

Figure 3.5: Performance of bias-corrected classifier —●— $\text{Profile}_s^{\text{KMM}}$ with Arabic world news dataset as training and Arabic Arabian news dataset as test, compared to the biased classifier —●— $\text{Profile}_s^{\text{SVM}}$.

we utilized an NER tool at the first step, the performance of the NER tool can affect the performance of Profile.

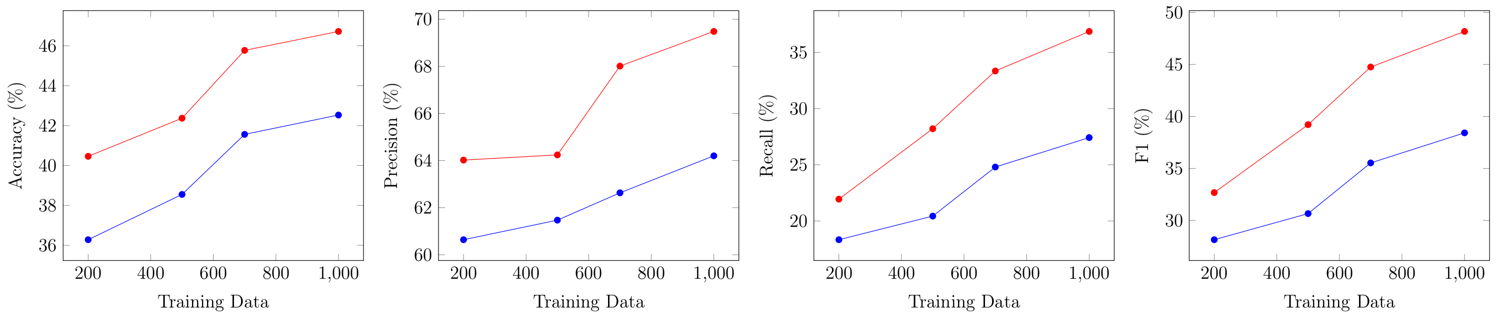
3.4.4 Inter-Language Bias Correction Results

In this section, we train our model with English Atrocity data and then use this model to test on Spanish and Arabic datasets.

Figure 3.6 presents the performance of the $\text{Profile}_s^{\text{KMM}}$ model which was trained with English Atrocity Event data and tested on Nocheyniebla Spanish Event Data to predict the primary focus location. Similarly, we tested the model considering Arabic dataset as the test set. The results are shown in Figure 3.7 with different sets of randomly selected training data sizes.

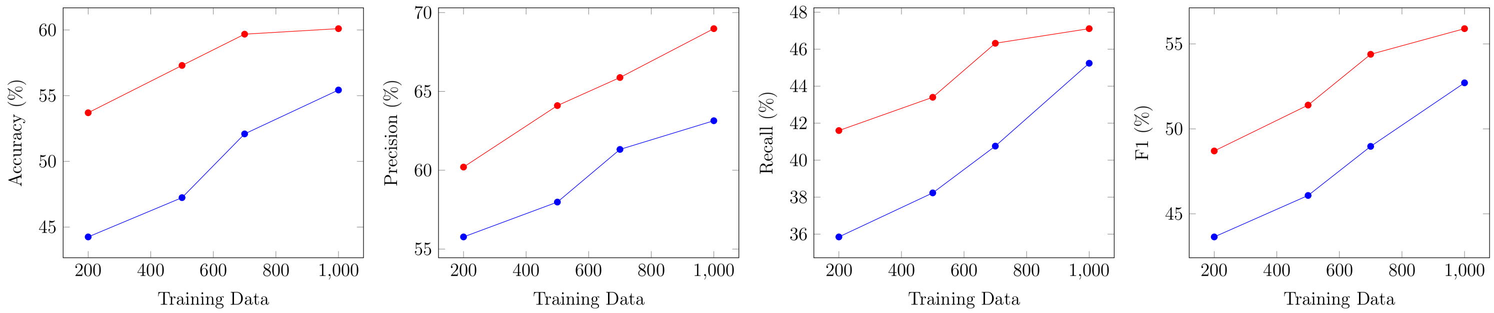
The main conclusions from these figures are as follows.

- For accuracy, precision, recall and F1-measure, we consistently achieved the best adaptation performance for different training sizes from all experiments on $\text{Profile}_s^{\text{KMM}}$ when compared to baseline systems.
- Based on these results, we can leverage the availability of labeled data in one language (such as English) to train the model and apply it on other languages to find the focus location, since labeled instances are not always readily available, or may be available scarcely in different languages.



(a) Accuracy (b) Precision (c) Recall (d) F1

Figure 3.6: Performance of bias-corrected classifier —●— $\text{Profile}_s^{\text{KMM}}$ with English Atrocity dataset as training and Spanish Nocheyniebla news dataset as test, compared to the biased classifier —●— $\text{Profile}_s^{\text{SVM}}$.



(a) Accuracy (b) Precision (c) Recall (d) F1

Figure 3.7: Performance of bias-corrected classifier —● $\text{Profile}_s^{\text{KMM}}$ with English Atrocity dataset as training and Arabic news dataset as test, compared to the biased classifier —● $\text{Profile}_s^{\text{SVM}}$.

CHAPTER 4

A SCALABLE FRAMEWORK FOR VULNERABILITY DETECTION IN SMART CONTRACTS[1]

4.1 Introduction

The increase in the adoption of decentralized cryptocurrency systems has resulted in deployment of smart contracts to solve various problems such as decentralized electronic fund management systems. Smart contracts are autonomous applications that execute on the blockchain in a decentralized way. With smart contracts, arbitrary computations can be performed in addition to transaction-based systems. Using blockchain, different entities can interact without a need for a centralized authority. Smart contracts can be used all across the chain from financial services to healthcare to insurance (Griggs et al., 2018; Cohn et al., 2017; Buterin et al., 2014). Due to the rapid growth in their deployments, various vulnerabilities have been exploited by attackers to steal cryptocurrencies worth millions of dollars.

Smart contracts are commonly written in a high-level language such as Ethereum's Solidity, and translated to low-level bytecode for deployment on the blockchain. Smart contracts can be executed usually by a virtual machine. In order to run smart contracts, fees are paid to miners to execute the smart contracts. Using ABI (Application Binary Interface), users can interact with a deployed smart contract to complete tasks. Since smart contracts are computer programs, various vulnerabilities have been discovered and exploited by attackers for financial gain.

These security vulnerabilities in smart contracts originate from a wide range of issues including deficient programming methodologies, languages design issues, and toolchains,

---

[1] The work presented in this chapter was performed in collaboration with L. Khan, and B. Thuraisingham at the University of Texas at Dallas. This work is currently submitted for publication. Lead author, Imani, conducted the majority of the research, including the full writing and design, the most of the implementation, and the evaluation.

buggy compilers such as those in the Solidity compiler[2]. Some of these vulnerabilities include transaction-ordering dependence, timestamp dependence, mishandled exceptions, reentrancy vulnerability, unsecured balance, destroyable contract, and stack-overflow (Luu et al., 2016; Brent et al., 2018).

Ethereum (Buterin et al., 2014) and Cardano[3] are two types of blockchains with smart contract capabilities. Ethereum has become the *de facto* standard platform for smart contract development, with a market capitalization of $19,854M (USD)[4] by the beginning of 2020. Due to this reason, we focus particularly on Ethereum smart contracts in this study.

Previous works in smart contract defense focused on discovering vulnerabilities in smart contracts. Oyente(Luu et al., 2016), Osiris (Torres et al., 2018), TeEther(Krupp and Rossow, 2018) and Zeus(Kalra et al., 2018b) performed vulnerability discoveries by leveraging symbolic execution, Z3 solvers (De Moura and Bjørner, 2008) and source code analysis by pre-defined rules. Symbolic execution has a scalability problem because of the long time cost in execution procedure. On the other hand, rule-based systems rely on human experts to define features or different rules to detect bugs in the programs. In this study, we propose a static analysis framework based on a machine learning model to address the aforementioned issues.

4.2 Background

In this section, we provide background information on the Ethereum virtual machine. After that, we discuss the various vulnerabilities present in Ethereum smart contract applications.

---

[2] https://solidity.readthedocs.io/en/latest/bugs.html

[3] https://www.cardano.org/en/home/

[4] https://coinmarketcap.com/currencies/ethereum/

4.2.1 Ethereum Virtual Machine

The Ethereum virtual machine (EVM) is one of the popular implementations of a smart contract framework. The EVM is a stack-based computer that takes as input a sequence of bytecode instructions which is executed to complete a task. The EVM has a key-value persistent store, a memory and a stack of 32-byte values. The EVM bytecode consists of more than 100 opcodes, such as `SUB`, `PUSH`, and `ADD`. Each opcode requires a particular set of fees in order to execute the instruction. The fee varies based on the function of the instruction. Listing 4.1 shows a sample EVM bytecode to perform addition and subtraction. In addition, fees are paid to persist data in the smart contract persistent store. To prevent denial of service attacks, a fee is paid before an instruction is executed. This ensures that attackers cannot execute their malicious code as this results in wastage of attackers resources (Nessa et al., 2008; Masud et al., 2016; Thuraisingham et al., 2008).

Furthermore, Ethereum has two types of token standards which includes ERC20 and ERC721. The ERC20 is a token standard that defines an interface to create custom cryptocurrencies while ERC721 is a standard that defines interface for creating non-fungible assets used in representing unique objects on the Ethereum blockchain, such as was used for CryptoKitties.

```
00000  PUSH1  0x01
00002  PUSH1  0x05
00004  SUB
00005  PUSH1  0x01
00007  PUSH1  0x05
00009  ADD
```

Listing 4.1: Addition and subtraction program in EVM bytecode

4.2.2 Common Ethereum Smart Contract Vulnerabilities

Reentrancy Reentrancy is a well-known vulnerability with a recent TheDAO[5] hack. In Sept of 2016, hackers stole over 3,600,000 Ether, or 60 million US Dollars at the time by exploiting a Reentrancy vulnerability in the contract code from the company TheDAO. Since then, developers have changed codes to limit the function being accessed from malicious code. However, still a huge number of old contracts are vulnerable to the Reentrancy approach.

The Reentrancy vulnerability starts from a function named *fall-back* function. Every contract can only have one fall-back function. The fall-back function has no arguments and no return values. This function will be called in two ways: First, when a contract receives a function call but no functions match; Second, when a contract receives Ethers. In Ethereum, when a contract calls another, the current execution waits for the call to finish. This can lead to an issue when the callee makes use of the intermediate state the caller is in. In other word, a malicious contract will write an external function call in its fall-back function (Yen et al., 2002).

Time Dependency Many applications in Ethereum require obtaining the timestamp to decide what operations to do. Block timestamp is typically used in two situations: as a deterministic random seed and as a global timestamp in a distributed network, such as Castle[6] and GovernMental[7]. TheRun contract[8] is an example of the former situation. TheRun contract uses the current timestamp in order to generate random numbers and award a jackpot based on the result. The timestamp of the contract is dependent on block timestamp, and the block timestamp must be agreed by block miners. Due to the synchronization, all

---

[5] https://etherscan.io/address/0xbb9bc244d798123fde783fcc1c72d3bb8c189413#code

[6] http://protect-the-castle.ether-contract.org/

[7] http://governmental.github.io/GovernMental/

[8] https://etherscan.io/address/0xcac337492149bdb66b088bf5914bedfbf78ccc18#code

contracts in a same block share the same timestamp. When a miner gets a new block, it has an ability to adjust the born time of the new block. This could be 900 seconds in early Solidity versions. Although it is changed to several seconds now, it still has a timestamp vulnerability.

Mishandled Exceptions There are some ways for a contract to call another, including sending an instruction or calling a contract's function directly (e.g. ContractA.FunctionB()). Meanwhile, exceptions can be raised in callee contracts because of not enough gas, exceeding call stack limit, division by zero, array index out of bound and so on. If the exception happens, the callee contract terminates, reverts its state and returns false. 27.9% of the contracts do not check the return values after calling other contracts via send (Luu et al., 2016). A malicious caller contract can cause the send to fail deliberately, regardless of what the callee does. Sometimes the attacker does not receive any direct benefit other than causing other users to lose their entitlement. But, in some other examples, the attacker can get the direct benefit (e.g. GovernMental[9]). These contracts pay investors interest for their investments from the subsequent investments by others. An attacker can first invest his money. Then, he can cause the failure in the previous investors, so he can receive his interest earlier!

Transaction-Ordering Dependence Since a transaction is in the mempool for a short while, one can know what actions will occur before it is included in a block. In the Ethereum Blockchain network, only miners can decide the order of transactions in a block. Therefore, the final state of a contract depends on the miner's decision about ordering the transactions.

In this situation, the miners can cause malicious behavior, for example, a smart contract offering a prize for providing the right solution to some problem. The contract owner can modify the prize as long as it has not been awarded, and users can submit their solutions to

---

[9] http://governmental.github.io/GovernMental/

the problem to win the prize. On the other hand, the owner can track all submissions and manage the ordering of transactions. Let's consider the scenario when there is an unprocessed user transaction with a valid submission; the miner has the ability to check it out and then submit his own transaction thus reducing the value of the prize to zero. In such a case, the owner submission will be invoked first. And, the other user not only will not win the prize, but he has also submitted the solution for free!

Critical Instruction Vulnerabilities EVM bytecode consists of instructions that allow external smart contracts to call other smart contract code. These instructions include the CALL, CALLCODE and DELEGATECALL instruction as shown in Figure 4.1. The CALL instruction allows a caller smart contract to call a target smart contract which can modify the storage or variables in the target contract. CALLCODE and DELEGATECALL allow a caller smart contract to call a target smart contract and modify the variable and storage in the caller's context. With these capabilities, attackers can inject code that can modify a smart contract's balance, change ownership or even self destruct a smart contract and withdraw all the tokens in the contract.

Smart Contract Owner Hijack Each smart contract has an owner when deployed. For example, (Breidenbach, Daian, Juels, and Sirer, Breidenbach et al.) show a code snippet for setting up a smart contract wallet for the Parity Ethereum Wallet from its github repository. The *initWallet* method is used to set the owner of the wallet. The *initWallet* method should only be executed once in the lifetime of the smart contract. After initiating a smart contract, allowing changes to the ownership of a contract through a call to *initWallet* can allow an arbitrary user to use methods such as delegate call to change the ownership of the contract. With the DELEGATECALL instruction, an attacker can pass a malicious payload to execute in the attacker's context. In this case, the attacker modified the owners of the smart contract by executing the *initWallet* method. After changing the owner of the wallet contract, the attacker is able to kill the contract which results in all the funds in the wallet being withdrawn to the attacker. High-value attacks on the Parity Multisig smart contract (Breidenbach, Daian, Juels, and Sirer, Breidenbach et al.; Krupp and Rossow, 2018) were completed using this vulnerability to hijack the ownership of the smart contract.

Arithmetic Vulnerabilities Arithmetic vulnerabilities such as integer overflow and underflow can occur during EVM code execution leading to loss of tokens or money. The stack consists of up to 1024 32-byte words which can hold a maximum value of 2^{256}. By adding a number to the max value, the new value rolls over to zero. Likewise for integer underflow, by subtracting from a zero value, the value rolls over to the maximum value (Consensys, Consensys)[10] because EVM uses unsigned int256 types (Konstantopoulos, Konstantopoulos). Integer underflow vulnerabilities can allow an attacker to roll over his initial balance to the maximum value, thereby gaining access to a large token balance which he does not own.

4.3 Proposed Framework

Figure 4.2 shows an overview of the proposed framework for vulnerability detection in smart contracts. In the following subsections, we will discuss this framework in more detail.

4.3.1 Smart Contract Collection

In the first step of our proposed framework, we query the Ethereum blockchain repository to extract the bytecode representation of smart contracts. In this step, we also discard duplicated smart contracts.

