

M.SC. ENGG. THESIS

Typosquatting Error Detection Analyzing DNS Log

by

Md. Anwar Parvez

Submitted to

Department of Computer Science and Engineering  
in partial fulfillment of the requirements for the degree of  
Master of Science in Computer Science and Engineering


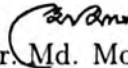

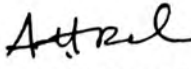
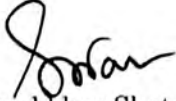


Department of Computer Science and Engineering  
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY  
DHAKA-1000

October, 2019

The thesis titled "Typosquatting Error Detection Analyzing DNS Log", submitted by Md. Anwar Parvez, Roll No. 1014052004 P, Session October 2014, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering on October 2, 2019.

### BOARD OF EXAMINERS

- |   |   |
|---|---|
| <p>1. Dr. Anindya Iqbal<br/>Associate Professor, Department of CSE,<br/>Bangladesh University of Engineering and Technology, Dhaka.</p>   | <br>Chairman<br>(Supervisor) |
| <p>2.  Dr. Md. Mostofa Akbar<br/>Professor and Head, Department of CSE,<br/>Bangladesh University of Engineering and Technology, Dhaka.</p>    | Member  |
| <p>3.  Dr. M Sohel Rahman<br/>Professor, Department of CSE,<br/>Bangladesh University of Engineering and Technology, Dhaka.</p>              | Member  |
| <p>4.  Dr. Atif Hasan Rahman<br/>Assistant Professor, Department of CSE,<br/>Bangladesh University of Engineering and Technology, Dhaka.</p> | Member  |
| <p>5.  Dr. Swakkhar Shatabda<br/>Associate Professor, Department of CSE,<br/>United International University, Dhaka, Bangladesh</p>          | Member<br>(External)  |

## Candidate's Declaration

This is hereby declared that the work titled “Typosquatting Error Detection Analyzing DNS Log” is the outcome of research carried out by me under the supervision of Dr. Anindya Iqbal, in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka. It is also declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.



---

Md. Anwar Parvez  
Candidate

# Acknowledgment

First of all, I would like to thank my supervisor, Dr. Anindya Iqbal, for assisting me throughout the thesis. Without his continuous inspiring enthusiasm, encouragement, supervision, guidance, and advice, not have been complete this thesis. I am especially grateful to him for giving valuable time whenever I needed, and always providing continuous support, motivation and endless patience towards the completion of the thesis. Last but not least, I am grateful to my guardians, families, and friends for their patience, cooperation, and inspiration during this period.

I would like to thank Mr. Arif Ahmed Tanim from BTCL (Bangladesh Telecommunications Company Limited) for providing me log from .bd domain server. I also want to thank Deputy General Manager of domain division Mr. Shohidul Islam for his consent to use their log data in research.

I would also like to thank to my research group member Ishtiyaque Ahmad especially for helping me in thesis implementation. I also want to thank the other members of my thesis committee: Dr. Md. Mostofa Akbar, Dr. M. Sohel Rahman, Dr. Atif Hasan Rahman and specially the external member Dr. Shakkhar Shatabda for their valuable suggestions.

# Abstract

Typosquatting is a form of internet cybersquatting generated from the mistakes (typos) made by internet users while typing a website address. It often leads the user to another unintended website. Sometimes it is exploited by cybersquatters to attract website traffic by redirecting common typos of popular websites to some other sites with malicious contents. A possible solution is defensive registration of similar domains and redirecting requests to the original site. This would be affordable for the owner of the original domain if a short list of such probable typo domain names can be predicted. In this thesis, we present a supervised learning based typographical error detection model analyzing domain server log that would suggest such a list. The detection scheme achieves as high as 98% accuracy. Existing works on typosquatting mostly try to generate typo sites by using different heuristic rules. However, to the best of our knowledge, none of them can predict probable typo variations of a given URL at pre-registration phase. We also present *TypoWriter*, an RNN based error prediction tool to fill this gap. *TypoWriter* achieves a good performance in terms of successful predictions that match with the ground-truth. It is compared with five widely used typo generation tools and substantial improvement is observed.

# Contents

<b><i>Board of Examiners</i></b>	i
<b><i>Candidate's Declaration</i></b>	ii
<b><i>Acknowledgment</i></b>	iii
<b><i>Abstract</i></b>	iv
<b>1 Introduction</b>	1
1.1 Significance	1
1.2 Limitations of Previous Works and Our Objective	2
1.3 Research Questions	4
1.4 Overview of Our Solution	5
1.5 Outline of the thesis	6
<b>2 Literature Review</b>	7
2.1 Research on Typosquatting	7
2.2 Existing Typo Generation Tools	9
2.3 Recurrent Neural Network	9
2.4 Summary	10
<b>3 Our Proposed Framework</b>	11
3.1 Detection from Log Analysis	11
3.1.1 Data Generation	12
3.1.2 Data Cleaning	12
3.1.3 Extraction of Typo Pair using Deterministic Rules	12
3.1.3.1 <i>Edit distance based estimation</i>	12
3.1.3.2 <i>Time difference</i>	13

3.1.3.3	Same IP	14
3.1.4	Gold-set Generation	14
3.1.5	Feature Selection	14
3.1.6	Classifier Design	15
3.2	Pre-registration Prediction	16
3.2.1	The Concept	16
3.2.1.1	Variable Length of Sequences	18
3.2.1.2	Global vs Local Dependency	18
3.2.2	Construction of n-grams	19
3.2.3	Training RNN Model	20
3.2.3.1	Preprocessing of Training Dataset	20
3.2.3.2	Model Architecture	21
3.2.4	Prediction	21
3.2.5	Ranking	22
3.3	Summary	23
<b>4</b>	<b>Result and Performance Evaluation</b>	<b>24</b>
4.1	Data Collection and Dataset	24
4.2	Experimental Setup	25
4.3	Detection Performance	25
4.4	Prediction Performance	27
4.4.1	Impact of Substring Length	27
4.4.2	Impact of $n_r$	29
4.4.3	Comparison with Generation Tools	30
4.5	Discussion	33
4.6	Summary	37
<b>5</b>	<b>Empirical Study</b>	<b>38</b>
5.1	Significant number of misspelled URLs exist in the DNS Log with single character modification of original URLs	38
5.2	Short original URLs suffer more from URL poaching	39
5.3	Typographical error can be categorized in different types	40
<b>6</b>	<b>Conclusion</b>	<b>44</b>

# List of Figures

3.1 Overview of our detection scheme . . . . .	11
3.2 Detailed approach to extract typo pairs from initially generated log . . . . .	13
3.3 Methodology followed for the construction of <i>TypoWriter</i> . . . . .	17
4.1 Matching ratios for 10 different domain names . . . . .	29
4.2 The increase of expected number of top typo sites with the increase of number of defensive domain registered . . . . .	30
4.3 Performance comparison of <i>TypoWriter</i> with other typo generation tools. . . . .	34
4.4 Number of typo visits by target site's popularity . . . . .	34
4.5 Number of typo visits by target site's category . . . . .	35
5.1 Percentage of modification types that caused typos . . . . .	40
5.2 Frequency of typo hit for different URL lengths . . . . .	40



# List of Tables

3.1	Features used for typosquatting detection	14
3.2	N-gram pairs constructed from some sample data	20
3.3	Model architectures with the minimum validation loss for different values of $n$	22
4.1	Parameters used for detection classifiers	26
4.2	10-fold cross validation performance measures using different machine learning models for detection task	27
4.3	Comparison of different machine learning models on test dataset	27
4.4	Number of correctly detected typo domain name out of top 5 most occurred typo sites for 10 different domain names	28
4.5	Matching ratios for 10 different domain names	29
4.6	Typo candidates generated by prediction models for $n = 2, 3$ .	31
4.7	Typo candidates generated by prediction models for $n = 4, 5$ .	32
4.8	The impact of DNS log volume on detection accuracy	36
4.9	Comparison among proposed methods with passive detection methodology and n-gram based method	36
4.10	Features used for typosquatting detection along with information gain ratio obtained for each feature	37
5.1	Character repetition, omission and replacement example	39
5.2	Character repetition, omission and replacement example	39
5.3	Different type of typosquatting error and their proportion	41
5.4	Keyboard proximity typosquatting for horizontal and vertical orientation	41
5.5	Dash and Dot related typosquatting error	43

# List of Algorithms

3.1 Algorithms for constructing n-gram . . . . .	19
--	----

# Chapter 1

## Introduction

Typosquatting refers to the mistakes (typos) made by internet users while typing a website address that is likely to lead the user to another unintended website, possibly owned by a malicious entity. The malicious people may register the domain name similar to a target website. Two different domain names may be similar in various ways. Some domain names are visually similar, some are phonetically, and some names may result from spelling error of others. Many Bengali popular names can be written with different similar phonetic spellings. Perpetrators may use these similar domain names to confuse valuable customers of popular domains. Registration of misspelling of popular domains is done by typosquatters for making money out of traffic from unintentional typing mistakes or fat finger errors made by internet surfers. Hence, this is considered a significant cybersecurity threat, detection and prevention of typosquatting is a challenging problem.

### 1.1 Significance

Typosquatting often leads to the way of dangerous cyber attacks. Hackers create fake websites that imitate the look and feel of the user's intended destination so that the user may not realize while visiting a different site. Thus typosquatting leads to phishing attacks. Sometimes these fake websites attempt to sell products and services that are in direct competition with products and services sold at the actually intended website. Often they are intended to steal users' personal information, such as credit card number, PIN, password, etc. Also, malicious software can be downloaded to users' devices simply from visiting these sites. Users do not even need to click on a link or accept a download request for dangerous software to be installed on his/her

computer. This is called *drive-by download* [1], which is responsible for many serious security breaches.

Typosquatting causes a lot of monetary loss each and every day. It is estimated that typosquatting costs the brands associated with most frequently visited 250 dot-COM sites \$285 million per year due to unnecessary advertising costs, loss of sales, and poor user experiences [2]. The top 255 most visited sites lose \$265,180,586 to typosquatters each year. Brands lose an additional \$19,986,288 as a result of misusing the typo domains that they own. The losses due to cybercrime are not always monetary in nature. Over 120,000 e-mails and 20 GB of corporate data were stolen from Fortune 500 Companies via typosquatting [3]. It is found that typosquatting costs a typical user 1.3 seconds per typosquatting event compared to receiving a browser error page, and legitimate sites lose approximately 5% of their mistyped traffic owing to the alternative of an unregistered typo [4].

## 1.2 Limitations of Previous Works and Our Objective

Domain name policy provides the basis for the existence of cybersquatters and typosquatters because it allows registration on a first-come, first served basis. From a website owner's perspective, one way of mitigating the loss occurred by typosquatting attacks is registering domain names similar to the original one (in other words, possible typo variations of the original one) before a typosquatter can do so. This scheme is known as defensive domain name registration. Owners and companies can develop dummy sites in those domains and redirect typo-requests (that land in those sites) to the original one.

In order to gain knowledge about typosquatting attacks and unintentional errors made by a user, a post-hoc analysis is commonly used. Some existing tools [4] [5] try to *detect* highly probable candidates of typosquatting errors for a given URL using this approach. Moore et. al. [6] in the year 2010 described a method for identifying typosquatting by generating possible misspelling with Levenshtein and fat-finger distance and crawled typo domains to analyze their revenue sources. Khan et al. [4] proposed an approach for detecting typosquatting by passively looking for domain resolutions and HTTP traffic within a live network. This work used time-based metrics and Levenshtein edit distance for clustering typo pair. Piredda et.

al. [5] addressed typosquatting detection by passive ISP traffic analysis using machine learning. Only n-gram based representation of typo domains are used as the feature set to identify typosquatting URL and hence it achieved unsatisfactory performance.