Let's assume that Fibonacci and Factorial programs are developed in Solidity smart contract codes as presented in Algorithm 2 and Algorithm 3, respectively. We will use these smart contracts as examples in the next steps. In the first step, we capture the machine code

---

[10] https://github.com/CoinCulture/evm-tools/blob/master/analysis/guide.md

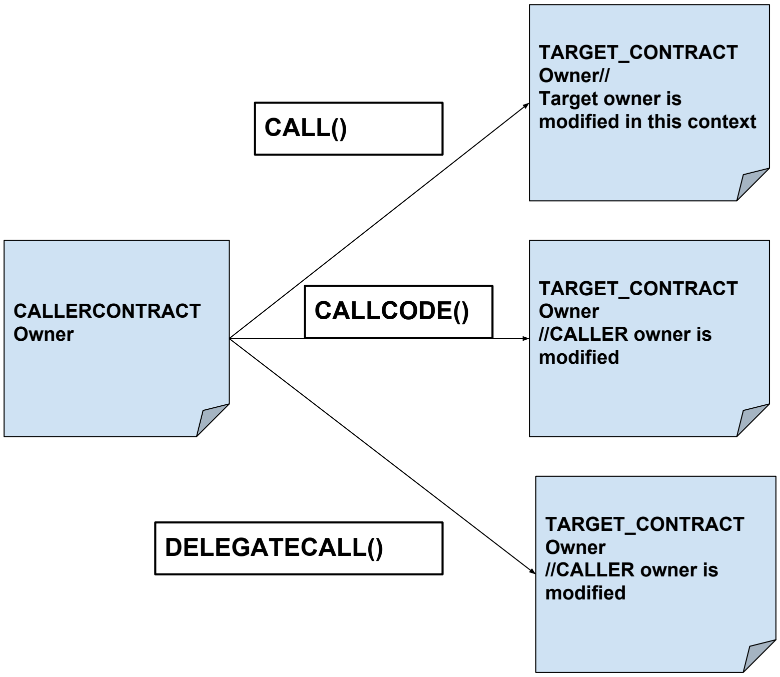


Figure 4.1: Smart contract CALL flow

representation of the mentioned smart contract in the form of bytecode, producing a file such as the ones shown in Figure 4.3 and Figure 4.4.

```
contract Fibonacci {
  function fib(uint n) returns(uint y) {
    if (n <= 1) {
      return n;
    } else {
      return this.fib(n - 1) + this.fib(n - 2);
    }
  }
}
```

Algorithm 2: High-level Solidity Fibonacci code example

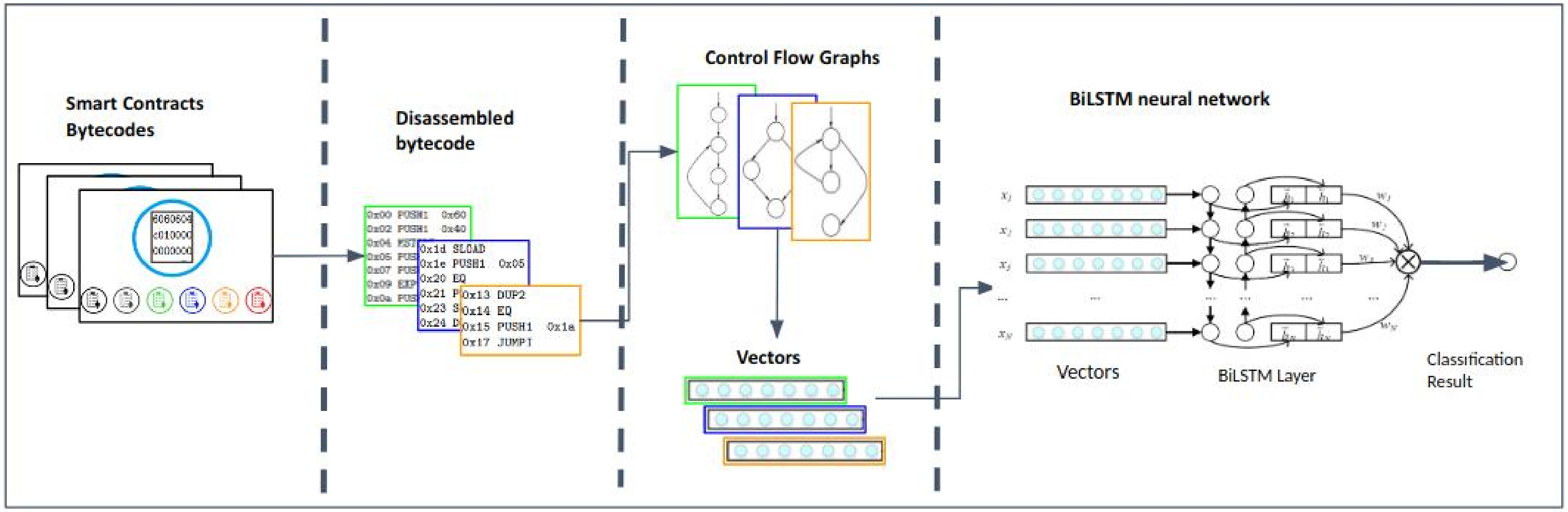


Figure 4.2: Smart contract vulnerability detection proposed framework

```
contract Factorial {
  function fact(uint n) returns(uint y) {
    if (n == 0) {
      return 1;
    } else {
      return this.fact(n - 1) * n;
    }
  }
}
```

Algorithm 3: High-level Solidity Factorial code example

```
0x606060405260e060020a6000350463c6c2ea178114601c575b
6002565b3460025760296004356042565b604080519182525190
81900360200190f35b605a600283035b600060018211603b5750
806055565b0190505b919050565b605160018403604256
```

Figure 4.3: Compiling code in Algorithm 2 to EVM bytecode by using Solidity 0.4.1 compiler

```
0x60606040526000357c0100000000000000000000000000000
00000000000000000000000000090048063193ddd2c14603757
6035565b005b6042600480505060 5a565b60405180821515815
2602001915050604051809103 90f35b6000600560006000505 4
149050606b565b9056
```

Figure 4.4: Compiling code in Algorithm 3 to EVM bytecode by using Solidity 0.4.1 compiler

4.3.2 Disassembling

The second step of the proposed framework is the disassembler, which converts Ethereum bytecode to a sequence of readable low-level mnemonics [11]. To do this conversion, the bytecode is scanned, then each instruction converts to its corresponding mnemonic and incrementing a program counter for each instruction and each inline operand. The output of this step is a series of low-level operations and their input arguments.

---

[11] In programming, a mnemonic is a name assigned to a machine function or an abbreviation for an operation. Each mnemonic represents a low level machine instruction or opcode in assembly. add, mul, lea, cmp, and je are examples of mnemonics.

Table 4.1: Disassembling Figure 4.4 bytecode to opcode

```
1: 0x0 PUSH1   0x60
2: 0x2 PUSH1   0x40
3: 0x4 MSTORE
4: 0x5 PUSH1   0xe0
5: 0x7 PUSH1   0x02
6: 0x9 EXP
7: 0xA PUSH1   0x00
8: 0xC CALLDATALOAD
9: 0xD DIV
10: 0xe PUSH4   0xc6c2ea17
11: 0x13 DUP2
...
12: 0x14 EQ
13: 0x15 PUSH1  0x1c
14: 0x17 *JUMPI
15: 0x18 JUMPDEST
16: 0x19 PUSH1  0x02
17: 0x1B *JUMP
18: 0x1C JUMPDEST
19: 0x1D CALLVALUE
20: 0x1E PUSH1   0x02
21: 0x20 *JUMPI
22: 0x21 PUSH1  0x29
...
```

In our examples (Algorithm 2 and Algorithm 3), after the second step, the machine code will be converted to the readable EVM opcodes by the disassembler as shown in Table 4.1 and Table 4.2.

4.3.3 Feature Engineering

Control Flow Graph Extraction

A Control Flow Graph (CFG) is a representation, using graph notation, of control flow or all paths that might be traversed through the execution of programs. Control flow graphs are mostly used in static analysis as well as compiler applications, as they can precisely represent the flow inside of a program unit.

The CFG of an EVM opcode program is initially unknown due to the stack locations, and it needs to be built incrementally and iteratively. To address this problem, we construct a CFG incrementally and we propagate the potential jump addresses' values. In opcode, all

Table 4.2: Disassembling Figure 4.3 bytecode to opcode

1: 0x0 PUSH1 0x60	12: 0x14 EQ
2: 0x2 PUSH1 0x40	13: 0x15 PUSH1 0x1a
3: 0x4 MSTORE	14: 0x17 JUMPI
4: 0x5 PUSH1 0xe0	15: 0x18 JUMPDEST
5: 0x7 PUSH1 0x02	16: 0x19 STOP
6: 0x9 EXP	17: 0x1A JUMPDEST
7: 0xA PUSH1 0x00	18: 0x1B PUSH1 0x00
8: 0xC CALLDATALOAD	19: 0x1D SLOAD
9: 0xD DIV	20: 0x1E PUSH1 0x05
10: 0xe PUSH4 0x193ddd2c	21: 0x20 EQ
11: 0x13 DUP2	22: 0x21 PUSH1 0x60
...	...

JUMP instructions must jump to a *JUMPDEST* instruction. Due to this fact, we can split the disassembled EVM bytecode into basic blocks. In this scenario, the main problem is identifying the JUMPDEST in the JUMP operation since it is not an explicit argument and it requires some efforts at runtime to pop it from stack. To overcome this issue, we need to resolve it in two phases (Brent et al., 2018).

In the first phase, we construct the basic blocks of smart contracts. Then, we build a symbolic stack with symbolic values to execute each block and pop its operations from the stack. The registers may point either to stack locations whose values were produced by the prior basic blocks or stack locations that were produced by the current basic block. (Brent et al., 2018) introduce symbolic labels for stack locations that are used to indicate data-dependencies among basic blocks, and try to resolve them via registers in the next phase.

In the second phase, the symbolic stack locations will be resolved incrementally and new JUMPDEST will also appear. As a result, the control flow graph will be completed incrementally.

After this step, the majority of JUMPDESTs or sets of potential JUMPDESTs for basic blocks have been resolved. For example, the corresponding CFG of Algorithm 2 is demonstrated in Figure 4.5. The CFG of Algorithm 3 is also presented in Figure 4.6.

Feature Extraction

We employ EVM bytechode control-flow graph to extract features. In this step, we use the depth-first search (DFS) algorithm to traverse the CFG nodes and extract the operation names in DFS paths as shown in Algorithm 4. This algorithm runs on all CFGs and its output is a vector of operation names. We will add zeros (padding) to the end of some vectors to get vectors with the same length (Parveen et al., 2011).

```
Depth First Search(control_flow_graph): {
  visited, stack, opname_fea = set(), [root], [root.opname]
  while stack: {
    vertex = stack.pop()
    if vertex not in visited: {
      visited.add(vertex)
      opname_fea.append(vertex.opname)
      stack.extend(graph[vertex] - visited)
      }
    }
  return opname_fea
  }
```

Algorithm 4: Depth First Search on CFG

4.3.4 Bidirectional Long Short-Term Memory

Neural networks have been successfully applied in many areas such as image processing (Breen et al., 2002), speech recognition, biomedical (Manoochehri et al., 2019), and natural language

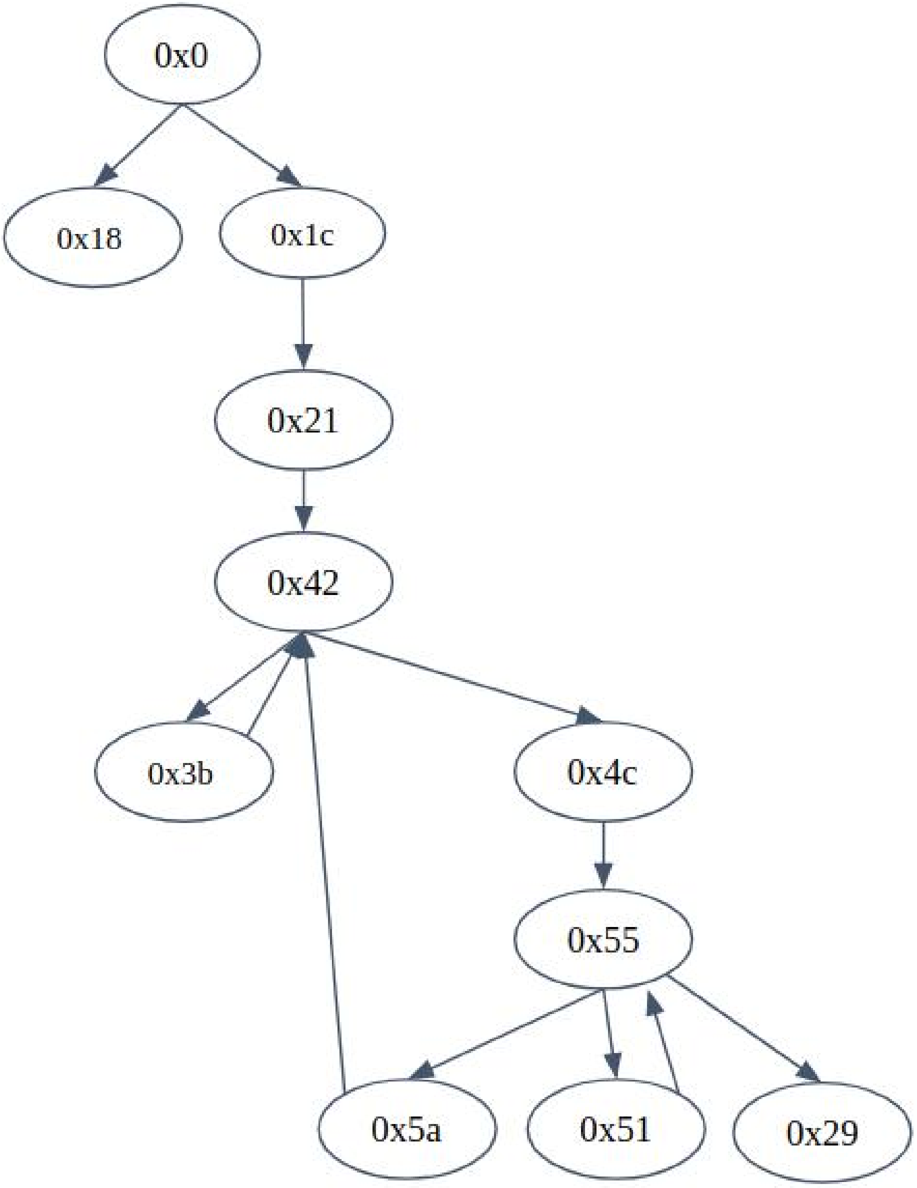


Figure 4.5: Control flow graph for the Fibonacci contract (Algorithm 2) by using (Brent et al., 2018)

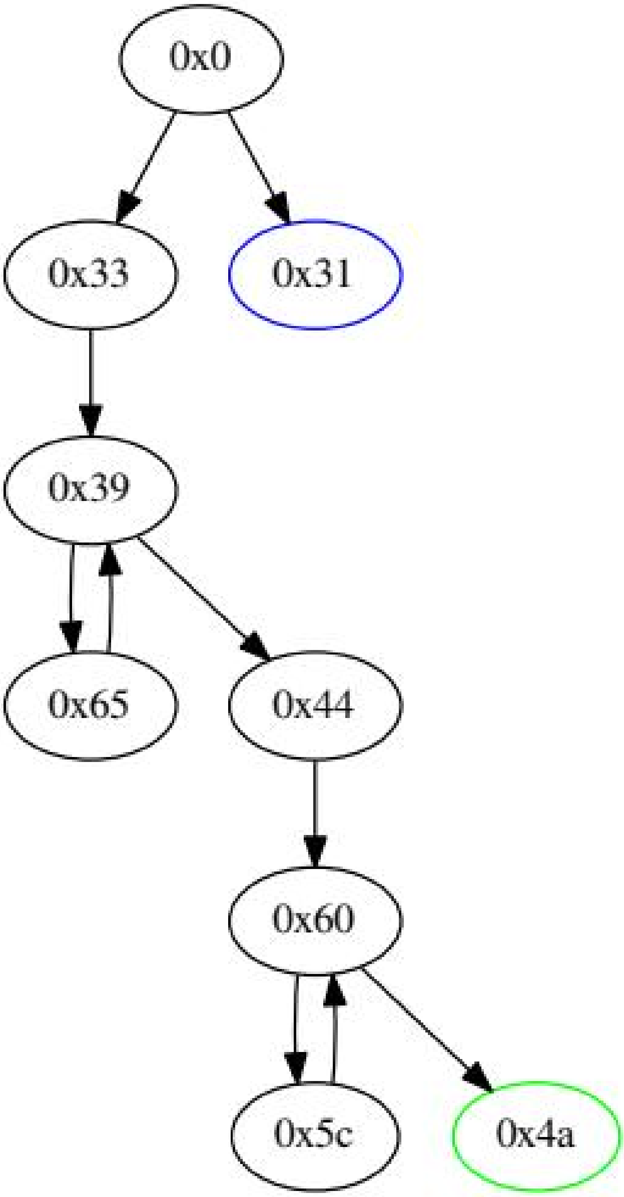


Figure 4.6: Control flow graph for the Factorial contract (Algorithm 3) by using (Brent et al., 2018)

processing. A recurrent neural network (RNN) is a type of neural network where connections between nodes form a directed graph along a temporal sequence (Mikolov et al., 2010). This type of neural network is appropriate for sequential data. However, they mainly suffer from the *Vanishing Gradient* (VG) problem, which makes them inappropriate for model training (Criminisi et al., 2012). Long short-term memory (LSTM) is another recurrent neural network (RNN) architecture which was proposed to overcome VG problem (Schmidhuber and Hochreiter, 1997). LSTM has feedback connections, unlike traditional feedforward neural networks. The fundamental idea of LSTM is to introduce an adaptive gating mechanism, which decides the degree to which LSTM units keep the previous state and memorize the extracted features of the current data input. LSTM cannot only process single data, but it can also process entire sequences of data. A lot of LSTM variants have been developed for various applications so far. Although LSTM processes sequences in temporal order, it ignores future context.

On the other hand, neural networks that can deal with contexts and sequences may be suitable for vulnerability detection in codes. In other word, neural networks for natural language processing may be also suitable for vulnerability detection because both of them deal with context as their important features (Montemagni and Pirrelli, 1998). We can notice that the argument(s) of a program is often affected by earlier operations in the program and may also be affected by later operations in the program. This means that the standard LSTM may be insufficient because it is unidirectional and we need to use Bidirectional LSTM (BiLSTM) for vulnerability detection.

One of the contributions of this study is using BiLSTM, which can automatically focus on the program operations that have decisive effect on classification, to capture the most important semantic information in a program context, without using extra knowledge.

We discuss these neural networks configurations in section 4.4.

4.4 Evaluation

In this section, we are going to describe our benchmarks and present the experimental results.

4.4.1 Benchmarks

We collected our corpus by retrieving bytecode for every contract deployed as of May 2019. Then, we removed the duplicated contracts. In total there are only 205,848 unique smart contract bytecodes. Ethereum contracts vary from simple to more complex. The number of the instructions in Ethereum contracts varies from 8 to 13,050 with average 1,545.27 and median 1,213. The number of distinct instructions in the contracts varies from 7 to 57. In this project, we use EVM bytecode instead of smart contract source codes since most of the smart contracts' source code is not publicly available on public repositories.

Due to the lack of labeled data, we run Oyente on all contracts to label them by different vulnerabilities including ordering, timestamp, mishandle exception, and reentrancy or benign. Then, we discard the contracts that Oyente could not complete the process in 60 seconds. At the end of this process, there are 176,968 unique bytecodes. This data is our gold standard for our further process. Figure 4.7 shows the percentage of benign and vulnerabilities in smart contracts detected by Oyente. 88% of smart contracts are benign. 6% of smart contracts suffer from Ordering attacks. Besides, some of the smart contracts suffer from more than one vulnerability. It is shown in Figure 4.8.

In this project, we consider all types of vulnerabilities as a non-benign class. So, we convert the problem to the binary classification problem.

4.4.2 Neural network configuration

We compared our model with the LSTM neural network. Its model and implementation parameters are depicted in Figure 4.9 and 4.10, respectively.

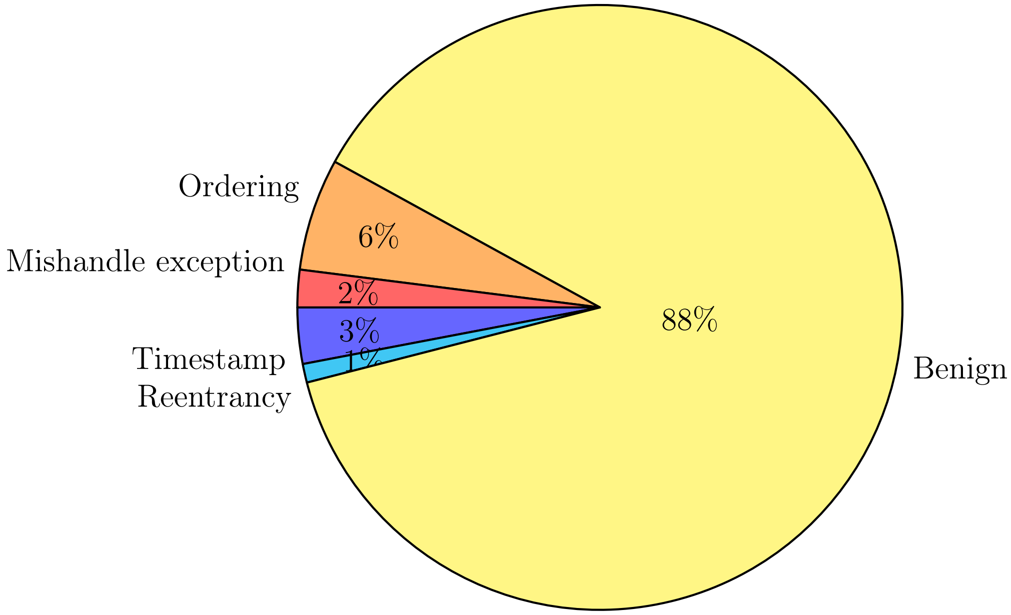


Figure 4.7: Percentage of benign and vulnerabilities in the benchmark

Figure 4.9 also highlights the structure of BiLSTM. The input to the learning process is in a vector representation which we extract from CFG of programs.

BiLSTM model has an input layer, a BiLSTM layer, a couple of flatten layers, a dense layer, and lastly an activation layer. The BiLSTM layer has both forward and backward directions. The BiLSTM layer contains some complex LSTM cells, which are treated as black-boxes in the present study. The dense layer reduces the number of dimensions of the vectors received from the BiLSTM layers. The activation layer takes the low-dimension vectors received from the dense layer which are flattened in the flatten layer as input, and is responsible for representing and formatting the classification result, which provides feedback for updating the neural network parameters in the learning phase. The output of the learning phase is a BiLSTM neural network with fine tuned model parameters, and the output is the classification results.

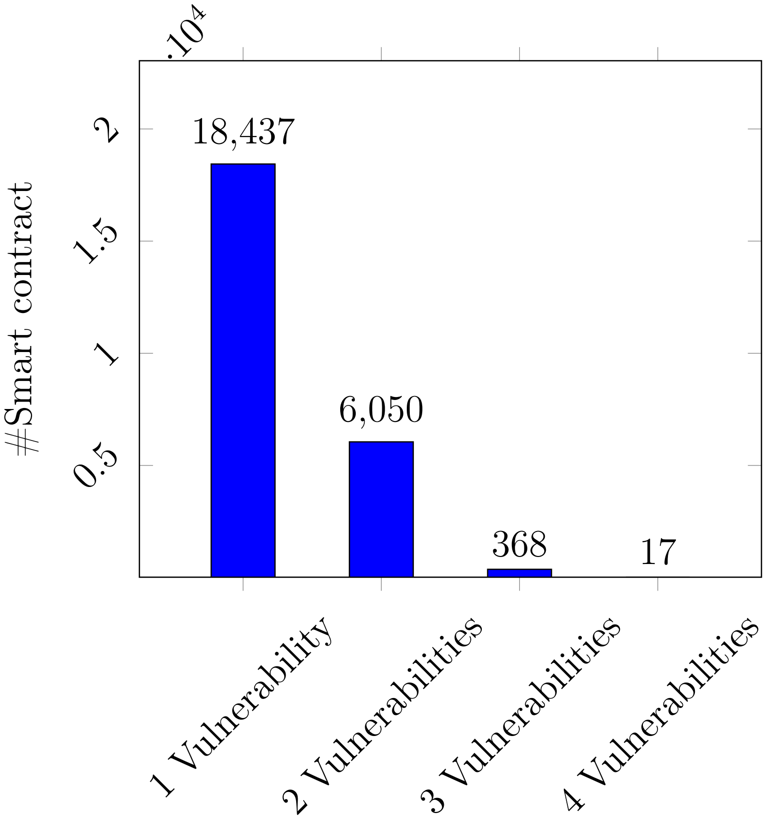


Figure 4.8: Number of smart contracts in Benchmark with one or more vulnerability

4.4.3 Experimental Results

The initial benchmark construction and feature extraction process were conducted on an Intel machine having Core-i7 3.40GHz CPU with 64 GB of RAM, running a standard Ubuntu Linux version 16.04 LTS.

We compare the performance of the proposed method with different learning models in Table 4.3. We use KNN (K=3), Random Forest (RF), SVM, LSTM, and BiLSTM in this regards. RF works better among the traditional learning models. We show SVM result with Linear kernel. However, the results of Linear and Radial Basis Function (RBF) kernels were very similar.

Our proposed method with using BiLST outperforms the other methods with 87% accuracy, 80% precision and 87% recall.

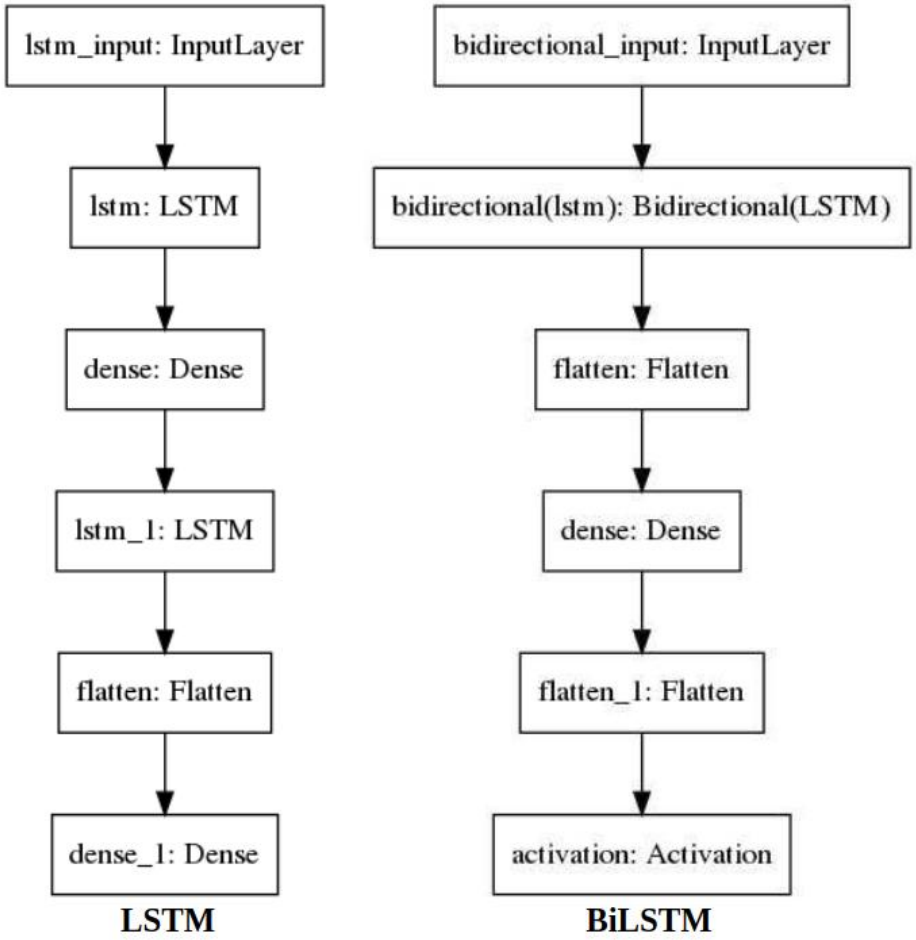
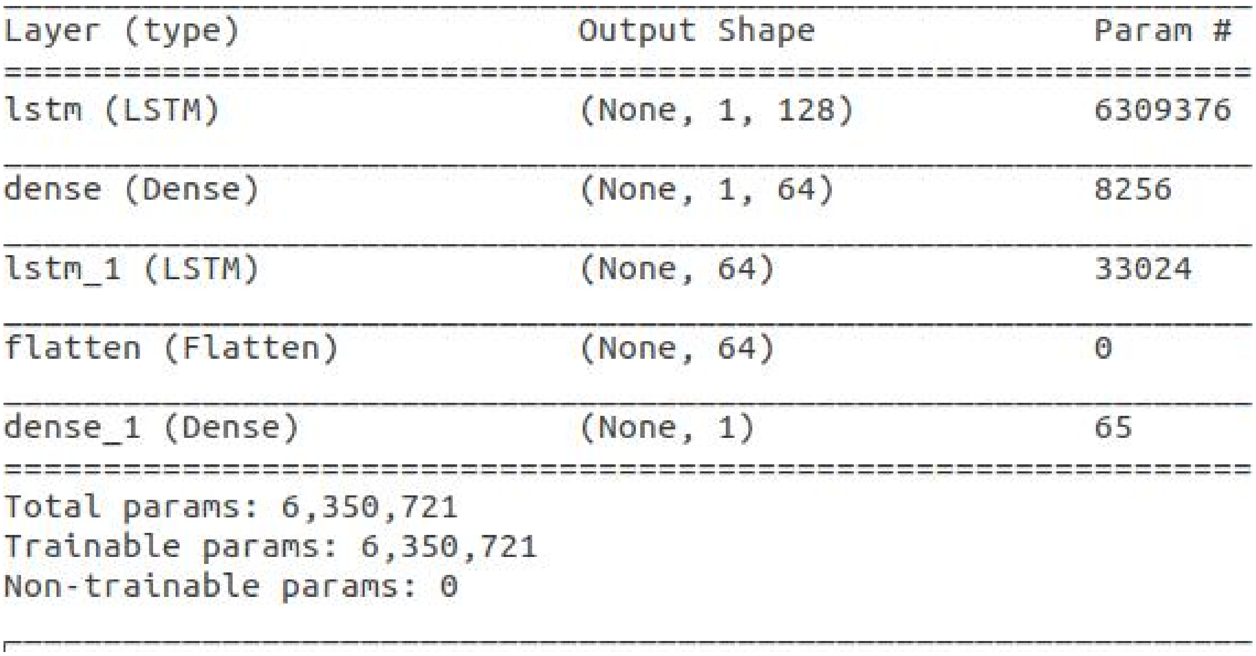


Figure 4.9: LSTM and BiLSTM models

Table 4.3: Comparison of the proposed method with KNN, Random Forest, and SVM

Model	**Accuracy**	**Precision**	**Recall**	**F1 score**
KNN (k=3)	0.59	0.65	0.60	0.62
Random Forest	0.76	0.76	0.77	0.77
SVM	0.72	0.73	0.73	0.73
LSTM	0.85	0.78	0.85	0.81
BiLSTM	0.87	0.80	0.87	0.83



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 1, 128)            6309376
_________________________________________________________________
dense (Dense)                (None, 1, 64)             8256
_________________________________________________________________
lstm_1 (LSTM)                (None, 64)                33024
_________________________________________________________________
flatten (Flatten)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 6,350,721
Trainable params: 6,350,721
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.10: Implementation of LSTM model for smart contract vulnerability detection in Keras

CHAPTER 5

A SCALABLE FRAMEWORK FOR WIND TURBINE FAULT DIAGNOSIS[1]

5.1 Introduction

The wind energy industry has experienced major growth over the last decade. Based on Global Wind Energy Council statistics, the total number of wind turbines across the world has increased from 6,100 in 1996 to 3,695,789 in 2014 (Pullen and Sawyer, 2014), showing an exponential growth in global demands for wind energy to produce electricity.

Wind energy is freely available, does not pollute the environment, and needs less space for energy production.

Wind turbines are big, complex and expensive machines which are installed in harsh environments. They are vulnerable to different defects and faults. Manual condition monitoring is cumbersome and costly. A failure in these machines can disable them for several weeks and cost hundreds of thousands of dollars. These costs, typically due to maintenance and repair of failed components, adversely affect the price of produced energy. Developing automatic condition monitoring systems for these machines is vital for the wind industry. Based on a report by the National Renewable Energy Lab (NREL), gearbox failure causes the longest downtime, about half a month on average, and is the most costly fault in these machines (Sheng and Veers, 2011). Different factors like poor lubrication, bending fatigue, fretting corrosion, and mechanical stress can cause a defect in a wind turbine gearbox. An early fault detection can prevent catastrophic failures. Monitoring vibration signals picked up from gearboxes is an effective way of condition monitoring. However, by increasing the number of

---

[1] This chapter contains material previously published as:
©2017 Springer. Reprinted, with permission, from Imani M. B., Heydarzadeh M., Chandra S., Khan L., Nourani M. (2019) SAIL: A scalable wind turbine fault diagnosis platform. in: Bouabana-Tebibel T., Bouzar-Benlabiod L., Rubin S. (eds) Theory and application of reuse, integration, and data science. IEEE IRI 2017. Advances in intelligent systems and computing, vol 838. Springer, Cham.
Lead author, Imani, conducted the majority of the research, including most of the writing, design, implementation, and evaluation.

turbines, the number of sensors to monitor will dramatically increase. This typically leads to a large volume of data that needs to be processed (Masud et al., 2008). In fact, computation of these data cannot be achieved on site or locally. To address this challenge, we propose a cloud-based solution for the following reasons. First, wind turbines are mostly installed in harsh environments including off shores and deserts. Providing a local computation platform for running a fault diagnosis algorithm requires a high level industrial specifications. Such a solution is expensive, specifically in typical cases where the fault diagnosis algorithm needs a powerful computation platform. Second, satellite links can provide network coverage in such remote areas; however, their services are still expensive, even with low bandwidth. Besides, wind turbines do not need continuous monitoring; regular maintenance with periods of a day or more is a generally an accepted practice. Finally, a local processing solution at every turbine is not appropriate to scale when monitoring wind farms with thousands of turbines. Therefore, monitoring each wind turbine with a standalone computer is not cost-effective.

Even though increasing the volume of data provides more opportunities for accurate data analysis , it produces considerable computational challenges. Recently, the volume of data has been growing so fast in different areas that it can no longer fit into memory for efficient processing. To handle a huge amount of data, engineers have developed different scalable tools. New processing technologies such as Google's MapReduce and Apache Hadoop have emerged as a result of such efforts. With the exponential increase and availability of tremendous amounts of data, *Big Data* is becoming one of the most popular terms in today's automated world. From an industry point of view, Big Data is going to play an important role in achieving fault-free and cost-efficient data collection and analysis in real-time systems. Automated fault detection of wind turbines can be viewed as a Big Data problem. Sensor data from wind turbines can result in a continuous stream of data, useful for diagnosis. Unfortunately, traditional approaches (Kusiak and Li, 2011) to address challenges of fault detection are inefficient in this setting. They typically require a high amount of memory and several hours to train an appropriate model and they cannot take advantage of a live stream of data. Using such traditional techniques may result in maintenance delays with increased costs.

From a signal processing perspective, there are still open research problems. Most of prior data-driven fault diagnosis algorithms are based on frequency domain analysis (Haddad et al., 2015; Immovilli et al., 2009; Riera-Guasp et al., 2012). This is due to semi-periodic nature of vibration signal. The majority of prior works use the Fast Fourier Transform (FFT) to estimate the spectrum of signals which lead to a fast and an unbiased estimation of the spectrum. However, using FFT in feature extraction for an automatic fault diagnosis system raises a few challenges. First, FFT is computationally expensive and does not reduce the dimensionality of data. Moreover, the frequency resolution of FFT is limited to the number of the point used in its calculations. As a result, if the frequency signature of vibration for different classes is similar, FFT cannot capture them.

5.2 Background

5.2.1 Industrial Applications

The two categories of fault diagnosis methods are model-based and data-driven. Model-based approaches use a mathematical model of the system (Chen and Patton, 2012). Such a model is usually obtained by physical modeling. Having a model reduces the uncertainty regarding measurements, however, physical modeling is challenging. On the other hand, the only assumption that data-driven methods make is the availability of some training data from the system. Then, signal processing and machine learning are applied to predict the system's state of health (Watson et al., 2010). Such approaches are easier to generalize compared to other approaches since they do not make any assumptions about the system.

The majority of prior gearbox fault diagnosis works are data-driven methods. However, since vibration signal is a non-stationary random signal, extracting a compact informative feature vector is a challenging signal processing task. The key difference among prior works is in the feature extraction step. Authors in (Zhang et al., 2012; Li et al., 2016; Raj and Murali, 2013) proposed time-domain statistical features. Specifically, they showed that higher order statistics like kurtosis and skewness can be employed for fault diagnosis of the gearbox (Lei et al., 2007). However, these features are very sensitive to noise and outliers. This limits their application to real-world industrial systems. The frequency domain features are proposed in (Watson et al., 2010; Hu et al., 2007; Nelwamondo et al., 2006; Mahamad and Hiyama, 2011). Frequency domain methods are robust and can distinguish different classes of faults. However, for developing an automatic fault diagnosis system, it is necessary to reduce the dimensionality of signal representation. Fourier transform and wavelet transform do not necessarily reduce the dimension of a signal's spectrum. Authors in (Sun et al., 2013; Amar et al., 2015) post-processed the spectrum of the signal by Principal Component Analysis (PCA) for dimensionality reduction. However, PCA requires large computations at run-time and can corrupt the spectrum of the signal. So, presenting the spectrum of vibration signal in a compact way is required for developing robust fault diagnosis system. In a preliminary study, we developed a parametric spectral analysis for feature extraction using an auto-regressive (AR) model (Imani et al., 2017). Although, using an AR model provides a method for compact representation, it requires a high model order for capturing complex signals. This issue can be addressed using a parametric model which has zeros. In this work, we used an *auto-regressive moving average (ARMA)* model which has zeros and poles and can model vibration signal with smaller orders. Moreover, using an ARMA model provides a high resolution in estimating spectrum of signal. This will be discussed in more details later (Abrol and Khan, 2010).

The majority of research in this field is devoted to developing signal processing algorithms for fault diagnosis. However, this problem has another challenge for data analytics. Scaling a real-time fault diagnosis algorithm for a wind farm with thousands of turbines requires a massive computational power which is currently available on cloud servers. This dimension of research is related to the emerging field of *Industrial Internet of Things* (IIoT). From this perspective, authors in (Chen et al., 2007; Qi et al., 2016; Shao et al., 2016) applied the state-of-the-art Big Data analytics methods to the fault diagnosis problem in industrial systems in different areas.