Detection of typosquatting enables a site owner to get insights regarding the type of errors users are making while typing a URL. Yet, in most of the cases, the visited typo sites are already registered by typosquatters and hence cannot be used for defensive registration any longer. In contrast, if the owner can get a list of most probable typo variations of a domain name before registration, she might make defensive registrations for those typo domain names and thus effectively prevent typosquatting attacks. Therefore, we argue that prevention is a better measure than detection in order to avoid typosquatting loss. However, there is a cost associated with the registration of every site and hence without a short list of possible typosquatting candidates, most of the owners cannot afford such defensive registration. In this research, we have designed a tool named *TypoWriter* which can provide a feasibly short list of possible typosquatting domain names to make defensive registration widely possible. This is accomplished by using a prediction model, i.e., based on the to-be-registered domain name only. To the best of our knowledge, this is the first attempt to *predict* highly probable typo errors of a given URL. This is a significant challenge to predict typos at the pre-registration phase without any post-hoc information or DNS log analysis. One of the goals of our work is to create a framework for detecting typosquatting error by analyzing DNS log of *.bd* domain provided by Bangladesh Telecommunications Company Limited (BTCL). The other objective of our work is to design a predictive model that would suggest a candidate set for preventive domain registration. We developed a recurrent neural network (RNN) based model to predict possible typos of a given site. We make the following contributions in this thesis

- Designing a deterministic approach to select typosquatting error-set by analyzing DNS log of BTCL which yields development of a training dataset of pairs.
- Designing a supervised learning based model and set of appropriate features of which some were not used previously for automatic detection of typosquatting errors.
- Identifying insightful patterns and characteristics of typosquatting errors resulting from the study of detected errors. To encourage reproduction, we published the dataset and tool [7].

- Developing a Recurrent Neural Network (RNN) based prediction system which can predict possible typo candidates for a new website before registration. To the best of our knowledge, this is the first work that tries to predict as a pre-hoc manner.

### 1.3 Research Questions

In this thesis, we address the following research questions. In other studies of typosquatting, rule-based methods are applied to generate typos. However, we can get actually occurred typos in DNS log. The following research question investigates the phenomenon whether such errors can be detected automatically analyzing the DNS log.

**RQ1** *Can we detect typosquatting errors from DNS log?*

Extracting typosquatting error from the large size of DNS log requires significant effort. For this reason, a rule-based deterministic tool is needed which can be developed using the predictors found from literature. Then the quality of the outcome of this process will be verified. Therefore, the next research question is:

**RQ2** *Can we make a deterministic tool to detect typosquatting?*

The major focus of the earlier studies was to analyze the typo URLs those are generated manually by all possible character modification for preventive domain registration. Machine learning based detection technique with extracting typosquatting error from DNS log is a relatively new approach. Next question is related to the performance of the new approach.

**RQ3** *Can machine learning-based tools detect typosquatting and which classifier performs the best?*

To improve the performance of machine learning based tools and their detection qualities, various types of distance or similarity features are gathered from extracted typo pairs. Among those feature, some may contribute significantly to detection accuracy. Therefore, the next question is:

**RQ4** *Which features indicate typosquatting?*

Every day a large amount of DNS query logs are generated. Manual inspection is required for analyzing these logs to improve quality. To achieve acceptable detection quality, a certain amount of log need to be processed. How much log need to be processed to attain acceptable detection quality is the question under investigation.

**RQ5** *How long to wait for achieving acceptable detection quality from log analysis?*

If it is possible to register additional domains for defensive redirection, it would reduce the possibility of typosquatting attacks from the beginning. Hence, we would like to investigate if it is possible to predict probable typo candidates from a given domain name.

**RQ6** *Can we find out which defensive domains to register to prevent typosquatting at the inception of a new website (without log analysis)?*

## 1.4 Overview of Our Solution

First, we collected DNS log of 5 months (223.5 GB). Next, we removed the invalid entries and found 117GB of valid data. Applying some deterministic rules related to edit distance, time difference, IP address, etc. and performing a careful manual inspection, we refined the dataset. A gold set was prepared from the refined data. Next, we selected 9 features covering lexical and phonetic similarity, keyboard proximity, time distance, etc. observing the properties of gold set entities. Finally, we experimented with 7 widely used classifiers. Among them, random forest classifier performed the best with accuracy over 98% and F-score over 90% on 10-fold cross-validation and test dataset. We have also analyzed the volume of log data required to detect typosquatting with certain accuracy and found that our proposed model could achieve more than 92% accuracy with log size of 700,000 only that may be collected in a couple of days. We have developed the tool *TypoWriter* that would suggest a candidate set for preventive domain registration using Recurrent Neural Network (RNN). It was found that splitting a domain name into 3-grams and then performing the prediction over these 3-grams, we were able to achieve a true positive rate (TPR) of 40% by considering top 5 predicted candidates only.

## 1.5 Outline of the thesis

Chapter 2 mentions some significant typosquatting attacks over the last few years and discusses some of the related works with typosquatting error detection.

Chapter 3 illustrates the detail of different methods and algorithms we have used in the thesis.

Chapter 4 focuses on the experimental setups and results. It also illustrates the experimental data as well as the environment of our research and examines the experimental results. Finally, the analysis of different experimental results are presented in this chapter.

We show the empirical study of our work with identified insightful patterns and characteristics of typosquatting errors in Chapter 5.

Finally, Chapter 6 concludes our thesis. This Chapter also includes the outlines of some future works related to this dissertation.



# Chapter 2

## Literature Review

Several works have been done on the detection of typosquatting and its preventive measurements. Below we discuss some of them.

### 2.1 Research on Typosquatting

Hussain et al. [8] showed typosquatting scenario of popular domains of Bangladesh. Domains are checked if they are live or not after the generation of possible typo domains. Then the potential typo domains are content-analyzed to see if they aim at abusing users with spam, scam or any other media. Khan et al. [4] presented a strategy for quantifying typosquatting harm via intent inference technique. Intent inference allows us to define a new metric for quantifying harm to users. Szurdi et al. [9] performed a comprehensive study of typosquatting domain registrations within the.com TLD. Piyush [10] overview of cybersquatting and its prominent effects in India. Wang et al. [11] proposed Strider Typo-Patrol to discover large-scale, systematic typosquatters. They showed that a large number of typosquatting domains are active and a large percentage of them are parked with a handful of major domain parking services, which serve syndicated advertisements on these domains. They also described the Strider URL Tracer to allow website owners to systematically monitor typo-squatting domains of their sites.

Agten et al. [12] collected data about the typosquatting domains of the 500 most popular sites for seven months. Analyzing those data they revealed that 95% of the popular domains are actively targeted by typosquatters. Some trademark owners protected themselves by proactively registering their own typosquatting domains. According to their study, typosquatting domains change hands from typosquatters

to legitimate owners and vice versa. Typosquatters vary their monetization strategy by hosting different types of pages. Spaulding et al. attempted to determine the effectiveness of several typosquatting techniques in [13], [14] and [15]. Nick et al. [16] presented soundsquatting, where an attacker takes advantage of homophones, i.e., words that sound alike, and registers homophone including variants of popular domain names. They designed a tool for the automatic generation of soundsquatting domains. Using that tool, they discovered that attackers are already monetizing them in various unethical and illegal ways. Spaulding et al. [17] reviewed the landscape of domain name typosquatting, highlighting models and advanced techniques for typosquatted domain names generation, models for their monetization, and the existing literature on countermeasures. Liu et al. [18] proposed TypoPegging, which is a novel quantitative method to measure the visual similarity of two given domains. The proposed method is based on generalized Levenshtein distance. To accelerate the searching process, triangle inequality of visual distance metric and locality sensitive hashing algorithm were used.

Banarjee et al. [19] proposed SUT, a practical countermeasure based on network metrics to detect phony websites. They found that the power of SUT lies in use of the network-layer profile of phony sites, and less in perceived popularity of the site. Moore et al. [6] described a method for the identification of typosquatting. They found that 80% typosquatting was supported by pay-per-click ads and 20% include static redirection to other sites. Using regression analysis, they found that websites in categories with higher pay-per-click ad prices face more typosquatting registrations. Kidmose et al. [20] developed technique for pre-registration domain detection which can be applied to other forms of abuse as well. Existing works on both the pre and post-registration detection were focused on a few Top-Level domains (TLDs). Piredda et al. [5] addressed the problem of Typosquatting detection by leveraging a passive Domain Name System (DNS) traffic analysis. Additionally, they exploited machine learning to learn a similarity measure between domain names capable of detecting typo-squatted ones from the analyzed DNS traffic.

Typosquatting detection via smartphone has discoursed in [21]. Authors shedded light on how Advanced Persistent Threat (APT) attack through spear phishing can occur in Smartphone and how to detect it. Feature selection for the phishing website is discussed in [22]. They aimed to list and identify the important features for machine learning-based detection of phishing websites.

The major focus of the earlier studies [20] [5] was to analyze the typo URLs that are generated manually by all possible character modification for preventive domain registration. In our case, 13,674 typo error pairs were extracted from ccTLD Naming Server’s five-month log to build an automatic detection tool and then different properties were analyzed.

## 2.2 Existing Typo Generation Tools

To the best of our knowledge, there is no existing work on probabilistic prediction of URL typo and ranking among the predictions. However, a number of tools are available that generate typo variations of URL by applying a set of deterministic rules. A typo generation tool named dnstwist [23] finds similar looking domains in order to detect typosquatters, phishing attacks, fraud, and corporate espionage. Catphish [24] is another tool that generates similar looking domains for phishing attacks and categorizes domain status to evade proxy categorization. Andrew Horton developed a command line based tool urlcrazy [25] which generates domain typos and variations to detect typosquatting, URL hijacking, phishing, and corporate espionage. Web-based tool SeoChat populate typo list based on common spelling errors and keyboard proximity. Another web-based tool Typofinder [26] generates typos based on TLD swapping, bit flipping, and swapping characters with homoglyphs. All of these typo generation tools generate a large number of typosquatting domain names without providing any relative ordering among them.

## 2.3 Recurrent Neural Network

Recurrent Neural Networks (RNN) have made significant contributions in many applications related to natural language processing, speech recognition, music composition, etc. Unlike feedforward networks, they may have connections between nodes of the same layer which can act as a memory by storing representations of recent input values. RNNs can leverage their memory property to make better predictions for sequences of arbitrary length. One variation of RNN is Long short-term memory (LSTM) invented by Hochreiter and Schmidhuber [27] in 1997. LSTM has been able to achieve the best performance in multiple applications domains compared to other RNN variants. It can learn to bridge time intervals in excess of 1000 steps even in case of noisy, incompressible input sequences, without loss of short time lag capabilities.

As a result, it overcomes many of the limitations faced by Recurrent Neural Networks. Due to its proven good performance for text sequence prediction, we have used LSTM for the typosquatting detection task.

## 2.4 Summary

In this chapter, we have reviewed some of the research works done in the past decades related to our work. The purpose of the overall discussion in this chapter is to set up a baseline for the framework we are going to propose in the following chapter.

# Chapter 3

## Our Proposed Framework

The work presented in this thesis can be divided into two phases. In the first phase, we proposed a method to *detect* typosquatting errors from the DNS log. Then in the second phase, we developed a system to *predict* defensive domain names. The detailed methodology is discussed as followed.

### 3.1 Detection from Log Analysis

The *detection* task was performed following some sequential steps which are discussed below. An overview of the steps followed in this phase is depicted in Figure 3.1.