5.2.2 Big Data Analytics

Many different Big Data analysis tools with different features have been developed so far, such as Hadoop (Shvachko et al., 2010) and Apache Spark (Spark, Spark). In our work, we have selected Spark over other traditional distributed frameworks (e.g., Hadoop and MapReduce, etc.) since it is more efficient for stream data processing. Apache Spark is a fast and general purpose cluster computing engine for large-scale data analytics that was developed at the University of California, Berkeley AMP Lab (Spark, Spark). Many different applications have been developed based on Spark clustering so far, including political event coding (Solaimani et al., 2016), and geolocation extraction (Imani et al., 2017).

Another feature that Spark offers is supporting scalable, high-throughput, fault-tolerant, and real-time data stream processing. Spark streaming uses a “discretized stream" which is an incremental stream processing model. Data can be fed by many sources like Apache Kafka (Kafka, 2014).

Machine Learning library (MLlib) is Spark's distributed and scalable machine learning library (Meng et al., 2016). It improves the computational efficiency by using data-parallelism or model-parallelism techniques to store and process data or models. MLlib includes more than 50 common algorithms for classification, clustering, regression, collaborative filtering, and dimensionality reduction.

5.3 SAIL Architecture

5.3.1 Scalable Monitoring System

In the wind turbine industry, different sensors are being used to detect and predict various faults and malfunctions. As the number of turbine increases, the number of sensors, and consequently the amount of gathered data, grow as well. The traditional methods for data processing are time-consuming and inefficient for a huge amount of stream data. Therefore, Big Data analytics can play an important role in processing and mining these rapidly producing data in real-time.

5.3.2 Bottom-Up Architecture

We propose a bottom-up three-layer platform depicted in Figure 5.1. At the bottom, the sensor-layer contains data collection nodes which are used to collect data from wind turbines. A node is attached to each turbine and provides sensory circuits and network connectivity. The next layer is the fog-layer which contains fog-servers. These servers collect data from all nodes in a wind farm. Fog-servers compress data to save the communication bandwidth. Then, they send the collected data to a cloud server via the Internet. In the cloud-layer, the analytic server runs the proposed fault diagnosis algorithm. After that, the results are logged and reported to different users. This architecture complies with the paradigm in the emerging field of Industrial Internet of Things (IIoT).

5.4 Fault Diagnosis Framework

In this section, the proposed scalable architecture is discussed. Figure 5.2 shows the multi-layer structure of SAIL. This figure shows where each piece of algorithm runs. Algorithm training runs offline. These are depicted as gray boxes. Other modules work and process data online. Development of a mechanic fault can take minutes or even hours so the whole

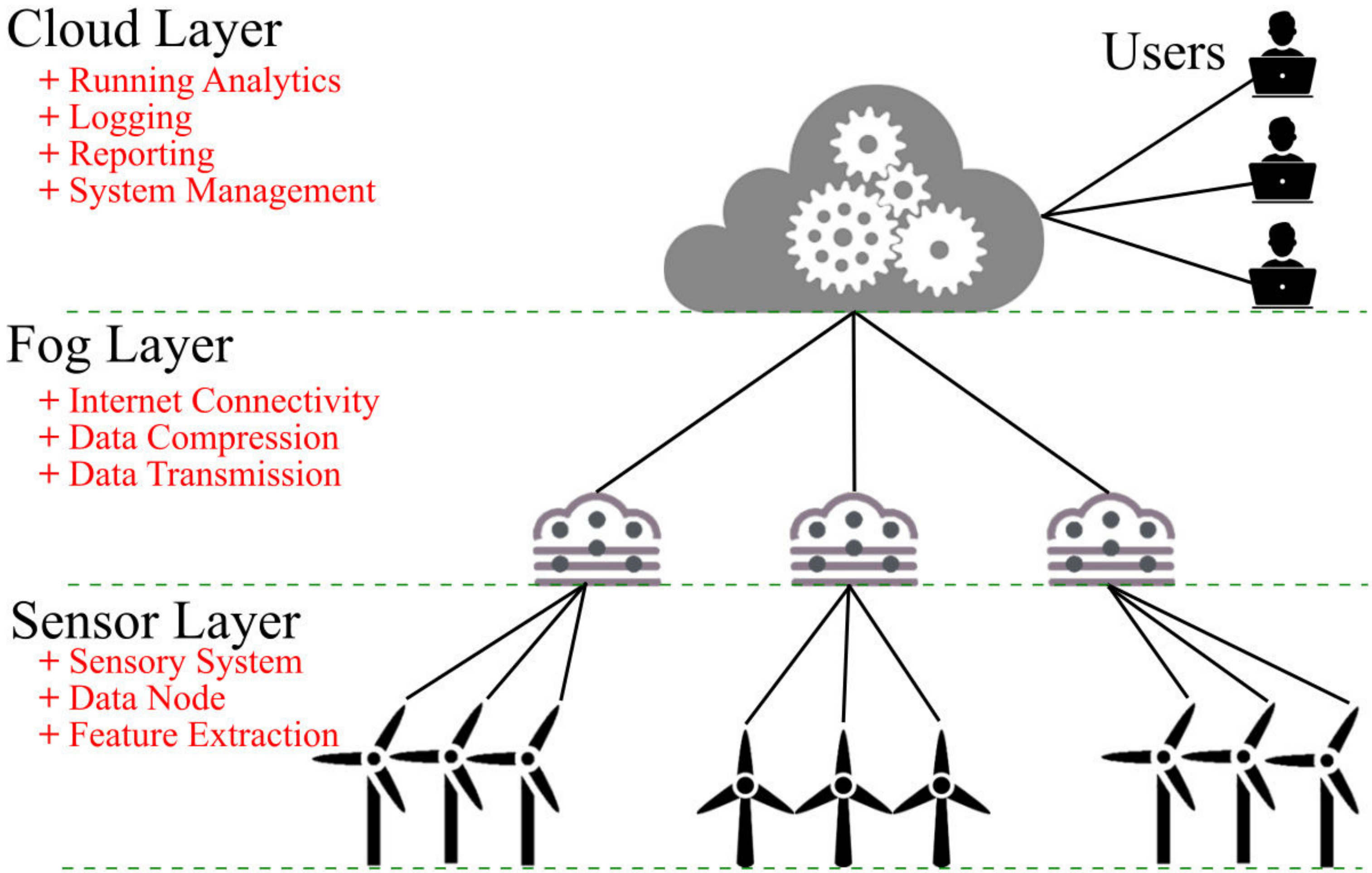


Figure 5.1: The proposed three-layer platform (SAIL).

system can generate early alerts in an online manner. There are three layers in the proposed system: sensor-layer, fog-layer, and cloud-layer. Furthermore, each layer contains several components which are discussed in this section.

5.4.1 Sensor Layer

In this layer, each turbine in a farm has a node which provides sensory circuits and local network connectivity. In each node, vibration data is collected using accelerometers. After extracting the features, it broadcasts the data stream to the fog layer.

Feature Extraction

We applied the model-based spectral analysis method explained in detail in Appendix A for feature extraction. We considered an ARMA model for the vibration signal and use the parameters of the model as features for diagnosing the health state of the system. These features capture the spectrum of the signal and simultaneously reduce its dimensionality. The proposed system uses a p order on-line ARMA estimator to a sliding window of vibration signal from the device to extract coefficients. Then, our features are the vector of all model coefficients $\boldsymbol{\theta} = [a_1, a_2, \cdots, a_p]^t$. Vector $\boldsymbol{\theta}$ is fed to a trained random forest classifier to diagnose the status of the device.

Here, the order of online estimators is unknown and needs to be estimated. As discussed in Subsection A.3, AIC criteria can be used for order selection. However, we do not deal with just one random process because each class of fault produces a different random process as vibration signal. Consequently, each class has its own order. The order of the online estimator should be chosen such that the fitted model captures the spectral properties of the vibration signal for each fault. So, we set this parameter as the largest estimated p in training data for all classes of vibration data. The pseudo code illustrated in Algorithm 5 shows the algorithm used for order selection which finds the maximum order for each window of vibration in training data for all classes of faults.

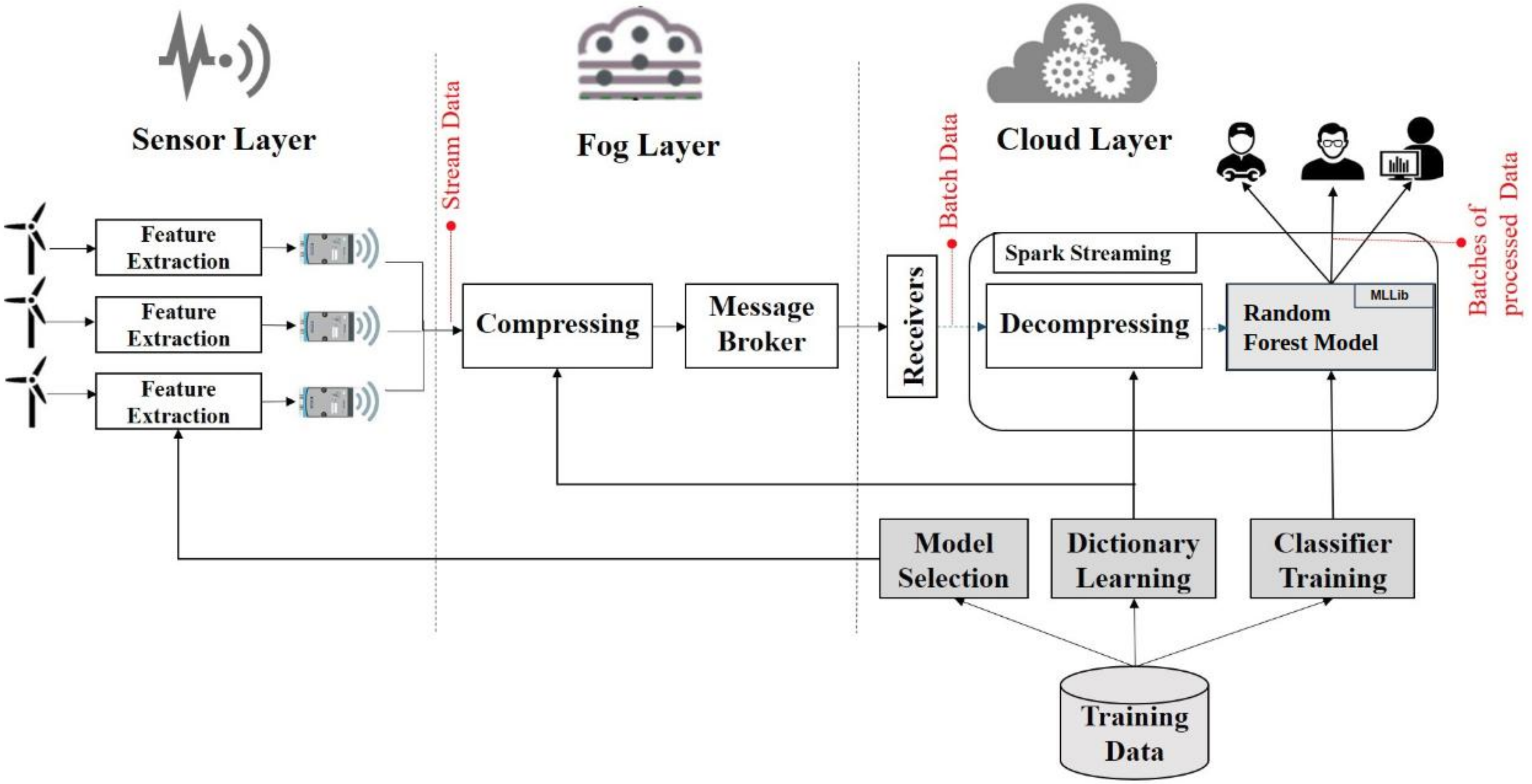


Figure 5.2: The proposed fault diagnosing framework

```
Input: Training data $\mathbb{D}$, window length $L$
Output: Order of on-line estimator $p$
Initialize: remove noise by low-pass filtering of $\mathbb{D}$
foreach class, select the data in that class ($\mathbb{D}_i$) do
    for $k^{th}$ window of $\mathbb{D}_i$ do
        for $j \leftarrow 1$ to $p_{max}$ do
            Fit a $j^{th}$ order ARMA model
            $F_k(j) \leftarrow$ AIC of the fitted model (Eq. A.6 )
        end
        $p_i(k) \leftarrow \underset{j}{argmin}\{F_k(j)\}$
    end
    $p(i) \leftarrow \underset{k}{max}\{p_i(k)\}$
end
$p \leftarrow \underset{i}{max}\{p(i)\}$
Return $p$ as the order of on-line estimator
```

Algorithm 5: On-line PSD Estimator Order Selection

5.4.2 Fog Layer

This layer compresses the collected data from the wind turbine farm to save communication bandwidth. As we discussed dictionary learning and compression in subsection A.4, the dictionary learning is offline since it uses the training data. However, the data compression is online for signals coming from wind turbines. Then, this layer sends the collected compressed-data to the next layer, i.e. cloud layer, using Kafka as a message broker.

Message Broker Module

Various message brokers are available to integrate with Spark. We have applied Kafka 3.3.4 because of its stability and compatibility with Apache Spark. Kafka is a real-time publish-subscribe messaging system developed by LinkedIn (Kafka, 2014). Kafka publisher-clients write messages in topic categories, and each topic category is divided into several partitions, and messages within a partition are totally ordered. Kafka subscriber clients read messages on a topic. It provides reliable message delivery with the correct order. Our framework continuously monitors each incoming datum from different sources. We read these streaming data and transport it through a message broker. It enables us to feed these large volumes of raw data to the Spark streaming module (Figure 5.2).

5.4.3 Cloud Layer

Initially, we perform model and parameter selection for feature extraction (subsection 5.4.1) and train a diagnosis model offline by using raw training data. Moreover, dictionary learning for compressing and decompressing is done in offline mode. Then, the prepared model will be used in the online fault diagnosing framework, which is presented in Figure 5.2.

In this layer, the Spark Streaming module receives a stream of data and converts them to batches of data. After decompressing the data, we use the classifier to diagnose whether there is faulty data or not (subsection: 5.4.3).

To evaluate the model's performance, we show the experimental results with offline data in Section 5.5. In this section, we will introduce our feature extraction method and machine learning algorithm which are used in this study.

Random Forest Classifier

Decision trees are simple classifiers which make an output label by comparing different features with threshold values in a tree structure. Although trees are unbiased classifiers, their variance is high. One way of decreasing their variance and thus increasing the classification accuracy is by *bagging*, which combines the output of several trees (Hastie, 2009). An ensemble of several trees is usually referred to as a *forest*. In a forest, each tree votes for the label of an input and the final decision is made using the majority voting technique. Although each tree is a simple classifier, the forest can learn difficult classification tasks by bagging trees together. One advantage of a forest is its suitable structure for real-time implementation since each can be evaluated in parallel and the final decision can be made by combining all results together. *Random Forest* (RF) is an effective way of training bagged trees where tree parameters are chosen randomly (Hastie, 2009). In a random forest, each tree is trained using a random subsample of training data. Algorithm 6 shows the random forest algorithm. In this study, we used RF in MLlib which supports the Random Forest's parallelization.

```
Input: Training data ($\mathbb{D}$), number of trees ($B$), Minimum node
       size ($N_m$)
Output: A random forest classifier
for $i \leftarrow 1$ to $B$ do
    $\mathbb{Z} \leftarrow$ a random subset of $\mathbb{D}$
     Grow a random tree ($T_b$) using $\mathbb{Z}$ as:
     for $j \leftarrow 1$ to $N_m$ do
         Select $m$ variable at random from $p$ variable
          Pick the best variable/ split-point among the m
          Split the node into two daughter

    end
end
Return The ensemble of trees $\{T_b\}_1^B$
```

Algorithm 6: Random Forest Classifier.

5.5 Experimental Results

We present two categories of experiments to evaluate SAIL platform and the proposed fault diagnosing algorithm in this section. First, we measure the accuracy of the proposed fault diagnosing method by applying it to two well-known vibration benchmark datasets, i.e. NREL and CWRU datasets. Using benchmark datasets provides an opportunity to compare our algorithm with prior works regardless of our scalable implementation. Next, we evaluate the scalability and run-time of the proposed platform. In this experiment, we created multiple sensor node instances which feed duplicates of these datasets to the SAIL.

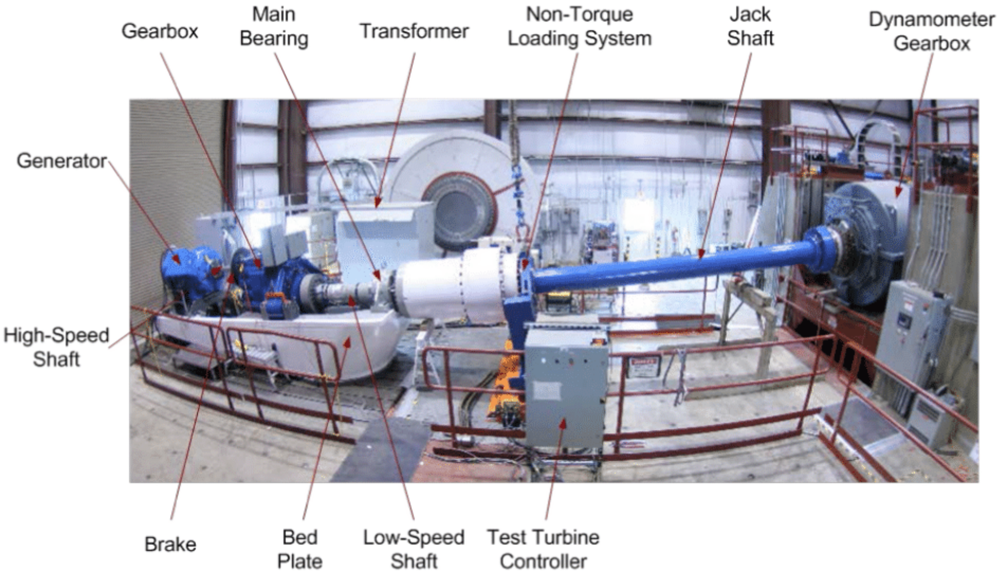


Figure 5.3: The test turbine (image from ((NREL), 2017))

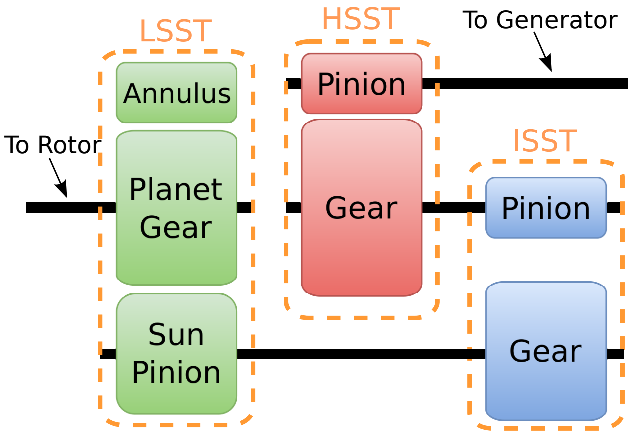


Figure 5.4: The gearbox block diagram

5.5.1 Algorithm Accuracy Test

NREL Dataset

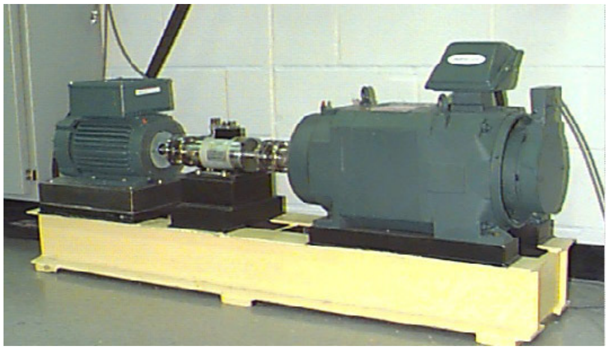
The vibration data used in this research is provided by the National Renewable Energy Lab (NREL) ((NREL), 2017). The test turbine is a 750 kW three-bladed upwind turbine with stall control. Figure 5.3 shows the turbine.