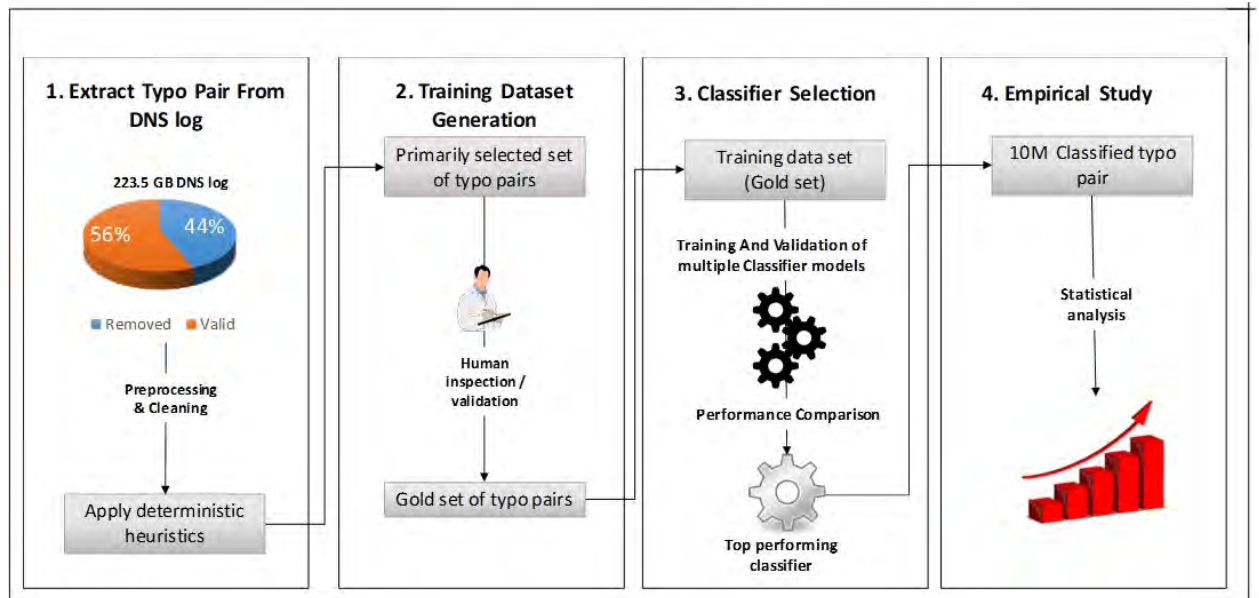


Figure 3.1: Overview of our detection scheme

### 3.1.1 Data Generation

We used a real-world DNS log in our experiments. A DNS log record is generated whenever a DNS request is issued by a client. That is, when a user browses a domain name by its URL, a DNS log record is created. Attributes created by DNS log include requesting time, source IP, destination URL, query type, etc. The size of a DNS log is huge. For example, billions of records are captured at a top-level domain server every month. It is even larger at a root domain server. The DNS log dataset we used here is captured by three root DNS servers of Bangladesh Telecommunications Company Limited between 09/01/2017 and 02/31/2018. The log files consist of over 223.5 gigabytes of raw DNS log data. We also collected all 40 thousand registered under the .bd domain and related information.

### 3.1.2 Data Cleaning

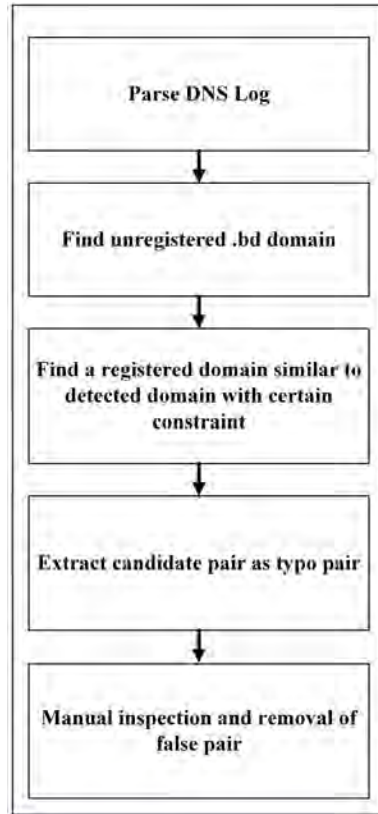
Cleaning and preprocessing are required to work with collected domain server log. All the websites in .bd domain are required to have three parts: the domain name, the top level domain (.com, .org, .net, .ac, .co, .dhaka, .edu, .gov, .info etc.) and finally .bd. We removed all the data that did not comply with this requirement considering this data as noise. We also removed the DNS server related tags which were not essential for our task. We found that 90 percent of domain names are covered by name length from 10 to 30. For qualitative study we only consider those DNS query requests having a length less or equal to 30. After cleaning, our dataset size was 117 gigabytes (reduced by almost 50%).

### 3.1.3 Extraction of Typo Pair using Deterministic Rules

The processed log is then used for generating potential typo pairs. Each pair was formed with an expected domain and typo domain name. The expected domain is the domain name a user intended to visit and typo domain is the domain name the user mistakenly typed. For example,  $\langle google.com, gogle.com \rangle$  could be a possible typo pair in this context.

#### 3.1.3.1 *Edit distance based estimation*

A naive way of extracting typo pairs is to list all the domain name pairs whose name differ by not more than a certain number of characters. The most frequent occurrences



**Figure 3.2:** Detailed approach to extract typo pairs from initially generated log

of mistyping are those that involve a one-character distance, also called the Damerau-Levenshtein (DL) distance [28] one, from the correct spelling. In this research, we focus on typosquatting domains of Damerau-Levenshtein distance one (DL-1) that are generated using the most common operations: addition, deletion, substitution of one character, transposition of neighboring characters, etc. We considered a typo pair combining two factors: distance pair and time difference. If  $n$  number of edits are required to rectify mistakenly typed domain name, then typo is at  $n$  distance from the desired domain name. In other words, their Damerau-Levenshtein distance is  $n$ . For example, let *bankasia.com.bd* be the desired domain name. Someone mistakenly typed *ankasia.com.bd*. Here, the insertion of the character 'b' gives the desired domain name. Hence, the expected domain and typo domain are at 1 edit distance.

### 3.1.3.2 Time difference

Someone may hit any unregistered domain and within a short interval s/he is likely to hit actual registered domain. It may be deduced that actually s/he wanted to visit the

2nd website but mistakenly typed 1st one. Here, the 2nd domain is the expected one and 1st one is a typo. In our experiment, we considered up to the 1-minute interval between expected domain hit and typo domain hit.

### 3.1.3.3 Same IP

To ensure typo mistake, we checked whether both the typo and expected requests came from the same IP or not. We first extracted all such typo pairs originating from the same IP address recorded within the 1-minute interval and then considered pairs having edit distance less or equal to 2 in order to keep the model simple.

## 3.1.4 Gold-set Generation

Our list of typo pairs may contain some false typo pairs which have fulfilled required criteria coincidentally but actually, they are not typosquatting error. Hence, we need to refine the typo pair list. We manually selected some pairs as a true typo and rejected others from the initially prepared typo list. Refined typo pair list is referred to as gold set in our experiment. Gold Set consists of five thousand typo error pairs with 60% true typo pair and 40% randomly selected false typo pairs.

## 3.1.5 Feature Selection

The performance of a supervised machine learning method depends on the combination of the selected features. We explored the following features to identify typo.

**Table 3.1:** Features used for typosquatting detection

Features
Visual Similarity
Edit distance
keyboard Similarity
Bigram Similarity
Phonetic Similarity (Double Metaphone)
Typo length
Registered domain Length
Phonetic Similarity (Soundex)
Time Interval

Finally, 9 features as presented in Table [3.1](#) were selected as contributing significantly in our context.



1. Lexical similarity: These features include both character-based and word-based features. These features represent edit distance of typo pairs and number of characters misspelled by the user. The feature set includes total characters, special character usage, and several word-level features such as total words, characters per word, the frequency of large words, how many bigrams are common in typo pair, etc.
2. Phonetic similarity: Phonetic similarity feature includes words that are pronounced with a similar sound but are spelled differently. By using a homophone dataset [16] we can use phonetic similarity feature of typo pairs.
3. Keyboard proximity distance: We considered keyboard proximity distance of modified or substitute character. For example, ‘typo’ and ‘tylo’ are really close (p and l are physically close on the keyboard), while ‘typo’ and ‘tyqo’ are far apart.
4. Visual similarity: A domain is doubtful if it looks very similar to some popular websites and not registered by their owners. Visual similarity distance can be used as a typo detection feature. For example, ‘sbohoj’ and ‘shohoj’ are visually close (b and h are close).
5. Time distance: The conditional probability of visiting the target domain will be much higher for typosquatting domains than for unrelated domains which just happen to have a small edit distance between them. The time distance between the visit of lexically similar typo and registered domain can be used as features.

### 3.1.6 Classifier Design

We considered the *detection* problem as a binary classification problem. (where the two classes are benign and typosquatting domains) using 9 selected features mentioned in Table 3.1. We represented each typo pair as  $(\vec{x}, y)$  where  $\vec{x} \in \mathbb{R}^n$  is a vector of  $n$  features and  $y \in \{true, false\}$  is the detection result of typo error. In our study,  $n$  is set to 9 for typo features. On the training dataset we applied 10 fold cross validation technique to experiment with different machine learning classification algorithms such as - Naive Bayes, decision tree based ID3 [29], artificial neural network [30] with back-propagation, random forest [31], AdaBoost [32], k nearest neighbor (k-NN) [33], and support vector machines (SVM) [34]. The detection result is presented in section 4.3.

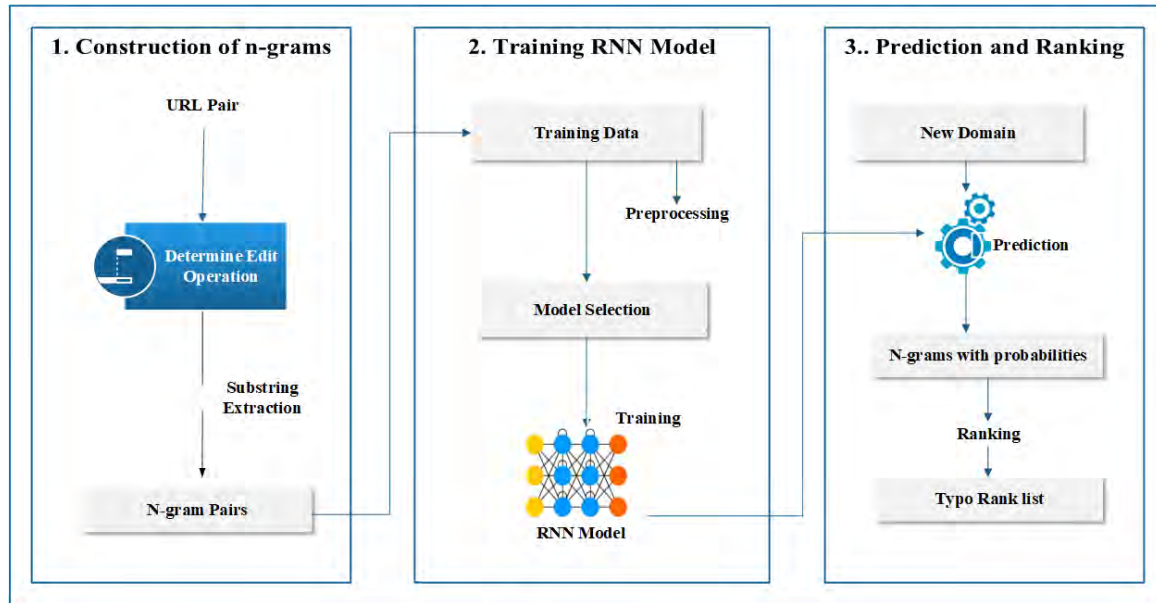
## 3.2 Pre-registration Prediction

The harmful practice of Typosquatting involves deliberately registering Internet domain names containing typographical errors that primarily target popular domain names. The result of our experiment indicates that visual similarity, phonetic similarity, and keyboard proximity are associated with typosquatting error. Our predictive typo registration model helps a new website owner to prevent typosquatting attack by defensive registration of most probable typo domains. List of predictive probable typo domains with rank helps a new website owner to make a better decision about how many domains he or she will register to protect his or her brand. *TypoWriter* uses a prediction model which is trained using a supervised learning method. Supervised learning requires labeled data for training. Therefore, we trained our model with a dataset containing a number of registered domain names and typo variations of those domain names occurring in real life. However, this type of dataset is not publicly available. Hence, we prepared such a dataset analyzing DNS log. The dataset preparation was conducted following three major steps. First, we collected a large amount of DNS log and preprocessed it. Second, we detected typo mistakes for URL made by users. Finally, we constructed a dataset containing domain name pairs where one domain name is the typo variation of another.

We constructed a subset of our formerly available URL pair set by removing the entries that contained typo errors which were not part of the domain name. As a result, the constructed typo pairs in which the mistake is only in the domain name were selected. Hence, typo pair such as (*google.com.bd*, *google.co.bd*) where the typo error is not within the domain name part *google*, were discarded. After that, we extracted only the domain name part from both the registered domain name and typo domain name. In this way, we obtained a total of 3,745 pairs from the DNS log. Later, we divided the dataset into two parts for training and testing purposes, respectively. The test dataset consisted of the top ten domain names having the most number of unique typo errors in the dataset. The rest were included in the training dataset. Here we present a detailed description of the design principle and methodology used for *TypoWriter*.

### 3.2.1 The Concept

Unlike existing typo generation tools, *TypoWriter* does not generate an exhaustive list of typo domains based on deterministic rules. Rather, it was designed to reflect



**Figure 3.3:** Methodology followed for the construction of *TypoWriter*

user behavior while making such predictions. Moreover, the predictions made by *TypoWriter* are ranked in an order of probability of their occurrences. The task of predicting possible typos for a domain name has been considered as a sequence to sequence prediction problem in this work. Given a sequence of characters  $s$  as original domain name, *TypoWriter* predicts an ordered list of such sequences  $t_1, t_2, t_3, \dots, t_k$  each of which is a probable typo variation of  $s$ . The ordering of the sequences is done based on their probability of occurrence. Feedforward neural networks are widely used in prediction problems where the outcome against an input is labeled with a single value. In contrast, RNNs are better suited for the prediction of sequences with temporal dependency among the elements. From our constructed dataset, it was observed that typo made by a user while typing a domain name depends not only on the value of the character but also on the sequence of characters it is a part of. For example, it was observed that *telitalk* was a common typo error for the domain name *teletalk*. It occurred due to the replacement of the fourth character ‘e’ by ‘i’. On the other hand, *tiletalk*, which can be constructed by replacing the second character ‘e’ with ‘i’, was not present in the data at all. Owing to the dependency of an error on the relative location of a character in a sequence, an RNN model was chosen over a feedforward neural network. Some key issues faced in this regard and the approaches adopted to address them are discussed below.

### 3.2.1.1 Variable Length of Sequences

The length of the domain names existing in our dataset ranged from 5 to 57. Similarly, the length of output sequences, i.e., the typo domain names also varied from 4 to 58. Therefore, determining the input and output size of the model was a challenge. One way of addressing this issue is to consider the maximum length of input sequences as input size and the maximum length of output sequences as the output size of the model. Sequences with shorter length can be padded with a predefined dummy value to make the length equal to the maximum length. However, experimentation with this approach produced a poor result due to a relatively large variation in the lengths and unavailability of enough data to account for that variation. For instance, output sequences with length 4 were padded to make the length 58 which made the trained model biased to produce the padded value more frequently. Hence, making the sequences equal in length by padding was avoided. On the contrary, a scheme of splitting both the input and output sequences into n-grams was adopted.

### 3.2.1.2 Global vs Local Dependency

The occurrence of typo depends both on the character in question and the sequence of characters it is a part of. Even so, the dependency was found not to be extended over the entire sequence. Rather, a local dependency was observed. For example, a common type of the domain name *xiclassadmission* was *xiclassadmisson*, mistakenly removing ‘i’ before the character ‘o’. This mistake is presumably due to phonetic similarity between the two sequences if not just a random typo error. Therefore, the omission of this ‘i’ is influenced by the character ‘o’ following it. However, it is evident that this omission is not influenced by the first character of the sequence ‘x’. Accordingly, the dependency of typo errors on nearby characters is local in contrary to a global dependency on the entire domain name. However, the exact span of this local influence could not be determined by manual observation and was found to be varied in sequences.

In order to address the discussed issues, we constructed n-grams from the domain names for training our model. Different steps followed in the prediction and ranking scheme are discussed below.

### 3.2.2 Construction of n-grams

The prepared training dataset contains the registered domain name and the corresponding typo domain name pairs. In all such pairs, the registered domain name can be transformed into the corresponding typo domain name by performing a single edit operation: insert, delete, or replace. Instead of using the entire registered domain name as input and typo domain name as output to the RNN model, we constructed all possible n-gram pairs from the sequences that contain the character involved in such an edit operation and then used those as input and output. For any pair  $(reg\_domain, typo\_domain)$  in the dataset, we identified the edit operation required to convert  $reg\_domain$  into  $typo\_domain$  as well as the index  $i$  in  $reg\_domain$  where such an operation is required. We also defined two special markers *PAD* and *BLANK* to be used in specific cases. The technique of n-gram construction varies for each edit operation. Algorithm 3.1 shows the steps for n-gram construction from a registered domain name and typo domain name pair.

---

**Algorithm 3.1** Algorithms for constructing n-gram

---

**Input:** Registered Domain name  $reg\_domain$ , Typo domain name  $typo\_domain$ ,  $n$

**Output:** A set of string pairs  $L$

```

1:  $op \leftarrow$  find edit operation required to convert  $reg\_domain$  into  $typo\_domain$ 
2:  $idx \leftarrow$  find character index of  $reg\_domain$  where edit operation is required
3:  $l \leftarrow$  length of  $reg\_domain$ 
4:  $L \leftarrow \phi$ 
5: if  $op$  is INSERT then
6:   for  $i = \max(idx - n + 1, 0)$  to  $\min(idx, l - n)$  do
7:      $in \leftarrow$  Substring of length  $n$  starting from index  $i$  in  $reg\_domain$ 
8:      $out \leftarrow$  Substring of length  $(n + 1)$  starting from index  $i$  in  $typo\_domain$ 
9:      $L \leftarrow L \cup \{(in, out)\}$ 
10:  end for
11: else
12:   if  $op$  is DELETE then
13:     insert special marker BLANK at index  $idx$  of  $typo\_domain$ 
14:   end if
15:   for  $i = \max(idx - n + 1, 0)$  to  $\min(idx, l - n)$  do
16:      $in \leftarrow$  Substring of length  $n$  starting from index  $i$  in  $reg\_domain$ 
17:      $out \leftarrow$  Substring of length  $n$  starting from index  $i$  in  $typo\_domain$ 
18:      $L \leftarrow L \cup \{(in, out)\}$ 
19:   end for
20: end if
21: return  $L$ 

```

---

**Table 3.2:** N-gram pairs constructed from some sample data

Edit Operation	Registered Domain Name	Typo Domain Name	Constructed Pairs
Insert	everjobs	everyjobs	(ver, very)
			(erj, eryj)
			(rjo, ryjo)
			(job, yjob)
Delete	bdfinance	bdfinace	(nan, na#\$)
			(anc, a#c\$)
			(nce, #ce\$)
Replace	register	regester	(egi, ege\$)
			(gis, ges\$)
			(ist, est\$)

### 3.2.3 Training RNN Model

The heart of our entire prediction scheme is an RNN based sequence to the sequence prediction model. The model was trained with the n-grams obtained from registered domain names as input and the corresponding substrings obtained from typo domain name as output. We experimented with different architectures and hyperparameter values for the model and chose the one with the best performance.

#### 3.2.3.1 Preprocessing of Training Dataset

At first, we constructed pairs in the form of  $(inputsequence, outputsequence)$ . The length of each input sequence was  $n$  since it was actually an n-gram obtained from a registered domain name. However, the length of the output sequence could be different based on the edit operation required to construct the corresponding typo domain name. If the typo domain name was the result of a replace or delete operation, the length of the output sequence was the same as that of the input sequence i.e.  $n$ . On the other hand, output sequences generated by insert operations had a length equal to  $n + 1$ . In order to make all the output sequences of equal length, we used padding. All the output sequences with length  $n$  were padded with the special marker *PAD* to make the length  $n + 1$ . Consequently, the uniformity in length between both input and output sequences could be achieved which helped to determine the architecture of the RNN model. The input and output sequences obtained for  $n = 3$  from some sample domain name pairs are demonstrated in Table 3.2. The sequences were then encoded by applying one hot encoding. In order to do the encoding, we first created

a vocabulary set consisting of the unique characters and special markers present in the sequences. In total, the vocabulary set contained 40 elements: 26 letters from the English alphabet, 10 digits, the two characters dot (.) and dash (-), and the two special markers *BLANK* and *PAD*. Each element of the vocabulary set was assigned a unique index value within 0 and 39. After performing one hot encoding, each character in the sequences was represented by a binary array of length 40 i.e. equal to the size of the vocabulary set. This array contained zero in all positions except only at the index of the character in the vocabulary set. The sequence of arrays obtained after such encoding together constructed the training dataset for our prediction model. We constructed four such datasets for values of  $n$  in  $\{2, 3, 4, 5\}$  to train four different prediction models.

### 3.2.3.2 Model Architecture

We experimented with four different RNN models corresponding to different values of  $n$ . The model yielding best result on the test data was selected for prediction purpose. The architecture of each model was different as the size of input and output to each model varies with the value of  $n$ . The number of nodes in the input layer for each model was  $n$  while that in the output layer was  $n + 1$ . We performed a grid search to determine the number of hidden LSTM layers within  $\{1, 2, 3, 4\}$ . The model with four hidden layers suffered from overfitting and hence we did not increase the number of layers any further. Similarly, the number of nodes in each hidden layer was also determined experimentally by performing a grid search within  $[30,50]$ . Each model was trained with 80% of the corresponding training dataset while the remaining 20% was used for cross-validation of the model. Each model was trained for 200 epochs using categorical cross-entropy loss function and adam optimizer. The model construction and training were performed using deep learning framework Keras [35]. The model parameter values with the lowest validation loss in each case as shown in Table 3.3 were subsequently used for testing.

### 3.2.4 Prediction

In order to predict a possible typo variation of a given domain name, we first constructed all possible  $n$ -grams from the domain name. Then each  $n$ -gram was converted into a one hot vector sequence which could be given as input to the trained RNN model. Given the input, the trained RNN model would generate  $(n + 1)$  output

**Table 3.3:** Model architectures with the minimum validation loss for different values of  $n$ 

Value of $n$	Number of Hidden Layers	Nodes in Hidden Layers
2	2	35, 36
3	2	38, 40
4	2	43, 43
5	2	45, 49

arrays of length 40. The  $k$ -th array contained the probabilities of all the characters in the vocabulary set to be the  $k$ -th character of the output sequence. Therefore, taking one character from each array, an output sequence of  $(n + 1)$  characters can be obtained. The probability of such an output sequence could be calculated by multiplying the corresponding probability of each of the characters selected. In this way, for a given  $n$ -gram,  $40^{(n+1)}$  output sequences with an associated probability can be generated. If the predicted output sequence contained the special markers *PAD* or *BLANK*, those markers were removed from the sequence. Replacing the input  $n$ -gram in the given domain name with the predicted output sequence would provide us a typo variation of the domain name. The probability of occurrence of such a typo variation was considered to be the same as the corresponding output sequence for the replaced  $n$ -gram.