For testing the proposed algorithm and platform, we used the wind turbine gearbox data provided by NREL. This dataset is collected from a damaged gearbox. The complete bed plate and drive-train of a turbine were installed at dynamo-meter test facility (DTF) of NREL ((NREL), 2017). The bed plate is fixed to the floor without the rotor, yaw bearing, or hub. The gearbox is shown in Figure 5.3 and has three-stages: low-speed stage (LSST), intermediate-speed stage (ISST) and high-speed stage (HSST), as shown in Figure 5.4. LSST is connected to the rotor and HSST is connected to the generator. In a wind turbine, the LSST shaft is connected to the rotor and the HSST shaft is connected to the generator.

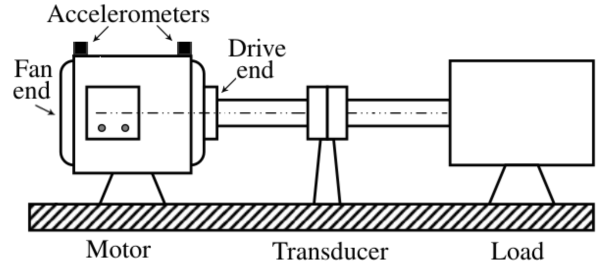
For measuring the vibration data, a set of accelerometer sensors is mounted to sides of the gearbox housing. On the side, the speed of HSST shaft is recorded. In this study, we only use the vibration data. The vibration data of each sensor is sampled with a PXI - 4472B data acquisition board from National Instruments at the rate of 40 KHz per channel. In this study, $V_i[n]$ for $i = 1, \cdots, 8$ denotes the samples of the i^{th} accelerometer for eight available sensors. The dataset is publicly accessible on request through ((NREL), 2017).

CWRU Dataset

We apply SAIL to the vibration data provided by the Case Western Reserve University (CWRU) bearing data center (Loparo, Loparo) as a benchmark. CWRU dataset provides a number of faults in bearing of gearboxes. Using these benchmarks provides a fair and standard framework for comparison of our algorithm with different algorithms. The experimental setup, depicted in Figure 5.5a, consists of a 2 HP motor, a torque transducer in the center, and a dynamometer as a load. Bearings of motor shaft are studied in this benchmark. Figure 5.5b shows the setup's block diagram.



(a) Setup (image from (Loparo, Loparo))



(b) Block diagram

Figure 5.5: Experimental setup of CWRU bearing data (Loparo, Loparo).

Single point faults were introduced to test bearings using an electro-discharge machine with diameters of $7, 14$ and 21 mils on different parts of bearing including the inner race, the outer race, and the ball. Vibration data was collected using two single axial accelerometers with magnetic bases. One accelerometer was mounted on the drive-end and the other on the fan-end of the motor's housing. Both were sampled at the rate of 12,000 samples per second in the constant shaft speed.

The experiment was repeated for different fault intensities (7, 14 and 21 mils) and locations (ball, races and drive/fan ends). Moreover, data were collected for four constant load conditions (0, 1, 2 and 3 HP) for each class of faults. This dataset has four classes of vibration data: fault-free, inner-race (IR) fault, outer-race (OR) fault and ball fault.

The overall statistics of these datasets, i.e. NREL and CWRU are presented in Table 5.1.

Table 5.1: Datasets Statistics

Data Status	**# Instances**
Normal	93501
Fault	103890

(a) NREL

Data Status	**# Instances**
Normal	8844
Ball Fault	15177
Inner Race Fault	15179
Outer Race Fault	31019

(b) CWRU

Accuracy and Comparison

We apply a 10-fold cross validation to report experimental results. The accuracy of SAIL is compared with prior work on NREL and CWRU dataset as reported in the literature in Table 5.2 and 5.3, respectively. These tables clearly demonstrate that SAIL outperforms the existing methods in performance. The proposed method can capture a small difference of vibration spectrum with high accuracy. Moreover, SAIL processes the vibration data in shorter windows. We propose a shorter window length and a compact feature vector which results in a simpler classifier structure and increasing confidence in its performance. In fact, a shorter window length means a lower required memory and computation power. These factors form a bottleneck on industrial-class embedded computers, which will be used for the computation platform at the node level.

Table 5.2: Comparison with prior works Using NREL benchmark

Ref.	Window Size	# of Features	Feature Extraction	Classifier	Accuracy(%)
(Zhang et al., 2012)	40,000	N/A	Time and Frequency clustering	Neural Network	98.46
(Zappalá et al., 2014)	8,000	375	Side band power factor	SVM	90
(Imani et al., 2017)	512	80	Reflection Coefficients of AR model	RF	98.93
SAIL	512	48	**Model-based Spectral Analysis**	RF	**99.53**

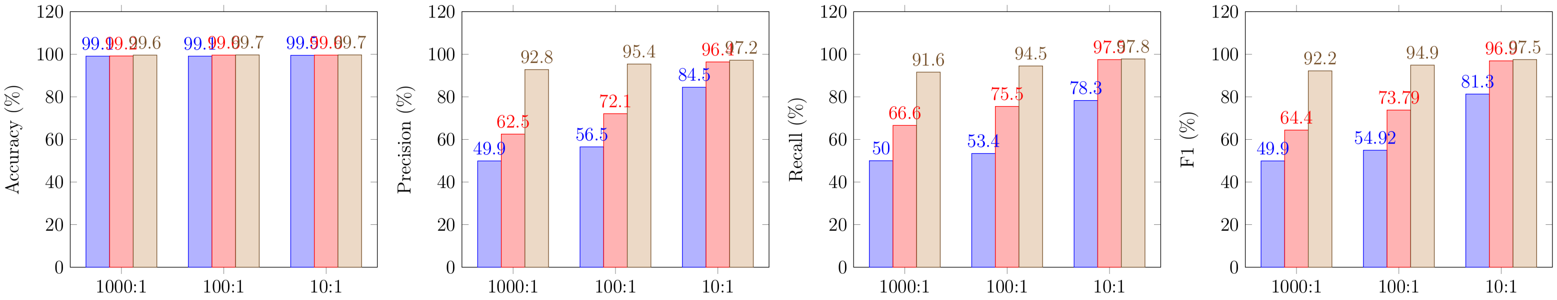
Table 5.3: Comparing various Fault Diagnosis methods using CWRU benchmark

Ref.	**Window Size**	**# of Features**	**Type of Features**	**Classifier**	**Accuracy (%)**
(Harmouche et al., 2015)	12,000	45	Global spectrum	Linear classifier	95
(Lei et al., 2007)	4,096	24	statistical& frequency domain	Multiple ANFIS	91.33
(Amar et al., 2015)	4104	4104	vibration Spectrum Imaging of FFT	ANN	96.90
(Raj and Murali, 2013)	N/A	80	two morphological operators on both FFT and Wavelet	Fuzzy classifier	N/A
(Hu et al., 2007)	4,096	11	selected from 98 statistical & Wavelet packet features	Ensemble SVM	90
(Xu et al., 2009)	N/A	10	selected from a pool of 57 features in different domains	Fuzzy ARTMAP	89.63
(Li et al., 2016)	35,750	20	nearest-farthest distance preserving projection of statistical features	N/A	99.75
SAIL	**1024**	**21**	**Model-based Spectral Analysis**	**RF**	**99.15**

Data Imbalance Problem

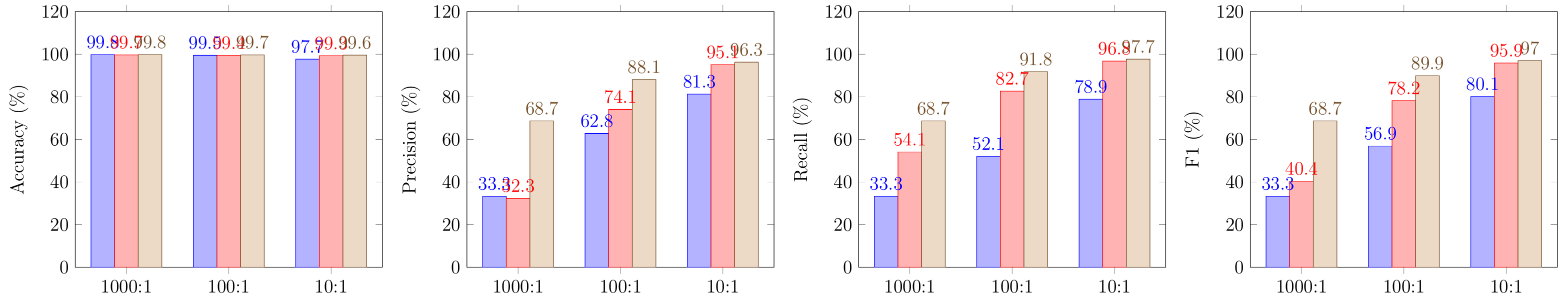
These datasets contain almost the same number of fault instances as normal instances. However, in the real world, the turbine's fault and normal data are not balanced due to the presence of few fault cases compared to a large number of normal instances. In other words, in wind turbines, the fault instances are always rare compared with occurrences of normal instances when the turbine is fully operational. As a result, the dataset becomes highly imbalanced for directly applying classification techniques. In such cases, standard classifiers tend to be overwhelmed by the large classes and ignore the small ones. For a typical fault (e.g., gearbox faults, rotor imbalance, blade angle asymmetry) the ratio of normal to fault instances can be as high as 1000:1 (Verma and Kusiak, 2011). Different solutions such as oversampling (O) and undersampling (U) approaches have been applied to the class-imbalance (Imb) problem. In oversampling, the instances of the minority class are randomly duplicated until a fully-balanced dataset is realized. In undersampling, instances of the majority class are randomly discarded from the dataset until a full balance is reached (Estabrooks et al., 2004; Wang et al., 2004).

Figures 5.6 and 5.7 show the experimental results of SAIL for different imbalance ratios on NREL and CWRU dataset, respectively. In $SAIL_{Imb+RF}$, without using the imbalance solution technique, the RF classifier overfits with the normal class and ignores the fault class(es). As a result, $SAIL_{Imb+RF}$ does not perform well. Therefore, we address this problem using $SAIL_{O+RF}$. This works better than $SAIL_{U+RF}$ since, in the latter technique, the RF classifier could not learn a good discriminative model with fewer instances of the normal class.



(a) Accuracy (b) Precision (c) Recall (d) F1

Figure 5.6: Performance of $SAIL_{Imb+RF}$ ▮▮ with different ratios of normal to fault data instances, compared to $SAIL_{U+RF}$ ▮▮ and $SAIL_{O+RF}$ ▮▮ on NREL dataset



(a) Accuracy (b) Precision (c) Recall (d) F1

Figure 5.7: Performance of $SAIL_{Imb+RF}$ 🟦 with different ratios of normal to fault data instances, compared to $SAIL_{U+RF}$ 🟥 and $SAIL_{O+RF}$ 🟫 on CWRU dataset

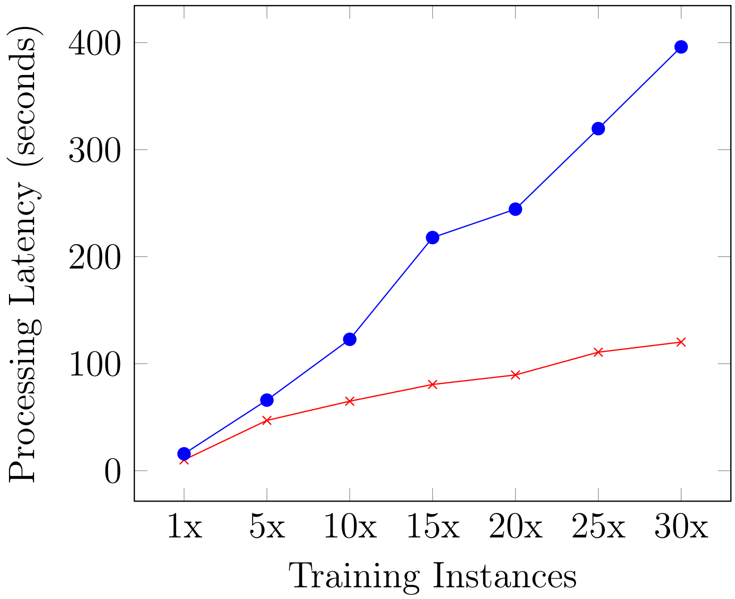
5.5.2 Platform Scalability Test

Cluster Setup

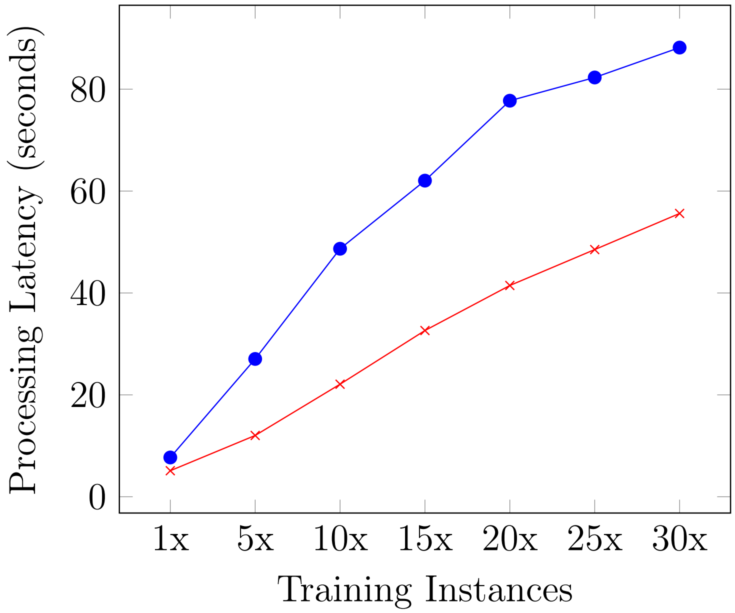
Our VMware cluster consists of 5 VMware ESXi [Palo Alto, CA] (VMware hypervisor server) 5.5 systems. Each of the systems has Intel(R) Xeon(R) CPU E5-2695 v2 2.40GHz processor, 64 GB DDR3 RAM, 4 TB hard disk, and dual NIC card. Each processor has 2 sockets and every socket has 12 cores. So there are 24 logical processors in total. All of the ESXi systems contain 3 virtual machines. Each of the virtual machines is configured with 8 vCPU, 16 GB DDR3 RAM and 1 TB Hard disk. As all the VM's are sharing the resources, performance may vary in runtime. We have installed Linux Centos v6.5 64 bit OS in each of the VM along with the JDK/JRE v1.8. We have installed Apache Spark version 2.1.0. We have also installed Apache Hadoop NextGen MapReduce (YARN) with version Hadoop v2.6.0 and formed a cluster.

Runtime Performance

We have presented the computational time efficiency of our offline and real-time framework (Figure 5.2) by using Apache Spark in this section. In the proposed framework, the model is learned using offline data. Figure 5.8 shows the process latency of model training by using the Spark-based Random Forest method in MLlib (Meng et al., 2016) ($SAIL_{RF}^{Spark}$) and the Random Forest implementation using Sklearn library (Pedregosa et al., 2011) in Python ($SAIL_{RF}$). To run these experiments, we created the new datasets by duplicating the original datasets for α times, where $\alpha \in \{5, 10 \ldots 30\}$, to show training latency by increasing the volume of the data. We have measured the process latency of the model after a certain number of training instances have been processed and plotted them. Figure 5.8 shows that the Spark-based approach performs significantly better than traditional methods when the number of training instances is increased.



(a) NREL

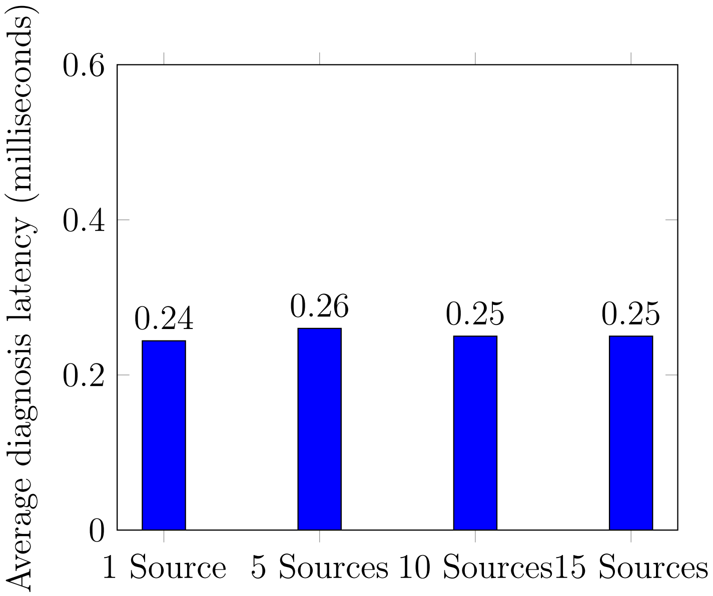


(b) CWRU

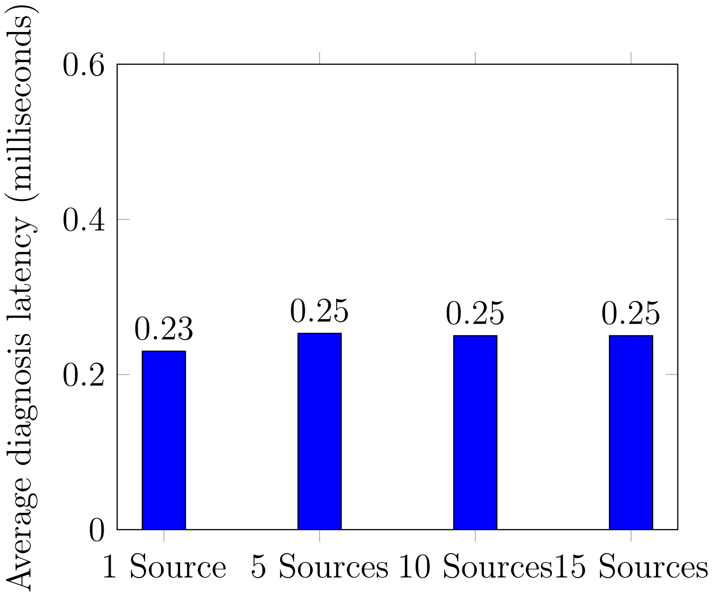
Figure 5.8: Comparing processing latency during training Random Forest model with Mllib in Spark ($SAIL_{RF}^{Spark}$) —×— and Sklearn library ($SAIL_{RF}$) —●— using NREL and CWRU dataset.

Parallelism Test

We have utilized a message broker using Kafka, which is highly available and scalable. It helps us to add more sources to behave as turbine sensors to collect more test data. We vary the number of source instances while we collect data. In fact, these source instances simulate wind turbines to generate data. All of these sources work in parallel and independent of each other. So when the number of sources increases, more data will be produced. Figure 5.9 shows the average process latency of fault diagnosis of online data. In this figure, we see that the average process latency of fault diagnosis does not increase when the number of sources increases. For a single source, the diagnosis latency for each signal is almost 0.24 ms, but it remains constant for α input sources, where $\alpha \in \{5, 10, 15\}$. Thus, the diagnosis latency for each data does not grow while we increase the number of input sources. As a result, the proposed system is scalable.



(a) NREL



(b) CWRU

Figure 5.9: Average diagnosis latency for real-time data from different number of input sources

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this chapter, we will draw a conclusion and discuss the future scope of our works. We will start with our big data primary focus location extraction frameworks and later we will focus on our smart contract vulnerability detection and wind turbine fault diagnosis frameworks.

6.1 Primary Focus Location Extraction

We showcased and developed a focus location extraction method executable on unstructured text-based news reports from different languages. In this method, we proposed a semantic approach to find the focus location among all the possible locations extracted using the named entity recognition tool. Firstly, we extracted the features using the sentence embedding algorithm. In this algorithm, we encoded the meaning of words and their relationship semantically into a vector regardless of language using the fastText_multilingual model. We then trained an SVM classifier to predict sentences which contain focus or non-focus locations. Finally, we used the proposed method on an Atrocity news event dataset, a subset of New York Times corpus, a Spanish news event dataset from Revista Noche y and an Arabic news event dataset from Alghad.

We applied the domain adaptation technique and conducted experiments over two different domains, since the training and test domain are not always the same. The experiment demonstrated the effectiveness of KMM in extracting focus location when there is bias between different domains.

Our key contribution in this work is extracting the exact focus location at the locality level where an event occurred. The proposed approach works based on the semantic relationship among the words in the sentences and is independent of a geographical dictionary. The performance of our approach exceeds other methods considerably. Furthermore, we proposed the use of a bias correction method to prevent performance loss when the training and test domains are dissimilar.

One of the directions for our future research is extracting primary focus location based on the event. In the current approach, we assume that exactly one event is reported in each news article. Consequently, we extract only one primary focus location from the article. However, this assumption may not be true for all news articles. Therefore, we are going to extract one or multiple event(s) from news articles. The events include different action(s) and some actors. Then, we are going to extract multiple primary focus location(s) related to these events.

Another direction of our future work is developing some other bias correction method to improve the results over various domains in news reports. In that case, we can expand our dataset to different domains such as Sport news.

6.2 Vulnerability Detection in Smart Contracts

In this study, we proposed a new static analysis framework to detect vulnerabilities in Ethereum smart contracts. In this framework, we collected 176K unique smart contracts from the Ethereum blockchain. Then, we disassembled the low-level bytecode to higher level intermediate representation in the form of opcodes. In feature engineering stage, we extracted control flow graphs from opcodes and then converted the graphs to vectors by traversing through it. In the last stage, we used those vectors as inputs in BiLSTM model. Finally, we performed a large-scale empirical experiment by running the proposed method on all unique smart contracts which we collected in the initial step. We demonstrated that our scalable framework outperformed the other machine learning-based models.

Due to the lack of labeled data, we limited our benchmark on a few vulnerabilities including ordering, timestamp, mishandle exception, and reentrancy which can be detected by the Oyente tool. In future work, we are going to employ our approach on more vulnerabilities.

Besides, we are going to apply probabilistic graphical models (Smith et al., 2017; Rouhani et al., 2018) in our prediction algorthms.

Another future direction is developing guard code to avoid the vulnerabilities threat. We are going to develop a system to mitigate against attacks by inserting protection code in the raw bytecode. In other words, this system does not require access to the source code. The system is also unique since the Ethereum Virtual machine is different from existing computer architectures like x86 CPUs. To complete the bytecode rewriting, we are going to develop a tool to overcome the following challenges. First, we will develop an algorithm to identify all JUMP locations and rewrite JUMPDEST due to inserted protection code. Then, we will verify the modified code and the original code results in similar outcomes. Other challenges include identifying integer overflow and underflow vulnerabilities. To identify these arithmetic vulnerabilities, we need to determine the bit width size of the resulting variable from the arithmetic operations. In addition, we need to identify all possible JUMP target locations which includes function calls and direct jump calls.

6.3 SAIL: A Scalable Wind Turbine Fault Diagnosis Framework

In this study, we proposed a new data-driven fault diagnosis. In particular, we proposed the model-based spectral analysis as a feature extraction method for the vibration signal. We also presented a real-time wind turbine fault diagnosis framework based on Apache Spark. The experimental results show that the extracted features help to outperform the traditional models' accuracy. The Spark-based framework can reduce the offline training time and improve the performance of the fault prediction in a real-time system. Therefore, we conclude that SAIL is scalable for real-world wind farms.

For future work, we will focus on our fault diagnosis algorithm to make it automatic and plug-&-play. Besides computational requirements in dealing with big data, another challenge emerges from a machine learning perspective. As the size of data increases, its associated variability increases. In traditional machine learning, the variability is reduced by proper feature extraction. Our intuition about the problem inspires our current feature extraction. However, big data is less intuitive. In such a scenario, applying a feature learning algorithm is a possible solution. In the future, we intend to apply a deep neural network for feature learning in fault diagnosis of wind turbines.

APPENDIX

MATHEMATICAL FRAMEWORK FOR FEATURE EXTRACTION FROM WIND TURBINE VIBRATION

In this appendix, we introduce the mathematical framework underlying our proposed feature extraction method in wind turbine fault diagnosis.

A.1 ARMA Representation of Vibration Data

We assume that vibration data is an ARMA time series. Then, we use parameters of this model for feature extraction.

Let $x[n]$ be a stationary time series. A time series model is a mathematical equation for predicting samples of time series. Assuming that each sample of $x[n]$ is a linear combination of past samples leads to the well-known *autoregressive (AR)* model. Mathematically, the AR model is defined by $x[n] = \sum_{k=1}^{p} a_k x[n-k] + \epsilon[n]$ where a_k's are parameters of the model, p is model order and $\epsilon[n]$ is a zero mean white noise with variance of σ_ϵ^2. A discrete sinusoid function with frequency f_0, $x[n] = sin(2\pi f_0 n)$, can be written as a second order AR model without noise as $x[n] = a_1 x(n-1) + a_2 x(n-2)$ where $a_1 = 2cos(2\pi f_k)$ and $a_2 = 1$ with the initial condition of $x(-1) = -1$ and $x(-2) = 0$. By taking the Z transform, one can show that this model has a pair of poles at $f = \pm f_0$. Generally, a sum of p noise-free sinusoids, $x[n] = \sum_{i=1}^{p} sin(2\pi f_k)$, can be written as an AR model with order of $2p$:

$$x[n] = \sum_{k=1}^{2p} a_k x[n-k] \tag{A.1}$$

This model has a transform function in the form of

$$H(z) = \frac{1}{1 + \sum_{k=1}^{2p} a_k z^{-k}} \tag{A.2}$$

which has $2p$ poles corresponding to frequency of sines at $f = \pm f_k$. This transfer function in frequency domain models the spectrum of the time series. If the sum of sines is corrupted by additive noise, like $y[n] = x[n] + \epsilon[n]$, the AR model assumption is not valid. However, one can write an AR model for $x[n] = y[n] - \epsilon[n] = \sum_{k=1}^{2p} a_k(y[n-k] - \epsilon[n-k]))$ which leads to the following model

$$y[n] = \sum_{k=1}^{2p} a_k y[n-k] + \sum_{k=0}^{2p} a_k \epsilon[n-k] \tag{A.3}$$

Compared to Eqn. A.1, this has a moving average term which filters the noise. This model is a special case of ARMA model. A general ARMA is defined by $y[n] = \sum_{k=1}^{p} a_k y[n-k] + \sum_{k=1}^{q} b_k \epsilon[n-k]$ and denoted as $ARMA(p,q)$. However, in Eqn. A.3 the moving average and the autoregressive parts are the same. By defining $\boldsymbol{a} = [1, a_1, a_2, \cdots a_{2p}]^t$, $\boldsymbol{y} = [y[n], y[n-1] y[n-2] \cdots y[n-2p]]^t$ and $\boldsymbol{w} = [\epsilon[n], \epsilon[n-1], \epsilon[n-2], c \ldots, \epsilon[n-2p]]^t$ Eqn. A.3 can be written in matrix form as $\boldsymbol{y}^t\boldsymbol{a} = \boldsymbol{w}^t\boldsymbol{a}$. By pre-multiplying this equation by $\boldsymbol{y}$ and taking the expectation, one can get $E[\boldsymbol{y}\boldsymbol{y}^t]\boldsymbol{a} = E[\boldsymbol{y}\boldsymbol{w}^t]\boldsymbol{a} = E[(\boldsymbol{x} + \boldsymbol{w})\boldsymbol{w}^t\boldsymbol{a}] = E[\boldsymbol{w}\boldsymbol{w}^t]\boldsymbol{a}$. Intuitively, this is true since $\boldsymbol{a}$ and $\boldsymbol{x}$ are deterministic and $\boldsymbol{w}$ is zero mean and white. If $\boldsymbol{\Gamma}_{yy} = E[\boldsymbol{y}\boldsymbol{y}^t]$ denotes the auto-correlation matrix of $y[n]$, then, $\boldsymbol{\Gamma}_{yy}\boldsymbol{a} = \sigma_\epsilon^2\boldsymbol{a}$. Furthermore, it can be written as an eigenvalue problem as:

$$(\boldsymbol{\Gamma}_{yy} - \sigma_\epsilon^2\boldsymbol{I})\boldsymbol{a} = \boldsymbol{0} \tag{A.4}$$

where σ_ϵ^2 is eigenvalue and $\boldsymbol{a}$ is eigenvector. By solving Eqn. A.4, ARMA parameters and noise variance can be found. In practice, $\boldsymbol{\Gamma}_{yy}$ is not available and needs to be estimated which is discussed in the next sub-section.

A.2 Harmonic Decomposition

Consider a summation of p randomly phased sinusoid with white noise. Let A_i and $P_i = A_i^2/2$ denote the amplitude and power of the ith sinusoid in summation. Considering the white noise assumption, the auto-correlation function has a form of $\gamma(k) = \sigma_\epsilon^2\delta(k) + \sum_{i=1}^{p} P_i \, cos(2\pi f_i k)$.

This auto-correlation can be written in a matrix form as:

$$\begin{bmatrix} cos(2\pi f_1) & cos(2\pi f_2) & \cdots & cos(2\pi f_p) \\ cos(4\pi f_1) & cos(4\pi f_2) & \cdots & cos(4\pi f_p) \\ \vdots & \vdots & \ddots & \vdots \\ cos(2p\pi f_1) & cos(2p\pi f_2) & \cdots & cos(2p\pi f_p) \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_p \end{bmatrix} = \begin{bmatrix} \gamma_{yy}(1) \\ \gamma_{yy}(2) \\ \vdots \\ \gamma_{yy}(p) \end{bmatrix} \tag{A.5}$$

Having a set of samples of $y[n]$, there is maximum likelihood estimator for the auto-correlation function as $\hat{\gamma}_{yy}[k] = \sum_n y[n]y[n+k]$. If frequencies are known by calculating Eqn. A.5 and estimating $\hat{\gamma}_{yy}$, one can estimate sine powers using Eqn. A.5. The remaining problem is to determine the p frequencies which can be estimated by finding poles of Eqn. A.2 which in turn requires the knowledge of the eigenvector $\boldsymbol{a}$ in Eqn. A.4. For solving the eigenvalue problem in Eqn. A.4, the matrix $\mathbf{\Gamma}_{yy}$ can be formed using $\hat{\gamma}_{yy}$. For any arbitrary dimension of $\mathbf{\Gamma}_{yy}$ like $d \times d$, this matrix has d eigenvectors. However, the eigenvector equivalent to noise variance is needed which requires the knowledge of noise variance. One can show that the noise variance, σ_ϵ^2, is equivalent to the minimum eigenvalue of $\mathbf{\Gamma}_{yy}$ when $d \geq (2p+1)$ (Stoica, 2005; Breen et al., 2002).

Finally, the power spectrum of the measured vibration signal ($y[n]$) can be estimated as the following: First, estimate $\hat{\gamma}_{yy}[k]$ for $d = (2p+1)$ and form the matrix $\mathbf{\Gamma}_{yy}$. Then, eigenvalues of $\mathbf{\Gamma}_{yy}$ need to be found and the minimum eigenvalue is equivalent to σ_ϵ^2. The corresponding eigenvector holds the parameters of the model $ARMA(2p, 2p)$. Using these parameters, one can find poles of the transfer function in Eqn. A.2 which gives the frequencies. Having these frequencies and using Eqn. A.5, one can find amplitude of each frequency and thus the spectrum of the signal. This method is usually referred to as the Pisarenko power spectrum estimation (Stoica, 2005) which resolves frequencies with arbitrary accuracy not limited to $\frac{F_s}{2N}$. Due to this fact, model-based spectral analysis is sometimes referred to as *super resolution* method (Stoica, 2005).

Generally speaking, the vibration data collected from a mechanical machine is not a stationary random process. However, the assumption of stationarity is valid over short windows of times. We verified this assumption in our work using a hypothesis test suggested by Kwiatkowski et al (Kwiatkowski et al., 1992). The details of this test are beyond the scope of this study. Briefly, we propose a grid search mechanism over different values of window length; we choose the smallest length for which the hypothesis test does not fail.

A.3 Model Order Selection

The above-mentioned method estimates the spectrum lines using an ARMA model for vibration data. However, in developing this model it is assumed that the model order or number of sinusoids is known. In practice, the exact number of sinusoids (p) is not known. However, from the physical modeling perspective, one can estimate p by statistical methods.

For a fitted model, one can calculate $\hat{y}[n] = \sum_{k=1}^{p} A_k sin(2\pi f_k n)$ and form the residual signal as $e[n] = y[n] - \hat{y}[n]$, which is equivalent to our estimation for noise $\epsilon[n]$. However, since $\epsilon[n]$ is assumed white zero mean Gaussian, one can calculate the likelihood of the model as $\mathcal{L} = \prod_i [(2\pi\sigma_\epsilon^2)^{-1/2} exp(e[n]/\sigma_\epsilon)^2]$ with the assumption of independent and identical distribution. In a good estimation, each sample of residual, $e[n]$ is close to zero and its probability becomes bigger and as a result, the probability of all samples becomes large.

A higher p leads to a higher likelihood and lower error. However, a too high p is too sensitive to noise. For balancing this trade-off, the Akike information criteria (AIC) (Stoica, 2005) is used. AIC puts a penalty term on the likelihood ($\mathcal{L}$):

$$AIC = 2p - 2ln(\mathcal{L}) \tag{A.6}$$

which penalizes a $2p$ term for number of parameters in model. Given a set of candidate model from data, the model with minimum AIC is preferred. Here, we sweep an interval for

p around our initial guess-based physical modeling and a model order with minimum AIC will be chosen.

A.4 Dictionary Learning and Compression

A bottleneck in proposing a cloud based solution for monitoring wind-farms is the connectivity bandwidth. The majority of wind-turbines are installed offshore where providing the network connectivity is challenging and expensive. The sensor node extracts ARMA coefficients over rolling window of vibration data. The stream of ARMA coefficients is fed to fog server. Here, the fog server compresses this stream and sends them to the cloud. In this section, the proposed compression algorithm is reviewed.

The basic concept in compression is assigning shorter codes to symbols with higher frequencies such as the well-known Huffman coding algorithm. It is very useful to transform a signal to another domain which leads to a more sparse representation. It has been proven that some transforms like Fourier or Wavelet provide a sparse representation of broad group a time series (Mallat, 1999). Such transforms are exactly or approximately invertible, which provides a mechanism for reconstructing the original time series after decompression. Applying a variable length coding algorithm to a more sparse representation provides a higher compression gain which is desirable. This gain is usually traded off by computation cost of transformation.

Generally, the transformation is done by decomposing the signal as a linear combination of basis functions (*atoms*). In the Fourier transform, these basis functions are complex exponential. In wavelet transform, different wavelet basis functions are proposed. Usually, the set of basis functions, so called *dictionary*, is an orthogonal set which preserves some sort of completeness. Orthogonal basis functions are useful since the inner product operation can be used for finding the decomposition coefficients. A set of n real orthogonal basis functions form a complete basis in $\mathbf{R}^n$, if any arbitrary signal with length n like $\boldsymbol{x} \in \mathbf{R}^n$ can be uniquely represented using the elements of the set. In many traditional compression applications, these basis functions are chosen from existing dictionaries like Fourier or wavelet dictionaries. However, it is possible to design a dictionary for a class of signal to get a better compression rate. In this sub-section, we briefly review the concept of sparse decomposition and the proposed compression algorithm foundation. Here, the term signal refers to the stream of ARMA coefficients extracted in sensor node.