### 3.2.5 Ranking

To decide the order of candidate typo domain names, we used a probabilistic ranking scheme. Since each of the typo domain names generated by our prediction model is associated with a probability of occurrence, ordering can be done based on this probability. The typo domain name with the highest probability is assigned the highest rank and expected to be the first choice while making defensive registration to prevent typosquatting. Similarly, all the typo domain names can be ranked in the decreasing order of their probabilities to obtain an exhaustive rank list. One downside of this exhaustive ordering is that a large number of possible typo variations can be generated from our prediction model. From a given domain name of length  $l$ , a number of  $(l - n + 1)$   $n$ -grams can be constructed which altogether would yield a total of  $(l - n + 1) * 40^{(n+1)}$  typo variations. This number is often impractical even for small values of  $l$  and  $n$ . Hence, we define a parameter  $n_r$ , which denotes the



number of most probable typo variations of a domain name that is required and reduce our search space accordingly. We used beam search with a width equal to  $n_r$ . In this searching process, we picked top  $\min(n_r, 40)$  characters from each of the  $(n+1)$  output arrays for a particular n-gram input. As a result, the number of output sequences obtained became  $n_r^{(n+1)}$ . From these output sequences, we kept the top  $n_r$  and discarded the rest. In this way, for a given domain name, the total number of typo domains generated was  $(l - n + 1) * n_r$  which is linear in terms of both  $n$  and  $n_r$ . These typo domain names were then sorted in descending order of their probabilities of occurrence and the top  $n_r$  of them were selected in the rank list.

### 3.3 Summary

In this chapter, we discuss the detailed methodology of our work. So, we presented the detection and prediction model we developed.

# Chapter 4

## Result and Performance Evaluation

We implement our proposed algorithms, executed them on our DNS log and perform the manual inspection to prepare gold dataset. By using this gold dataset, we have developed the proposed prediction model with ranking Schema. This chapter contains the detail of the dataset, experimental settings, results, and analysis of our thesis work. In Section 4.1, we explain our datasets. Afterwards, in Section 4.2, we explain our experimental environment and settings for our implementation. Rest of this chapter contains the experimental results and analysis of the results. In result section, first we will present the results obtained in both *detection* and *prediction* tasks. After that, we will discuss our findings based on the research questions introduced in Section 1.3.

### 4.1 Data Collection and Dataset

This research work is done using the data of registered .bd web domain of Bangladesh. Our datasets are DNS data obtained from BTCL, a major telecom operator of Bangladesh. A DNS resolution request is made whenever a user tries to browse a website by its URL. For every such request, a DNS log is generated. We collected a chunk of DNS log captured by Bangladesh Telecommunications Company Limited (BTCL) [36] from September 2017 to February 2018. The log consisted of records for all requests made to visit .bd domains over a period of six months. To retain privacy, source IP addresses originally recorded in the DNS log were replaced by a corresponding hash value. As a result, it was possible to identify if two requests were made from the same source IP without revealing the actual IP address. Besides source IP, the DNS log also contained the time-stamp of each request, destination URL, query type, and some other tags related to the DNS server. The total size of the raw log was 223.5 gigabytes.

## 4.2 Experimental Setup

In our thesis work, the proposed supervised machine learning model for processing the DNS data for identifying the typo mistake activities and error patterns. According to the framework of our model, the collected log was first cleaned to retain only the information that is relevant to our work. All the websites in *.bd* domain are required to have three parts: the domain name, the top level domain (.com, .org, .net, .ac, .edu, etc.) and finally *.bd*. It was observed that the data contained a significant number of requests that do not comply with this requirement. We removed all such records from the log. We also removed the DNS server related tags that were not essential for our work. Finally, we got a list of rows where each row consisted of the time stamp of visit, hashed source IP address, and the URL of the requested website. After cleaning, the size of the dataset became 117 gigabytes (reduced by almost 50%).

The raw DNS data are contained in simple but very large text files. For processing this data using our algorithm, we developed a program based on our proposed detection classifiers and clustering algorithms in the layers of our hierarchical framework using JAVA. We also utilize the WEKA data mining tool to implement all these classifier and clustering techniques in JAVA.

For RNN predictive model construction and training, we use deep learning framework Keras [35]. Keras is an open source neural network library written in Python running on top of TensorFlow [37], Microsoft Cognitive Toolkit and Theano. A Python program was developed to process detected typo pairs for constructing n-gram. Those n-grams were used in training predictive RNN based model.

The experiments on Big Data like DNS log is resource demanding and time-consuming. So, all the experimental implementations of this thesis are done on a number of personal computer parallelly equipped with Intel Core i7 CPUs running at 2.5 GHz or more and equipped with 4 to 8 GB RAM.

## 4.3 Detection Performance

First of all, we separated our log data month-wise and prepared 5 groups of dataset. We used data of 4 months for training and validation of the machine learning model. We experimented with the 7 classifiers varying their hyper-parameters to maximize the performance results of the 10-fold cross-validation. Finally, we calculated the average result of 10 fold data which is shown in Table 4.2. Next, to check if there is any

possibility of over-fitting, we applied the model on a test set, i.e. DNS requests in February 2018. The parameter values given in Table 4.1 provide the highest performance for each classifier. The training and testing were performed using the WEKA tool [38]. All classifiers used in our experiment belong to different classes to ensure wide coverage of different possible classifiers.

**Table 4.1:** Parameters used for detection classifiers

Classifier	Parameter Name	Value
Naive Bayes	No Parameters	NULL
Decision Tree(ID3)	No Parameters	NULL
Random Forest	Size Per Bag	100
	Number of Iterations	200
	Number of Trees	200
AdaBoost	Classifier Used	J48
	Number of Iterations	100
Support Vector Machine	Type of SVM	C-SVM, $C = 1$
	Kernel Function	$\exp(-\gamma \ u - v\ ^2)$
	Class Weights	1, 1
Neural Network	Learning Rate	.05
	Maximum Epochs	2000
	Number of Hidden Layers	4
	Number of Nodes in Hidden Layer	10, 10, 6, 10
k Nearest Neighbor	Number of Neighbors%	6

We trained seven different classifiers for the detection task with 10-fold cross-validation. Performance comparison of those different machine learning models on the cross-validation data is shown in Table 4.2. We see that the random forest classifier outperforms other classifiers in terms of accuracy, precision and F-score. Next, we have experimented on a test dataset which is of February, 2018 with a size of 20GB. It is to be noted that in training data we used log upto January, 2018.

**Table 4.2:** 10-fold cross validation performance measures using different machine learning models for detection task

Classifier	Accuracy	Precision	Recall	F-score
Naive Bayes	89.14%	75.24%	74.47%	74.85%
Decision Tree	98.20%	95.65%	95.96%	95.80%
Random Forest	98.31%	96.26%	95.88%	96.07%
AdaBoost	95.05%	85.48%	90.92%	88.12%
SVM	92.35%	73.18%	89.30%	80.44%
Neural Network	96.50%	88.44%	94.94%	91.57%
k Nearest Neighbor	97.39%	94.10%	93.78%	93.94%

**Table 4.3:** Comparison of different machine learning models on test dataset

Classifier	Accuracy	Precision	Recall	F-score
Naive Bayes	87.32%	75.85%	95.64%	84.60%
Decision Tree	98.70%	98.48%	98.68%	98.58%
Random Forest	98.82%	98.55%	98.89%	98.72%
AdaBoost	97.27%	95.71%	98.29%	96.98%
SVM	86.40%	73.56%	95.85%	83.24%
Neural Network	98.63%	97.44%	99.58%	98.50%
k Nearest Neighbor	97.43%	95.99%	98.37%	97.16%

## 4.4 Prediction Performance

*TypoWriter* generates an ordered list of top  $n_r$  probable typo variations for a given domain name. The ideal way of evaluating the performance of a tool like this would be to make predictions for a newly registered domain name and then comparing the occurrence of typo for that domain over a period of time. We simulated this idea by separating ten domain names from our dataset for performance evaluation. These domains were not used for training purpose and therefore can be considered as newly registered domain names to *TypoWriter*. We then compared the predictions made by *TypoWriter* with the ground-truths obtained from DNS log.

### 4.4.1 Impact of Substring Length

The prediction task in *TypoWriter* is performed by an RNN model which was trained using n-grams extracted from domain names. To determine the optimal value for

$n$ , we trained a prediction model for each value of  $n$  in  $\{2, 3, 4, 5\}$ . We tested the four trained models with the test dataset and compared the prediction performance achieved by each model. For this comparison purpose, we considered a fixed value of  $n_r=5$  which implies each model would predict the top 5 probable typo errors for each domain name in the test dataset. We then counted the number of matches with the ground-truth available for those domain names. Table 4.4 shows the number of such matches for different values of  $n$ . The actual predictions made by each model along with the ground-truths for each of the ten test domain names are shown in table 4.6 and 4.7. The highest average number of matches was obtained for  $n=3$ . Therefore, we considered this value as the most appropriate one and performed further performance evaluations using the model trained with  $n = 3$ .

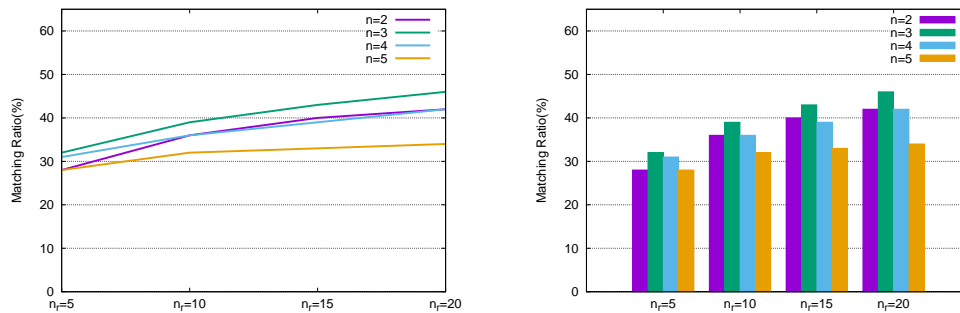
**Table 4.4:** Number of correctly detected typo domain name out of top 5 most occurred typo sites for 10 different domain names

Domain Name	$n = 2$	$n = 3$	$n = 4$	$n = 5$
teletalk	3	4	3	3
amway	0	1	0	0
omronhealthcare	0	1	1	1
educationboardresults	2	2	2	0
dhakaeducationboard	1	1	2	2
mybank	1	2	2	1
google	3	3	2	1
tigercricket	0	0	0	0
daraz	2	3	3	3
xiclassadmission	2	3	2	3
<b>Average</b>	<b>1.4</b>	<b>2.0</b>	<b>1.7</b>	<b>1.4</b>

Let us consider an example of url daraz.com. In our test dataset, it has 31 unique typos of total frequency 175. People mistyped this url for 175 times in 31 ways. We applied our predictive method for typo generation, matched with our ground truth data. We varied number of typo generation ( $n_r$ ) and  $n$  of  $n$ -gram RNN technique. When we generated 5 unique typos ( $n_r = 5$ ) with substring length  $n=2$ , 106 typos of 4 unique types matched with our ground truth data. When we generated 10 unique typos ( $n_r = 10$ ) with substring length  $n=2$ , 116 typos of 7 unique types matched with our ground truth data. If we generate more numbers of typos, chance of matching with ground truth increases. Then we calculated the ratio between matching with ground truth and total frequency of typo in test dataset.