A.5 Sparse Decomposition

Let $\boldsymbol{y} \in \mathbf{R}^n$ denotes a signal with length of n. For a given dictionary of signal atoms like $\boldsymbol{D}$, this signal can be represented as $\boldsymbol{y} = \boldsymbol{D}\boldsymbol{x}$, where $\boldsymbol{x} \in \mathbf{R}^K$ holds the signal representation coefficients in a new domain where K is the number of atoms. If $\boldsymbol{y} = \boldsymbol{D}\boldsymbol{x}$ is satisfied the representation is called exact and the compression is loss-less. In a lossy compression scheme, the representation can be an approximate $\boldsymbol{y} \approx \boldsymbol{D}\boldsymbol{x}$, and the error is bounded by satisfying a condition like $\|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}\|_p \leq \epsilon \; \|\boldsymbol{x}\|_p$, where $\|\boldsymbol{x}\|_p = (|x_1|^p + \cdots |x_n|^p)^{1/p}$ is the the l_p norm. In approximation applications, the l_2 norm or Euclidean norm is usually used. Each column of the dictionary matrix $\boldsymbol{D} \in \mathbf{R}^{n \times K}$ is a basis function (atom) where $\boldsymbol{d}_j \in \mathbf{R}^n$ and $\|\boldsymbol{d}_j\|_2 = 1$ for $j = 1, \cdots, K$. If $n < K$ and $\boldsymbol{D}$ is a full rank matrix, the dictionary is called over-complete and there is an infinite number of solutions for $\boldsymbol{x}$. One may choose $\boldsymbol{x}$ such that it results in a sparse solution, leading to a simpler representation. The sparsity can be quantized using l_0 norm which is equivalent to the number of nonzero elements of a vector. So, the sparse decomposition can be found by solving the following optimization:

$$\boldsymbol{x}^* = \arg\min \boldsymbol{x} \|\boldsymbol{x}\|_0 \qquad \textit{subject to } \boldsymbol{y} = \boldsymbol{D}\boldsymbol{x} \tag{A.7}$$

where $\|\boldsymbol{x}\|_0$ is the number of nonzero elements in $\boldsymbol{x}$. In approximate representation, the constraint in Eqn. A.7 will be replaced by $\|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}\|_2 \leq \epsilon$.

If the dictionary is orthogonal, the decomposition coefficients can easily be obtained by inner product operator. However, for sparse decomposition of a signal on an over-complete dictionary requires solving Eqn. A.7. An optimal algorithm for solving the problem in such a scenario is *Orthogonal matching pursuit* (OMP) (Davis et al., 1997). OMP has two steps in each iteration: (i) sweep (ii) update. These two steps are applied in a loop to minimize the residual of signal which is the difference of the signal and its sparse approximation. In the first iteration, in the sweep step, the inner product of the signal and all dictionary atoms are calculated and the atom with largest absolute value of inner product is selected. In the update step the residual of signal as the difference between $\boldsymbol{argument}$ y and $\boldsymbol{Dx}$ will be calculated and is projected to the space orthogonal to the span of all selected atoms.

A.6 Dictionary Learning Using K-SVD

In this sub-section the problem of dictionary learning is discussed which is an unsupervised problem. The proposed monitoring system uses the K-SVD algorithm (Aharon et al., 2006). This algorithm is run on the cloud to extract the compression dictionary, D. Then, the learned dictionary is sent to fogs for running the compression task.

Basically, K-SVD is a generalization of the well-known *k-means clustering* algorithm. It uses singular value decomposition (SVD) method in linear algebra to reduce the error of learning atoms of a dictionary. Suppose a set of N training signals like $\{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_N\}$ is given. This set can be represented as a matrix form of $\boldsymbol{Y} = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_N]$ which is an $n \times N$ matrix. Each $\boldsymbol{y}_i$ in this set has a sparse representation like $\boldsymbol{x}_i$. Let $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]$ denote the corresponding sparse representation of given training set. In Eqn. A.7 the sparse representation of a single vector is obtained by an optimization. Here, the dictionary needs to be chosen such that this representation is as sparse as possible for all vectors. So, the dictionary can be found by minimizing the sum of objective function in Eqn. A.7 for all vectors as $\arg\min \boldsymbol{D}, \boldsymbol{X} \sum_i \|\boldsymbol{x}_i\|_0$ subject to $\|\boldsymbol{y}_i - \boldsymbol{D}\boldsymbol{x}_i\|_2 \leq \epsilon$ for $i = 1, \cdots, N$. Using the matrix form, this cost can be equivalently written as:

$$< \boldsymbol{D}^*, \boldsymbol{X}^* >= \arg\min \boldsymbol{D}, \boldsymbol{X} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F \qquad \textit{subject to}\ \forall i, \|\boldsymbol{x}\|_0 \leq T_0 \tag{A.8}$$

where $\|\cdot\|_F$ is the Frobenius norm which is simply the sum of square of all elements. Minimizing the objective function in Eqn. A.8 minimizes both the error for choosing the sparse representation of a vector and the choice of the dictionary atoms.

K-SVD is an iterative algorithm which has two steps in each iteration: (i) Sparse coding step and (ii) codebook update step. In the sparse coding step, the approximate sparse representation of all training vectors, $\boldsymbol{Y}$, is obtained using a pursuit algorithm like OMP by solving the optimization in Eqn. A.7 to obtain $\boldsymbol{X}$. Next, in the codebook update stage, a single atom in $\boldsymbol{D}$, let's say $\boldsymbol{d}_k$, will be updated. For this purpose, all vectors in $\boldsymbol{X}$ which used $\boldsymbol{d}_k$ are grouped together. Then, the error of choosing $\boldsymbol{d}_k$ is minimized. For this purpose the SVD is used to find a rank 1 estimation of this error. In the next iteration, another atom in the dictionary will be updated. The details of K-SVD are beyond the scope of this study and reader can refer to (Aharon et al., 2006) for more details.

REFERENCES

Abrol, S. and L. Khan (2010). Twinner: understanding news queries with geo-content using twitter. In *Proceedings of the 6th Workshop on Geographic information Retrieval*, pp. 1–8.

Aharon, M., M. Elad, and A. Bruckstein (2006). K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing 54*(11), 4311–4322.

Al-Khateeb, T., M. M. Masud, L. Khan, C. Aggarwal, J. Han, and B. Thuraisingham (2012). Stream classification with recurring and novel class detection using class-based ensemble. In *2012 IEEE 12th International Conference on Data Mining*, pp. 31–40. IEEE.

Alex, B., K. Byrne, C. Grover, and R. Tobin (2015). Adapting the Edinburgh geoparser for historical georeferencing. *International Journal of Humanities and Arts Computing 9*(1), 15–35.

Amar, M., I. Gondal, and C. Wilson (2015). Vibration spectrum imaging: A novel bearing fault classification approach. *IEEE Transactions on Industrial Electronics 62*(1), 494–502.

Amitay, E., N. Har'El, R. Sivan, and A. Soffer (2004). Web-a-where: geotagging web content. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 273–280. ACM.

Anastácio, I., B. Martins, and P. Calado (2009). A comparison of different approaches for assigning geographic scopes to documents. *Proceedings of the 1st INForum-Simpósio de Informática*, 285–296.

Arora, S., Y. Liang, and T. Ma (2017). A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations. To Appear.*

Awad, M., L. Khan, F. Bastani, and I.-L. Yen (2004). An effective support vector machines (svms) performance using hierarchical clustering. In *16th IEEE international conference on tools with artificial intelligence*, pp. 663–667. IEEE.

Awad, M., L. Khan, and B. Thuraisingham (2008). Predicting www surfing using multiple evidence combination. *The VLDB Journal 17*(3), 401–417.

Awad, M. A. and L. R. Khan (2007). Web navigation prediction using multiple evidence combination and domain knowledge. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 37*(6), 1054–1062.

Ben-David, S., J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan (2010). A theory of learning from different domains. *Machine learning 79*(1-2), 151–175.

Breen, C., L. Khan, and A. Ponnusamy (2002). Image classification using neural networks and ontologies. In *Proceedings. 13th International Workshop on Database and Expert Systems Applications*, pp. 98–102. IEEE.

Breidenbach, L., P. Daian, A. Juels, and E. G. Sirer. An in-depth look at the parity multisig bug. `http://hackingdistributed.com/2017/07/22/deep-dive-parity-bug/`. (Accessed on 01/30/2018).

Brent, L., A. Jurisevic, M. Kong, E. Liu, F. Gauthier, V. Gramoli, R. Holz, and B. Scholz (2018). Vandal: A scalable security analysis framework for smart contracts. *arXiv preprint arXiv:1809.03981*.

Buterin, V. et al. (2014). A next-generation smart contract and decentralized application platform. *white paper 3*(37).

Chen, J. and R. J. Patton (2012). *Robust model-based fault diagnosis for dynamic systems*, Volume 3. Springer Science & Business Media.

Chen, Z., L. Zhang, Z. Wang, W. Liang, and Q. Li (2007). Research and application of data mining in fault diagnosis for big machines. In *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, pp. 3729–3734. IEEE.

Cohn, A., T. West, and C. Parker (2017). Smart after all: blockchain, smart contracts, parametric insurance, and smart energy grids. *Georgetown Law Technology Review 1*(2), 273–304.

Consensys. Known attacks. `https://consensys.github.io/smart-contract-best-practices/known_attacks/`. (Accessed on 01/30/2018).

Criminisi, A., J. Shotton, E. Konukoglu, et al. (2012). Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends® in Computer Graphics and Vision 7*(2–3), 81–227.

Davis, G., S. Mallat, and M. Avellaneda (1997). Adaptive greedy approximations. *Constructive approximation 13*(1), 57–98.

De Moura, L. and N. Bjørner (2008). Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340. Springer.

Ding, J., L. Gravano, and N. Shivakumar (1999). Computing geographical scopes of web resources.

D'Ignazio, C., R. Bhargava, E. Zuckerman, and L. Beck (2014). Cliff-Clavin: Determining geographic focus for news. *NewsKDD: Data Science for News Publishing, at KDD 2014*.

Eisenstein, J., B. O'Connor, N. A. Smith, and E. P. Xing (2010). A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 1277–1287. Association for Computational Linguistics.

Erik, V. (2008). Cluster-centric approach to news event extraction. *New Trends in Multimedia and Network Information Systems 181*, 276.

Estabrooks, A., T. Jo, and N. Japkowicz (2004). A multiple resampling method for learning from imbalanced data sets. *Computational intelligence 20*(1), 18–36.

Finkel, J. R., T. Grenager, and C. Manning (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 363–370. Association for Computational Linguistics.

Fleischman, M. (2001). Automated subcategorization of named entities. In *ACL (Companion Volume)*, pp. 25–30.

geoparser. [online]. *URL: Available: https://geoparser.io/.*

Griggs, K. N., O. Ossipova, C. P. Kohlios, A. N. Baccarini, E. A. Howson, and T. Hayajneh (2018). Healthcare blockchain system using smart contracts for secure automated remote patient monitoring. *Journal of medical systems 42*(7), 130.

Gritta, M., M. T. Pilehvar, N. Limsopatham, and N. Collier (2017). What's missing in geographical parsing? *Language Resources and Evaluation*, 1–21.

Gunasekaran, A. K., M. B. Imani, L. Khan, C. Grant, P. T. Brandt, and J. S. Holmes (2018). Sperg: Scalable political event report geoparsing in big data. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 187–192. IEEE.

Haddad, R. Z., C. A. Lopez, J. Pons-Llinares, J. Antonino-Daviu, and E. G. Strangas (2015). Outer race bearing fault detection in induction machines using stator current signals. In *Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on*, pp. 801–808. IEEE.

Harmouche, J., C. Delpha, and D. Diallo (2015). Improved fault diagnosis of ball bearings based on the global spectrum of vibration signals. *IEEE Transactions on Energy Conversion 30*(1), 376–383.

Hastie, T. (2009). *The elements of statistical learning : data mining, inference, and prediction.* New York: Springer.

Hu, Q., Z. He, Z. Zhang, and Y. Zi (2007). Fault diagnosis of rotating machinery based on improved wavelet package transform and svms ensemble. *Mechanical Systems and Signal Processing 21*(2), 688–705.

Huang, J., A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola (2006). Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pp. 601–608.

Imani, M. B., S. Chandra, S. Ma, L. Khan, and B. Thuraisingham (2017). Focus location extraction from political news reports with bias correction. In *Big Data (Big Data), 2017 IEEE International Conference on*, pp. 1956–1964. IEEE.

Imani, M. B., M. Heydarzadeh, S. Chandra, L. Khan, and M. Nourani (2019). Sail: A scalable wind turbine fault diagnosis platform. In *International Conference on Information Reuse and Integration*, pp. 96–118. Springer.

Imani, M. B., M. Heydarzadeh, L. Khan, and M. Nourani (2017). A scalable spark-based fault diagnosis platform for gearbox fault diagnosis in wind farms. In *2017 IEEE International Conference on Information Reuse and Integration (IRI)*, pp. 100–107. IEEE.

Imani, M. B., L. Khan, and B. Thuraisingham (2019). Where did the political news event happen? primary focus location extraction in different languages. In *2019 IEEE 5th International Conference on Collaboration and Internet Computing (CIC)*, pp. 61–70.

Immovilli, F., M. Cocconcelli, A. Bellini, and R. Rubini (2009). Detection of generalized-roughness bearing fault by spectral-kurtosis energy of vibration or current signals. *Industrial Electronics, IEEE Transactions on 56*(11), 4710–4717.

Kafka, A. (2014). A high-throughput, distributed messaging system. *URL: kafka.apache.org as of 5*(1).

Kalra, S., S. Goel, M. Dhawan, and S. Sharma (2018a). Zeus: Analyzing safety of smart contracts. In *NDSS*.

Kalra, S., S. Goel, M. Dhawan, and S. Sharma (2018b). Zeus: Analyzing safety of smart contracts. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*.

Kanamori, T., S. Hido, and M. Sugiyama (2009). A least-squares approach to direct importance estimation. *The Journal of Machine Learning Research 10*, 1391–1445.

King, D. Mitll/mitie.

King, D. E. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research 10*(Jul), 1755–1758.

Kinsella, S., V. Murdock, and N. O'Hare (2011). I'm eating a sandwich in glasgow: modeling locations with tweets. In *Proceedings of the 3rd international workshop on Search and mining user-generated contents*, pp. 61–68. ACM.

Konstantopoulos, G. How to secure your smart contracts: 6 solidity vulnerabilities and how to avoid them (part 1). https://medium.com/loom-network/how-to-secure-your-smart-contracts-6-solidity-vulnerabilities-and-how-to- avoid-them-part-1-c33048d4d17d. (Accessed on 01/30/2018).

Krupp, J. and C. Rossow (2018). teether: Gnawing at ethereum to automatically exploit smart contracts. In *27th USENIX Security Symposium (USENIX Security 18)*, Baltimore, MD, pp. 1317–1333. USENIX Association.

Kusiak, A. and W. Li (2011). The prediction and diagnosis of wind turbine faults. *Renewable Energy 36*(1), 16–23.

Kwiatkowski, D., P. C. Phillips, P. Schmidt, and Y. Shin (1992). Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of econometrics 54*(1-3), 159–178.

Lavee, G., L. Khan, and B. Thuraisingham (2007). A framework for a video analysis tool for suspicious event detection. *Multimedia Tools and Applications 35*(1), 109–123.

Lee, S. and G. G. Lee (2005). Heuristic methods for reducing errors of geographic named entities learned by bootstrapping. In *International Conference on Natural Language Processing*, pp. 658–669. Springer.

Lei, Y., Z. He, Y. Zi, and Q. Hu (2007). Fault diagnosis of rotating machinery based on multiple anfis combination with gas. *Mechanical systems and signal processing 21*(5), 2280–2294.

Leidner, J. L., G. Sinclair, and B. Webber (2003). Grounding spatial named entities for information extraction and question answering. In *Proceedings of the HLT-NAACL 2003 workshop on Analysis of geographic references-Volume 1*, pp. 31–38. Association for Computational Linguistics.

Li, W., S. Zhang, and S. Rakheja (2016). Feature denoising and nearest–farthest distance preserving projection for machine fault diagnosis. *IEEE Transactions on Industrial Informatics 12*(1), 393–404.

Loparo, K. Case western reserve university bearing data center. (Accessed on 04/01/2015).

Luu, L., D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor (2016). Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 254–269. ACM.

Mahamad, A. K. and T. Hiyama (2011). Fault classification based artificial intelligent methods of induction motor bearing. *International Journal of innovative computing, information and control 7*(9), 5477–5494.

Mallat, S. (1999). *A wavelet tour of signal processing.* Academic press.

Manning, C. D., M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL (System Demonstrations)*, pp. 55–60.

Manoochehri, H. E., S. S. Kadiyala, and M. Nourani (2019). Predicting drug-target interactions using weisfeiler-lehman neural network. In *2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pp. 1–4. IEEE.

Martins, B. and M. J. Silva (2005). A graph-ranking algorithm for geo-referencing documents. In *ICDM*, Volume 5, pp. 741–744.

Masud, M., L. Khan, and B. Thuraisingham (2016). *Data mining tools for malware detection.* Auerbach Publications.

Masud, M. M., T. M. Al-Khateeb, K. W. Hamlen, J. Gao, L. Khan, J. Han, and B. Thuraisingham (2008). Cloud-based malware detection for evolving data streams. *ACM transactions on management information systems (TMIS) 2*(3), 1–27.

Masud, M. M., T. M. Al-Khateeb, L. Khan, C. Aggarwal, J. Gao, J. Han, and B. Thuraisingham (2011). Detecting recurring and novel classes in concept-drifting data streams. In *2011 IEEE 11th International Conference on Data Mining*, pp. 1176–1181. IEEE.

Masud, M. M., J. Gao, L. Khan, J. Han, and B. Thuraisingham (2010). Classification and novel class detection in data streams with active mining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 311–324. Springer.

Meng, X., J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al. (2016). Mllib: Machine learning in apache spark. *Journal of Machine Learning Research 17*(34), 1–7.

Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur (2010). Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Montemagni, S. and V. Pirrelli (1998). Augmenting wordnet-like lexical resources with distributional evidence. an application-oriented perspective. In *Usage of WordNet in Natural Language Processing Systems*.

mordecai. [online]. *URL: Available: https://github.com/openeventdata/mordecai*.