**Table 4.5:** Matching ratios for 10 different domain names

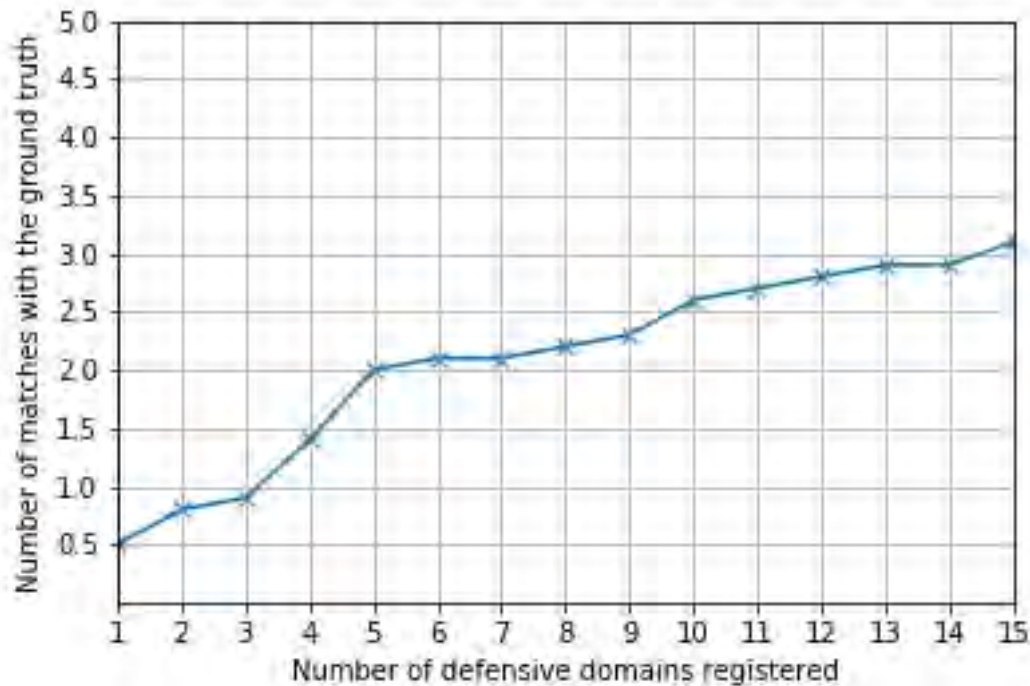
	n=2	n=3	n=4	n=5
$n_r = 5$	28%	32%	31%	28%
$n_r = 10$	36%	39%	36%	32%
$n_r = 15$	40%	43%	39%	33%
$n_r = 20$	42%	46%	42%	34%

**Figure 4.1:** Matching ratios for 10 different domain names

We selected 10 URL addresses which had more than 20 unique typos in our dataset. Then we conducted our experiment varying  $n_r$  from 5 to 20 ( $n_r = 5, 10, 15, 20$  etc) and  $n$  from 2 to 5 ( $n = 2, 3, 4, 5$  etc). We calculated matching ratios for all combinations and showed on Table 4.5 and Figure 4.1. From our experiment, we have seen maximum matching ratio was for  $n=3$ .

#### 4.4.2 Impact of $n_r$

We found that for the model trained with 3-gram data set if we take top 5 predicted typo domain names, there is on average 2 domain names in common with the ground-truth. This performance was obtained for a fixed value of  $n_r=5$ . However, the number of matches is likely to increase if  $n_r$  is increased, i.e., the number of domains registered is increased. We experimented with different values of  $n_r$  ranging from 1 to 15 and observed the performance produced by *TypoWriter*. Figure 4.2 shows the results obtained for different values of  $n_r$ . The average number of matches with the ground-truth was found to be monotonically increasing with an increase of  $n_r$ . However, the rate of this increase was not uniform. Registering 5 top defensive domain names generated by our prediction model seems to be a cost-effective choice as the rate in increase of the actual occurrence of these candidates in the ground-truth slows down



**Figure 4.2:** The increase of expected number of top typo sites with the increase of number of defensive domain registered

significantly after this point.

### 4.4.3 Comparison with Generation Tools

We compared the prediction performance of *TypoWriter* with the five tools discussed in Section 2.2, namely, dnstwist, catphish, urlcrazy, typofinder, and SeoChat for this performance comparison. However, none of these tools provide any ranking among the predictions made by them. We selected the top five domain names having the most number of unique typo occurrences from our dataset. The typo domains available for each of these domains were considered as the ground-truth for that domain. It is to be noted that the prediction model used by *TypoWriter* was not trained with these domain names and hence they are unobserved data to *TypoWriter*. We executed each tool by giving a domain name as input and collected the output. The number of output domains generated by each of the tools was different due to the difference in their generation methodologies. We also generated a list of typo domain names predicted by *TypoWriter* which were associated with a probability higher than 0.01.



**Table 4.6:** Typo candidates generated by prediction models for  $n = 2, 3$ .

Domain Name	Ground Truth	$n = 2$	$n = 3$
teletalk	taletalk	teleralk	taketalk
	teketalk	taketalk	teletak
	telitalk	teletak	telitalk
	tetetalk	taletalk	taletalk
	teletak	teketalk	teketalk
amway	ameay	ammway	amaay
	alway	am9ay	am9ay
	amuay	amwyy	amyay
	pmway	amgay	amgay
	amay	mway	amay
omronhealthcare	omronhealthare	omronhealthcar	omronhealthcare
	oimronhealthcare	omronhealtcare	omronhealttcare
	omrenhealthcare	omronhealthacare	omronhealthare
	omronhealthcafe	omronhealthcre	omronealthcare
	lmronhealthcare	omronealthcare	omronhealthhare
educationboardresults	educationboardresult	educationbordresults	educationbordresults
	educationbordresults	edcuationboardresults	educationboardresult
	educationeboardresults	educatonboardresults	edcuationboardresults
	educationboardresuits	educationboardresult	edicationboardresults
	eduationboardresults	educationboardrsults	educationboardresslts
dhakaeducationboard	dhakaeducationbord	dhakaedcucationboard	dhakaedicationboard
	dhakeducationboard	dgakaeducationboard	dhakaedcationboard
	dhakaeductionboard	dakaeducationboard	dhkaeducationboard
	dhakaeducationboars	dhakaeducationbord	dhakaeducationbord
	dhakaeducatioboard	dhkaeducationboard	dhaaeducationboard
mybank	xybank	mybonk	mvbank
	mwbank	mwbank	mwbank
	oybank	qybank	xybank
	uybank	mybang	mybenk
	vybank	mybank	mybank
google	goocgle	googe	goocgle
	googgle	gooogle	googlee
	googlr	googgle	googge
	googleo	googke	google
	gooogle	goocgle	googgle
tigercricket	tigercricef	tigrcricket	tigercricket
	tigeercricket	tifercricket	tigercricer
	tigercricet	tigarcricket	tigercricrt
	tigercicket	tigercrcket	tigercricet
	tigbercricket	tigercricrt	tigercricrt

**Table 4.7:** Typo candidates generated by prediction models for  $n = 4, 5$ .

Domain Name	Ground Truth	$n = 4$	$n = 5$
teletalk	taletalk	teletall	taketalk
	teketalk	taketalk	teltalk
	telitalk	telitalk	telitalk
	tetetalk	taletalk	taletalk
	teletak	teketalk	teketalk
amway	ameay	amwey	ammay
	alway	amaay	amyay
	amuay	amwiy	amwy
	pmway	amwqy	amaay
	amay	amwly	amwyy
omronhealthcare	omronhealthare	omronhealthcarr	omronhealthcar
	oimronhealthcare	omronhalthcare	omronhealkhhare
	omrenhealthcare	omronhealthare	omronhealthare
	omronhealthcafe	omronhealthccre	omronhealthhare
	lmronhealthcare	omronhealthhare	omronhaalthcare
educationboardresults	educationboardresult	educationbordresults	educationboardresul
	educationbordresults	educationboardresult	educationbodresults
	educationeboardresults	educationboardresuuts	educationbrdresults
	educationboardresuits	educationboardoesults	educationboresults
	eduationboardresults	educationboardersults	educatiboardresults
dhakaeducationboard	dhakaeducationbord	dhakaedicationboard	dhakaeducationbrd
	dhakededucationboard	dhakaeducationbord	dhakaeducationbord
	dhakaeductionboard	dhakededucationboard	dhakededucationboard
	dhakaeducationboars	dhajaeducationboard	dhakaeduationboard
	dhakaeducatioboard	dhakacdicationboard	dhakaeducatiinboard
mybank	xybank	myb3nk	wybank
	mwbank	cybank	cybank
	oybank	mwbank	fybank
	uybank	xybank	mwbank
	vybank	fybank	mjbank
google	goocgle	goocgle	googge
	googgle	googll	googll
	googlr	googgle	googlee
	googleo	googlle	gocoge
	gooogle	goocle	googgle
tigercricket	tigercricef	tigerrcicket	tigercricer
	tigeercricket	tigetcricket	tigercrcket
	tigercrielet	tigercrickee	tigerrccket
	tigercicket	tigrrcricket	tigercrckket
	tigbercricket	tigercrickt	tigercrickat

Then we calculated the number of matches between ground-truth and the output produced by *TypoWriter* as well as the generation tools under consideration. In order to demonstrate a quantitative comparison, we define two performance metrics  $r_g$  and  $r_t$  where,

$$r_g = \frac{\text{Number of matches with ground-truth}}{\text{Number of output domains generated by tool}}$$

$$r_t = \frac{\text{Number of matches with ground-truth}}{\text{Total number of output domains in ground-truth}}$$

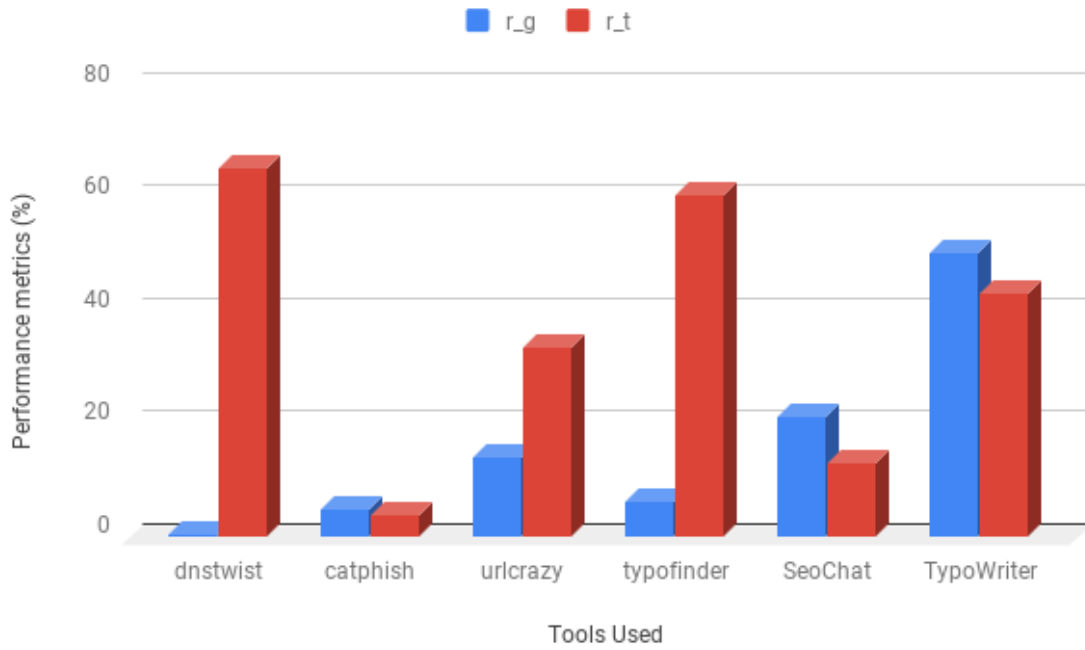
The performance comparison of *TypoWriter* with other generation tools in terms of  $r_g$  and  $r_t$  is demonstrated in Figure 4.3. Both dnstwist and typosfinder generate a large number of typo domain names. Consequently, they achieved the highest number of matches with the ground-truth and hence a high  $r_t$  score. Nevertheless, a high number of typo generations also resulted in a low  $r_g$  score for them. Theoretically, it is possible to obtain a perfect, i.e., 100%  $r_t$  score by generating an exhaustive list of all possible typo variations of a given domain name. However, since one of our goals is to provide a reasonably short list of typo variations to facilitate defensive registration, we aimed at achieving better performance in terms of  $r_g$  while maintaining a reasonable  $r_t$  score. The  $r_g$  score obtained by *TypoWriter* (50.30%) is the highest among all the tools and 136% better than the second-best score obtained by SeoChat (21.31%). Also, *TypoWriter* achieved an  $r_t$  score of 43.37%.