Nelwamondo, F. V., T. Marwala, and U. Mahola (2006). Early classifications of bearing faults using hidden markov models, gaussian mixture models, mel frequency cepstral coefficients and fractals. *International Journal of Innovative Computing, Information and Control 2*(6), 1281–1299.

Nessa, S., M. Abedin, W. E. Wong, L. Khan, and Y. Qi (2008). Software fault localization using n-gram analysis. In *International Conference on Wireless Algorithms, Systems, and Applications*, pp. 548–559. Springer.

(NREL), N. R. E. L. (2009 (accessed April 1, 2017)). *Gearbox Reliability Collaborative Research*.

Pan, S. J. and Q. Yang (2010). A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on 22*(10), 1345–1359.

Parveen, P., J. Evans, B. Thuraisingham, K. W. Hamlen, and L. Khan (2011). Insider threat detection using stream mining and graph mining. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pp. 1102–1110. IEEE.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research 12*(Oct), 2825–2830.

Pouliquen, B., M. Kimler, R. Steinberger, C. Ignat, T. Oellinger, K. Blackler, F. Fuart, W. Zaghouani, A. Widiger, A.-C. Forslund, et al. (2006). Geocoding multilingual texts: Recognition, disambiguation and visualisation. *arXiv preprint cs/0609065*.

Pullen, A. and S. Sawyer (2014). Global wind report. annual market update 2014.

Qi, G., W.-T. Tsai, Y. Hong, W. Wang, G. Hou, Z. Zhu, et al. (2016). Fault-diagnosis for reciprocating compressors using big data. In *Big Data Computing Service and Applications (BigDataService), 2016 IEEE Second International Conference on*, pp. 72–81. IEEE.

Raj, A. S. and N. Murali (2013). Early classification of bearing faults using morphological operators and fuzzy inference. *IEEE Transactions on Industrial Electronics 60*(2), 567–574.

Rauch, E., M. Bukatin, and K. Baker (2003). A confidence-based framework for disambiguating geographic terms. In *Proceedings of the HLT-NAACL 2003 workshop on Analysis of geographic references-Volume 1*, pp. 50–54. Association for Computational Linguistics.

Riera-Guasp, M., M. Pineda-Sanchez, J. Perez-Cruz, R. Puche-Panadero, J. Roger-Folch, and J. A. Antonino-Daviu (2012). Diagnosis of induction motor faults via gabor analysis of the current in transient regime. *IEEE Transactions on Instrumentation and Measurement 61*(6), 1583–1596.

Ritter, A., S. Clark, O. Etzioni, et al. (2011). Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1524–1534. Association for Computational Linguistics.

Rouhani, S., T. Rahman, and V. Gogate (2018). Algorithms for the nearest assignment problem. In *IJCAI*, pp. 5096–5102.

Sadilek, A., H. Kautz, and J. P. Bigham (2012). Finding your friends and following them to where you are. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pp. 723–732. ACM.

Samuel L. Smith, David H. P. Turban, S. H. and N. Y. Hammerla (2017). Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Schmidhuber, J. and S. Hochreiter (1997). Long short-term memory. *Neural Comput 9*(8), 1735–1780.

Schrodt, P. and J. Ulfelder (2009). Political instability task force worldwide atrocities dataset. *Lawrence, KS: Univ. Kansas, updated 8*.

Shao, Z., L. Wang, and H. Zhang (2016). A fault line selection method for small current grounding system based on big data. In *Power and Energy Engineering Conference (APPEEC), 2016 IEEE PES Asia-Pacific*, pp. 2470–2474. IEEE.

Sheng, S. and P. S. Veers (2011). *Wind turbine drivetrain condition monitoring-an overview.* National Renewable Energy Laboratory.

Shvachko, K., H. Kuang, S. Radia, and R. Chansler (2010). The hadoop distributed file system. In *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*, pp. 1–10. IEEE.

Silva, M. J., B. Martins, M. Chaves, A. P. Afonso, and N. Cardoso (2006). Adding geographic scopes to web resources. *Computers, Environment and Urban Systems 30*(4), 378–399.

Sinnott, R. O., L. Morandini, and S. Wu (2015). Smash: A cloud-based architecture for big data processing and visualization of traffic data. In *Data Science and Data Intensive Systems (DSDIS), 2015 IEEE International Conference on*, pp. 53–60. IEEE.

Smith, D., S. Rouhani, and V. Gogate (2017). Order statistics for probabilistic graphical models. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence.*

Solaimani, M., R. Gopalan, L. Khan, P. T. Brandt, and B. Thuraisingham (2016). Spark-based political event coding. In *Big Data Computing Service and Applications (BigDataService), 2016 IEEE Second International Conference on*, pp. 14–23. IEEE.

Solaimani, M., M. Iftekhar, L. Khan, B. Thuraisingham, and J. B. Ingram (2014). Spark-based anomaly detection over multi-source vmware performance data in real-time. In *Computational Intelligence in Cyber Security (CICS), 2014 IEEE Symposium on*, pp. 1–8. IEEE.

Spark. [online]. *URL: http://spark.apache.org/.*

Stoica, P. (2005). *Spectral analysis of signals.* Upper Saddle River, N.J: Pearson/Prentice Hall.

Storm, A. [online]- apache storm: distributed and fault-tolerant real-time computation. *URL: Available: http://storm.incubator.apache.org/.*

Sugiyama, M., S. Nakajima, H. Kashima, P. V. Buenau, and M. Kawanabe (2008). Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in neural information processing systems*, pp. 1433–1440.

Sun, W., G. A. Yang, Q. Chen, A. Palazoglu, and K. Feng (2013). Fault diagnosis of rolling bearing based on wavelet transform and envelope spectrum correlation. *Journal of Vibration and Control 19*(6), 924–941.

Tamura, Y., Y. Nobukawa, and S. Yamada (2015). A method of reliability assessment based on neural network and fault data clustering for cloud with big data. In *Information Science and Security (ICISS), 2015 2nd International Conference on*, pp. 1–4. IEEE.

Teitler, B. E., M. D. Lieberman, D. Panozzo, J. Sankaranarayanan, H. Samet, and J. Sperling (2008). Newsstand: A new view on news. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, pp. 18. ACM.

Thuraisingham, B., L. Khan, M. M. Masud, and K. W. Hamlen (2008). Data mining for security applications. In *2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, Volume 2, pp. 585–589. IEEE.

Torres, C. F., J. Schütte, and R. State (2018). Osiris: Hunting for integer bugs in ethereum smart contracts. In *Proceedings of the 34th Annual Computer Security Applications Conference*, ACSAC '18, New York, NY, USA, pp. 664–676. ACM.

Verma, A. and A. Kusiak (2011). Predictive analysis of wind turbine faults: A data mining approach. In *IIE Annual Conference. Proceedings*, pp. 1. Institute of Industrial and Systems Engineers (IISE).

Wang, L., L. Liu, and L. Khan (2004). Automatic image annotation and retrieval using subspace clustering algorithm. In *Proceedings of the 2nd ACM international workshop on Multimedia databases*, pp. 100–108.

Watson, S. J., B. J. Xiang, W. Yang, P. J. Tavner, and C. J. Crabtree (2010). Condition monitoring of the power output of wind turbine generators using wavelets. *Energy Conversion, IEEE Transactions on 25*(3), 715–721.

Wing, B. P. and J. Baldridge (2011). Simple supervised document geolocation with geodesic grids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 955–964. Association for Computational Linguistics.

Xu, Z., J. Xuan, T. Shi, B. Wu, and Y. Hu (2009). Application of a modified fuzzy artmap with feature-weight learning for the fault diagnosis of bearing. *Expert Systems with Applications 36*(6), 9961–9968.

Yen, I.-L., J. Goluguri, F. Bastani, L. Khan, and J. Linn (2002). A component-based approach for embedded software development. In *Proceedings Fifth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing. ISORC 2002*, pp. 402–410. IEEE.

Yin, S. and O. Kaynak (2015). Big data for modern industry: challenges and trends [point of view]. *Proceedings of the IEEE 103*(2), 143–146.

Yu, J. (2014). *GEOTAGGING NAMED ENTITIES IN WEB PAGES*. Ph. D. thesis, University of Alberta.

Yu, Y.-l. and C. Szepesvári (2012). Analysis of kernel mean matching under covariate shift. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pp. 607–614.

Zaharia, M., T. Das, H. Li, S. Shenker, and I. Stoica (2012). Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters. *HotCloud 12*, 10–10.

Zappalá, D., P. J. Tavner, C. J. Crabtree, and S. Sheng (2014). Side-band algorithm for automatic wind turbine gearbox fault detection and diagnosis. *IET Renewable Power Generation 8*(4), 380–389.

Zhang, Z., A. Verma, and A. Kusiak (2012). Fault analysis and condition monitoring of the wind turbine gearbox. *IEEE transactions on energy conversion 27*(2), 526–535.

Zikopoulos, P., C. Eaton, et al. (2011). *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media.

BIOGRAPHICAL SKETCH

Maryam Bahojb Imani received her Bachelor of Science degree in computer science from Shahid Beheshti University, Tehran, Iran in 2007 and her Master of Science degree in computer engineering (Artificial Intelligence) from Alzahra University, Tehran, Iran in 2011. She received a Master of Computer Science from The University of Texas at Dallas in 2019. She is currently a PhD candidate in the Computer Science Department at The University of Texas at Dallas. She has been working in the Big Data Analytics and Management Lab as a Research Assistant since Spring 2016. Her research interests are machine learning, big data analytics and natural language processing.

CURRICULUM VITAE

Maryam Bahojb Imani

February 1, 2020

Contact Information:

Dallas, Texas
Email: maryam.imani@utdallas.edu
Webpage: http://www.utdallas.edu/∼ maryam.imani/

Education:

PhD, **Computer Science**, Jan '16 -*Expected:* Mar '20

University of Texas at Dallas, Richardson, TX, USA

- Advisor: Latifur Khan, Ph.D , Co-Advisor: Bhavani Thuraisingham, Ph.D

MS, **Computer Science**, Jan '16 - May '19

University of Texas at Dallas, Richardson, TX, USA

MS, **Computer Engineering**, Sept '08 - Mar '11

Alzahra University, Tehran, Iran

BS, **Computer Science** , Sept '03 - May '07

Shahid Beheshti University, Tehran, Iran

Practical Experience:

Walmart Labs, CA, USA June '19 to Aug '19

- Data Science Intern
  I worked on product classification project. In this project, we need to do pre-processing and feature engineering on the product description. Then, classify the product in different levels of categories.

University of Texas at Dallas, TX, USA Jan '16 to present

Erik Jonsson School of Engineering and Computer Science

- Research Assistant in *Event Data Mining* project,
  I developed a new method in Python to extract the exact geolocation (focus location) of the events in text-based news reports. I employed a sentence embedding approach using word2vec model to extract features. And I used a bias correction method to train and test the focus location model on two different domains of textual data.
- Research Assistant in *Vulnerability Detection in Smart Contracts* project,
  I converted bytecodes to Control Flow Graphs (CFGs). After extracting some features from these graphs, I employed metric learning to train the model to predict vulnerabilities in unseen Smart Contracts.

Conversant Media, IL, USA May '18 to August '18

- Data Science Intern,
  I worked with Hive and Greenplum DB engines in the Decision Science Team. I used machine learning algorithms to improve its personalization platform. Conversant's personalization is based on the analysis of anonymized data at internet-scale.

Stony Brook University, NY, USA Jan '15 - Jan '16

Computer Science Department,

- Research Associate in *Adverse Drugs Event (ADE)* project,
  Adverse Drug Events (ADEs) are unintended and undesired reactions experienced by an individual due to the use, misuse, or discontinuation of medication. In ADE project, we have developed a new method in Java based on the Probabilistic Bayesian Network by using background information to find the medical drug or drugs' interaction that cause(s) some specific diseases and/or side effects.

Infoamn Consulting Co., Tehran, Iran May '10 – Nov '14

Engineering Department,

- *Member of information system team*,
  Used C#.NET and SQL Server to develop a Management Information System as a team player.

Amirkabir University (Tehran Polytechnic), Tehran, Iran Sept'12- Dec'14

Computer and Information Engineering Department,

- *Research Assistant in Adaptive Learning Path project*,
  We developed a web-based personalized learning path based on some artificial intelligence techniques, such as a Clustering and Swarm Intelligence algorithm using Java and MATLAB.

Alzahra University, Tehran, Iran Sept'08- Dec'11

Computer and Information Engineering Department,

- *Research Assistant in Data Mining lab*
  I proposed and developed a novel method based on ensemble learning and self-training in the field of semi-supervised learning for text classification by using C#.NET, MATLAB, and SQL Server. I also contributed to improving a feature selection method in text classification.

Teaching Experience:

The University of Texas at Dallas) Computer Science Department, Richardson, Texas

- Teaching Assistant of Introduction to Machine Learning (Spring 2018)
- Teaching Assistant of Big Data Analytics and Management (Spring 2019)
- Teaching Assistant of Developing and Securing the Cloud (Fall 2019)

Amirkabir University (Tehran Polytechnic) Computer and Information Engineering Department, Tehran, Iran

- Lecturer of Database Lab (Feb 2014- Jul 2014).
- Teaching Assistant of User Modeling (Feb 2014- Dec 2014)

Alzahra University Computer Engineering Department, Tehran, Iran

- Lecturer of below listed courses (Sep 2012- Dec 2014).
  Introduction to Programming in C/C++, and Special English for Computer Engineering
- Teaching Assistant of below listed courses (Sep 2009- Jun 2012)
  Principle of Compiler Design, Database Design, Operating System, Discrete mathematics, and Algorithm design

Abrar Institute of Higher Education Computer Engineering Department, Tehran, Iran

- Lecturer of below listed courses (Sep 2014- Dec 2014)
  Database Design, and Database Lab

Islamic Azad University(North Tehran branch) Engineering Department, Tehran, Iran

  - Lecturer of Data structures (Sep 2012- Jan 2013)

Technical Skills:

Languages (years)

- Python(5+), MATLAB(3), R(1+), Java(2), C#(2), C++ and C(1), SQL(3)

Tools and DBMSs

- Big Data: Spark and Hadoop
- Databases: Hive, Greenplum, Microsoft SQLServer and MySQL
- Data mining tools: Weka, R and Rapid Miner
- CASE tools: Rational Rose

Techniques

- Artificial Intelligence (Meta-Heuristic Algorithms, Expert System (CLIPS Shell), Pattern Recognition Methods)
- Data Mining and Text Mining
- Natural Language Processing

Honors and Awards:

- Ericsson Graduate Fellowship for the 2017-2018 and 2019-2020 academic years at the University of Texas at Dallas
- Best Paper Award in "IEEE International Conference on Collaboration and Internet Computing" conference (2019)
- Travel Scholarship awards for WiML(2016, 2019), ACM Tapia (2019), CRA-W(2018), and WiCys(2018) conferences
- Iran Telecommunication Research Center Scholarship, Supporting M.Sc. Thesis
- Distinguished Student Award for the 1st rank in M.Sc. at Alzahra University
- Semi-Finalist in the National Computer Science Olympiad of Iran, 2001

Services:

- Reviewer of Knowledge-Based Systems, and Computer Applications in Engineering Education journals, Journal of Experimental Theoretical Artificial Intelligence and International Conference of e-Learning and e-Teaching
- Organizer and Speaker of Women in Data Science (Dallas section) at the University of Texas at Dallas, 2018 and 2019.
- Treasurer and Event coordinator in Iranian Student Community (ISC) organization at the University of Texas at Dallas, 2016 - 2018

Publications:

Google Scholar

- **M. B. Imani**, L. Khan, B. Thuraisingham, Where Did the Political News Event Happen?, *IEEE International Conference on Collaboration and Internet Computing*, Los Angeles, USA, 2019 (Best Paper).
- **M. B. Imani**, M. Heydarzadeh, S. Chandra, L. Khan, M. Nourani, SAIL: A Scalable Wind Turbine Fault Diagnosis Platform, *Theory and Application of Reuse, Integration, and Data Science (96)*, 2019.
- A.K Gunasekaran, **M. B. Imani**, L. Khan, C. Grant, P. Brandt, J. Holmes, SPERG: Scalable Political Event Report Geoparsing in Big Data. To appear in *IEEE Intelligence and Security Informatics (ISI)*, 2018.
- **M. B. Imani**, S. Chandra, S. Ma, L. Khan, B. Thuraisingham, Focus Location Extraction from Political News Reports Across Different Domains. The IEEE Big Data Conference, Boston, USA, 2017.
- **M. B. Imani**, M. Heydarzadeh, L. Khan, M. Nourani, A Scalable Spark-Based Fault Diagnosis Platform for Gearbox Fault Diagnosis in Wind Farms, *Information Reuse and Integration Conference*, 2017.
- A. Kim, **M. B. Imani**, and P. T. Brandt. "Geolocation in Political Event Text Analysis." Conference of the Society for Political Methodology. Madison, WI. 2017 (Poster).
- M. R. Keyvanpour, **M. B. Imani**, Semi-Supervised Text Categorization: Exploiting Unlabeled Data Using Ensemble Learning Algorithms, *Intelligent Data Analysis Journal*, Vol. 17(3), Spring 2013.
- **M. B. Imani**, M. R. Keyvanpour, R. Azmi, A Novel Embedded Feature Selection Method: A Comparative Study in the Application of Text Categorization, *Applied Artificial Intelligence Journal*, Vol. 27(5), pp. 408427,31 May 2013.
- Kardan, **M. B. Imani**, Improving Persian POS tagging using the maximum entropy model, *Intelligent Systems Conference*, Iran, 2014.
- Kardan, **M. B. Imani**, M. Ale Ebrahim, ACO-Map: A Novel Adaptive Learning Path Method, the 7th National and 4th International Conference on *E-Learning and E-Teaching* (ICELET 2013), Iran, February 2013. (Best Paper Awarded).

- **M. B. Imani**, M. R. Keyvanpour, R. Azmi Semi-supervised Persian font recognition, *Procedia Computer Science Journal* Vol. 3 , pp. 336-342, 2010
- **M. B. Imani**, T. Pourhabibi, M.R. Keyvanpour, A New Feature Selection Method Based on Ant colony and Genetic Algorithm on Persian Font recognition, *International Journal of Machine Learning and Computing* (IJMLC), 2010.
- T. Pourhabibi, **M. B. Imani**, S. Haratizade, Feature Selection On Persian Font: a Comparative Analysis on GAA,GESA and GA, *Procedia Computer Science Journal* Vol 3, pp. 1249-1255, 2010.