## 4.5 Discussion

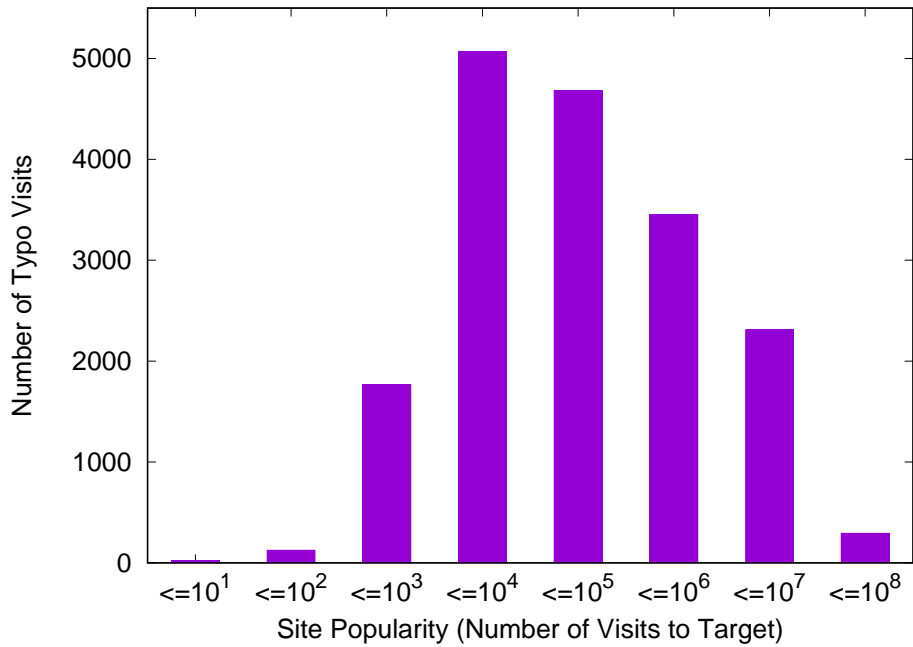
Based on the outcome of the experiments, we now discuss answers to the research questions.

**RQ1** *Can we detect typosquatting errors from DNS log?*

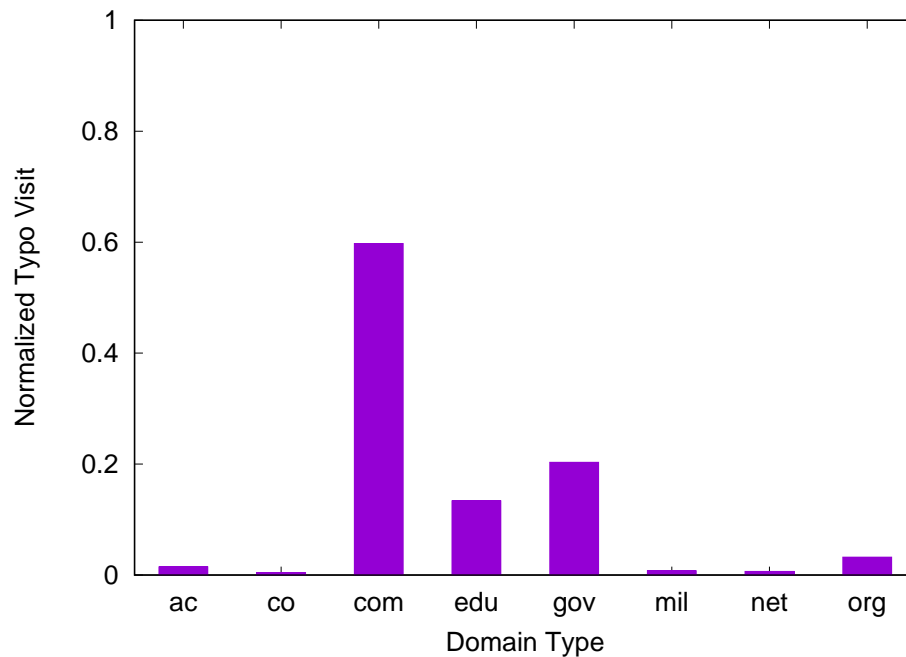
It has been seen from our experiment that the best performing classifier (Random Forest) trained on sets of DNS queries can detect typosquatting error with 98.31% validation accuracy and 98.82% test accuracy shown in Tables 4.2 and 4.3. This witness that using machine learning based model is successful to find malicious typosquatting domains.



**Figure 4.3:** Performance comparison of *TypoWriter* with other typo generation tools.



**Figure 4.4:** Number of typo visits by target site's popularity



**Figure 4.5:** Number of typo visits by target site’s category

It is seen from our experiment that most popular sites are less prone to typosquatting risk. Medium popular sites are heavily typo-squatted. Popularity is measured according to the hit count of that site. It is shown in Figure 4.4 We have categorized sub-domain of our data set and observed that .com sub-domain is highly typo-squatted which is shown in Figure 4.5

**RQ2** *Can we make a deterministic tool to detect typosquatting?*

It is possible to develop a deterministic tool from literature [4]. We found some insightful patterns analyzing data and made some rules. Using these rules we made a deterministic tool. The tool was applied to unlabeled DNS log data to identify other pairs. Then manual inspection confirmed that 70% of these are mistakenly considered typo pairs.

**RQ3** *Can machine learning-based tools detect typosquatting and which classifier performs the best?*

It has been seen from our experimental results that machine learning tools can detect typosquatting with high accuracy and f-score. We applied several machine learning algorithms on our dataset. It was seen that Random Forest outperformed other algorithms. Comparative analysis among different machine learning algorithm is shown in

Table 4.2. The results suggest the traditional methods (i.e., Naive Bayes and SVM) did not perform well.

**Table 4.8:** The impact of DNS log volume on detection accuracy

Volume of Log	Accuracy	Precession	Recall	F-score
100K	84.14%	55.34%	75.66%	63.93%
200K	86.69%	65.05%	78.82%	71.28%
300K	86.77%	66.99%	77.82%	72.00%
400K	91.04%	74.43%	88.46%	80.84%
500K	91.13%	70.87%	92.41%	80.22%
600K	91.87%	72.49%	94.12%	81.90%
700K	92.36%	72.17%	96.96%	82.75%

**Table 4.9:** Comparison among proposed methods with passive detection methodology and n-gram based method

Detection Method	Accuracy	Precision	Recall	F-score
Proposed	98.44%	98.55%	98.17%	98.36%
Passive Detection	59.35%	100.0%	22.66%	36.94%
N-gram Based	70.07%	53.09%	76.66%	62.73%

We thus believe that our approach may be particularly useful to improve the aforementioned existing systems aimed to detect malicious domains while passively monitoring the DNS traffic [5] [4]. Our proposed model is compared with passive detection model by Khan and n-gram based model by Piredda. We can see from our experiment that the proposed model performs better than the other 2 models. It is shown in Table 4.9.

**RQ4** *Which features indicate typosquatting?*

To understand the effect of different features, we rank the features based on information gain ratio. Table 4.10 shows information gain ratio of extracted features from typo pair. Our analysis shows that visual similarity, edit distance, and bigram similarity were among the most discriminating features. The performance of a supervised machine learning method depends on the combination of the selected features.

**RQ5** *How long to wait for achieving acceptable detection quality from log analysis?*

**Table 4.10:** Features used for typosquatting detection along with information gain ratio obtained for each feature

Features	Information Gain Ratio
Visual Similarity	0.6412
Edit distance	0.3649
keyboard Similarity	0.2241
Bigram Similarity	0.0802
Phonetic Similarity (Double Metaphone)	0.0521
Typo length	0.0517
Registered domain Length	0.0466
Phonetic Similarity (Soundex)	0.0363
Time Interval	0.0194

To find out the relation between DNS log size with TYPO detection parameter, we trained a supervised classifier with TYPO errors received from various size of logs and compared result for test dataset. DNS server generates 400,000 lines of valid log each day. Table 4.8 shows the impact of DNS log data size on typo detection quality. By using the two-day log, we can achieve 92.36% accuracy with 82.75% f-score. From this, we can say that to achieve acceptable detection quality less than 7-day log is sufficient.

**RQ6** *Can we find out which defensive domains to register to prevent typosquatting at the inception of a new website (without log analysis)?*

Before registering a website, i.e., without any relevant occurrence in the DNS log, we can still predict some candidate URLs for defensive registration. For that reason, we have used an RNN based model. From Table 5, we see that even at this early stage, we could achieve TPR of 40% (2 matches out of 5).

## 4.6 Summary

In this chapter at first, we have discussed the datasets used in our experiments and how we obtained it. Later we have explained our experimental setup and their parameters. Finally, we presented the findings and results of our work in the form of facts and figures.

# Chapter 5

## Empirical Study

We investigated patterns from our extracted misspelled typo domains from DNS logs. Analyzing log for the period of September 2017 to February 2018, we found that 67% DNS queries were for registered domain and 33% DNS queries for unregistered or typo domain. We identified more than 3 thousands unique typo pair from 6 month DNS log using developed typo detection model. We applied all those extracted typo pairs to find following insight.

### **5.1 Significant number of misspelled URLs exist in the DNS Log with single character modification of original URLs**

Character repetition and omission are among the leading causes of typographical error. A user may miss pressing a key while typing any web address or pressure on the key may be insufficient to write that character. Sometimes pressure on the key may be longer than required and hence that character is typed twice. Example of this type of errors is shown in Table [5.2](#).

We observe the ratio of modifying each of the misspelled URLs by single character insertion (SCI), single character omission (SCO) and single character substitution (SCS) from Fig. [5.1](#) where SCS leads by a big margin.



**Table 5.1:** Character repetition, omission and replacement example

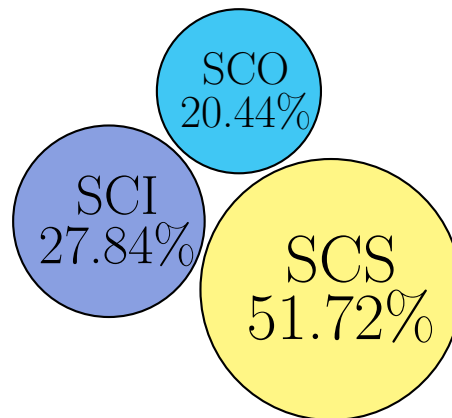
Category	Example	Observations
Keyboard proximity	newviaion.com.bd, ten- nia.com.bd	‘s’ replaced with horizontally adjacent key ‘a’
	sonslibank.com.bd, meghnsknit.com.bd	‘a’ replaced with horizontally adjacent key ‘s’
	nitolmiloy.com.bd, nao- gaom.gov.bd	‘n’ replaced with horizontally adjacent key ‘m’
Phonetic error	aviationnew.com.bd , jaarchitect.com.bd	Omission of terminal ‘s’
	taletalk.com.bd , ban- bais.gov.bd	‘e’ replaced with ‘a’
	worlditfaoundation.org.bd , gonokantha.com.bd	‘o’ replaced with ‘a’
Visual Similarity	admission.nu.ebu.bd , baraz.com.bd	‘d’ replaced with ‘b’
	buct.ac.bd, hangcr- splus.com.bd	‘e’ replaced with ‘c’
	taikingpoint.com.bd , electromartitd.com.bd	‘l’ replaced with ‘i’

**Table 5.2:** Character repetition, omission and replacement example

Observations	Example
Character inserted which is similar to adjacent (Repeating character)	daraz.comm.bd, cprogram- minng.com.bd, eastern- nuni.edu.bd, darazz.com.bd
Character omission from repeating character	cprograming.com.bd, dafodilvarsity.edu.bd, passport.gov.bd
Character replaced with adjacent char- acter	shongjog.rrg.bd, koonect.edu.bd, quantmm.org.bd, dhakaeu- ucationboard.gov.bd

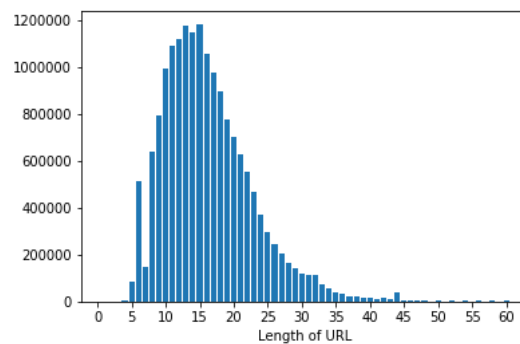
## 5.2 Short original URLs suffer more from URL poaching

We tried to find the relationship between URL length and typo hit. We found that URLs with short lengths had higher typo hit count. We have counted length wise request frequency to observe effect of length. Figure [5.2](#) shows length effect of processed



**Figure 5.1:** Percentage of modification types that caused typos

data.



**Figure 5.2:** Frequency of typo hit for different URL lengths

### 5.3 Typographical error can be categorized in different types

From our observation and analysis, it is seen that typographical errors can be of many types. According to the frequency of occurrence, we can deduce that phonetic error, keyboard proximity, and visual similarity among alphabet are the major categories of typographical error. Typographical error proportion is shown in Table [5.3](#). Here we discuss those.

**Keyboard proximity:** Users often mistype spelling for keyboard proximity. When user types in hurry, s/he may press left or right key of the desired key. Sometimes a user may type the wrong key which is situated below or above the desired key.

**Table 5.3:** Different type of typosquatting error and their proportion

Type	Ratio
Undefined	33.63%
Character Insertion	19.47%
Phonetic Perception	13.78%
Character Omission	11.67%
Keyboard Proximity	8.05%
Character Repetition	4.51%
Visual Similarity	4.20%
Existence of Dot , Dash and other symbol	3.31%
Existence of Digit	1.39%

If the wrong key is placed at a horizontal adjacent position of actual target key, we call it a horizontal key error. Similarly, vertical displacement is termed as a vertical key error. Horizontal key error frequency and vertical key error frequency are shown in Table 5.4.

**Table 5.4:** Keyboard proximity typosquatting for horizontal and vertical orientation

Key Orientation	Percentage
Replace by Horizontally adjacent key	78.2%
Replace by vertically adjacent key	21.2%

Though any key can be replaced by its neighbor key in the keyboard, we have found that some key alteration error happened repeatedly in our log. Some common keyboard character replacement error is shown in Table 5.1.

**Phonetic similarity:** Users may type wrong spelling for phonetic perception. Most of the time it occurs in case of the proper noun. As an example, ‘munsiganj’ is the name of a district in Bangladesh. Someone may type it as ‘munshigonj’ or ‘munshiganj’. From our experiment, we have seen that typo mistake occurs in 1988 pairs for phonetic perception. We classified phonetic error and some examples as shown in Table 5.1.

From our study the following observations are made:

- Character ‘h’ is often omitted from web address. For example, ⟨tec, tech⟩, ⟨teacher, teacer⟩
- Sub domain is mismatched. User often types .co instead of .com. For example, ⟨apple.com.bd, apple.co.bd⟩

- Subdomain name is misspelled sometimes. For government sites of Bangladesh, .gov subdomain is specialized. It is often typed as .govt. For example, ⟨passport.gov.bd, passport.govt.bd⟩
- Character ‘h’ is often inserted after Character ‘s’, ‘k’ or ‘t’. For example, ⟨narsingdi.gov.bd, narshingdi.gov.bd⟩, ⟨muktopaath.gov.bd, mukthopaath.gov.bd⟩
- Character ‘j’ and ‘z’ are often misspelled and used in place of each other. For example, ⟨daraz.com.bd, daraj.com.bd⟩
- Character ‘i’ and ‘e’ are often misspelled and used interchangeably for each other. For example, ⟨patuakhali.gov.bd, patuakhale.gov.bd⟩

**Visual similarity:** Some alphabetic characters have mutually similar look which often misguide people. We found significant number of cases where it happened. Some examples are shown in Table [5.1](#)

Following observations can be developed from our study.

- ‘c’, ‘o’, ‘e’ are characterized by an almost round envelope and look similar. For example, ⟨kuet.ac.bd, kuct.ac.bd⟩
- ‘t’, ‘f’, ‘l’, ‘i’ are characterized by vertical linearity and look similar. For example, ⟨film.com.bd, fiim.com.bd⟩
- ‘q’, ‘p’, ‘g’, ‘b’, ‘d’ are characterized by roundness and vertical linearity and look similar. For example, ⟨daraz.com.bd, baraz.com.bd⟩
- ‘m’, ‘n’ and ‘v’, ‘w’ are characterized by horizontal doubling of elements and look similar.

Web address or URL may be typed with mistyped digit or symbol. When users type url, they often confuse between hyphen and underscore. In our experiment, we observed that typo occurred in many cases due to the change in dot, dash, etc. Some examples are shown in Table [5.5](#).

**Table 5.5:** Dash and Dot related typosquatting error

Observations	Example
Omitted (.)	radiantcom.bd, xiclassad- missiongov.bd, mgnc- cbank.com.bd, join- bangladeshnavy.mil.bd, robisomadhan.com.bd
Dot replaced with dash	brac-org.bd, mmc-e- service.gov.bd, parjatan- gov.bd
Dash replaced with dot	ctg.wasa.org.bd, bise.ctg.gov.bd

# Chapter 6

## Conclusion

Typosquatting exists over two decades and continues to be a serious threat to this day. Detection of Typosquatting can be done analyzing DNS logs. However, to the best of our knowledge, no such work exists that consider the logs of the .bd domain, i.e., for the sites registered in Bangladesh. In this work, we have shown that a significant amount of typosquatting errors exist in the .bd domain. The typosquatting abuse can be a dangerous threat in Bangladesh where internet popularity is growing rapidly among mass people without formal computer literacy and awareness about security threats. In this work, we have used the typo errors extracted from DNS log of the .bd domain to identify and analyze typo errors. Although several prior works exist that examine various typosquatting techniques and how they get changed over time, none of them have considered how probable typosquatting pairs can be predicted for defensive registration. We also developed a tool to predict probable typo pairs for any given URL. To develop a predictive model, typosquatting pairs were detected from DNS log using supervised learning based model. The tool would help domain owners to take necessary measures for preventive registration in a cost-effective way. We have also empirically studied the user typo error patterns and reported the most frequent causes identified.

Our predictive typo registration model helps a new website owner to prevent typosquatting attack by defensive registration of most probable typo domains. List of predictive probable typo domains with rank helps a new website owner to make a better decision about how many domains he or she will register to protect his or her brand. Domain Registration authority like BTCL also can notify their potential clients with detected typo errors that are extracted from DNS log using the developer tools.

Our current work is restricted to websites of the .bd domain only. However, typosquatting is a global issue and brand owners all over the world have the practice of defensive domain name registration. We intend to do a comparative study of the defensive domain registration of sites in Bangladesh with those of other countries in the world. That will enable us to draw a bigger picture about the status of defensive domain registration in Bangladesh. So far, our study of typosquatting is limited to syntactic properties of domain names, to be more specific, the sequence of characters in a domain name. However, a domain name may have various semantic properties as well. A domain name may or may not contain a proper noun. Also, some domain names contain dictionary words, while some others have misspelled dictionary words in them (e.g. clarifai.com). Many names in .bd domain contain Bengali words spelled in roman letters. Our goal is to study the impact of such semantic features on the probability of a domain name being prone to typo error and extend our suggestion model accordingly.

# Bibliography

- [1] Van Lam Le, Ian Welch, Xiaoying Gao, and Peter Komisarczuk. Anatomy of drive-by download attack. In *Proceedings of the Eleventh Australasian Information Security Conference-Volume 138*, pages 49–58. Australian Computer Society, Inc., 2013.
- [2] FairWinds Partners. The Cost of Typosquatting. <https://www.fairwindspartners.com/resources-2/press/reports/the-cost-of-typosquatting/>, 2010. [Online; accessed 23-June-2018].
- [3] Kim Zitter. Researchers’ Typosquatting Stole 20GB of email From FORTUNE 500. <https://www.wired.com/2011/09/doppelganger-domains>, 2011. [Online; accessed 09-July-2018].
- [4] Mohammad Taha Khan, Xiang Huo, Zhou Li, and Chris Kanich. Every second counts: Quantifying the negative externalities of cybercrime via typosquatting. In *IEEE Symposium on Security and Privacy (SP)*, pages 135–150. IEEE, 2015.
- [5] Paolo Piredda, Davide Ariu, Battista Biggio, Iginio Corona, Luca Piras, Giorgio Giacinto, and Fabio Roli. Deepsquatting: Learning-based typosquatting detection at deeper domain levels. In *Conference of the Italian Association for Artificial Intelligence*, pages 347–358. Springer, 2017.
- [6] Tyler Moore and Benjamin Edelman. Measuring the perpetrators and funders of typosquatting. In *International Conference on Financial Cryptography and Data Security*, pages 175–191. Springer, 2010.
- [7] <https://github.com/typosquattingbd/typosquatting.git>, 2018. [Online; accessed 24 September-2018].



- [8] Muhammed Dastagir Husain and Anindya Iqbal. An empirical study on typosquatting abuse in bangladesh. In *International Conference on Networking, Systems and Security (NSysS)*, pages 47–54. IEEE, 2017.
- [9] Janos Szurdi, Balazs Kocso, Gabor Cseh, Jonathan Spring, Mark Felegyhazi, and Chris Kanich. The long” taile” of typosquatting domain names. In *USENIX Security Symposium*, pages 191–206, 2014.
- [10] Charan Piyush. A survey of the prominent effects of cybersquatting in india. *International Journal of Information Security and Cybercrime*, 4:47–51, 2015.
- [11] Yi-Min Wang, Doug Beck, Jeffrey Wang, Chad Verbowski, and Brad Daniels. Strider typo-patrol: Discovery and analysis of systematic typo-squatting. In *Proceedings of International conference on Steps to Reducing Unwanted Traffic on the Internet*, volume 6, pages 31–36, 2006.
- [12] Pieter Agten, Wouter Joosen, Frank Piessens, and Nick Nikiforakis. Seven months’ worth of mistakes: A longitudinal study of typosquatting abuse. In *Proceedings of Network and Distributed System Security Symposium*, pages 191–206. Internet Society, 2015.
- [13] Jeffrey Spaulding, Shambhu Upadhyaya, and Aziz Mohaisen. You’ve been tricked! a user study of the effectiveness of typosquatting techniques. In *International Conference on Distributed Computing Systems (ICDCS)*, pages 2593–2596. IEEE, 2017.
- [14] Nyang D. Mohaisen A. Spaulding, J. Understanding the effectiveness of typosquatting techniques. In *Proceedings of the fifth ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies*, 2017.
- [15] Jeffrey Spaulding, Ah Reum Kang, Shambhu Upadhyaya, and Aziz Mohaisen. A user study of the effectiveness of typosquatting techniques. In *International Conference on Communications and Network Security (CNS)*, pages 360–361. IEEE, 2016.
- [16] Nick Nikiforakis, Marco Balduzzi, Lieven Desmet, Frank Piessens, and Wouter Joosen. Soundsquatting: Uncovering the use of homophones in domain squatting. In *International Conference on Information Security*, pages 291–308. Springer, 2014.

- [17] Jeffrey Spaulding, Shambhu Upadhyaya, and Aziz Mohaisen. The landscape of domain name typosquatting: Techniques and countermeasures. In *International Conference on Availability, Reliability and Security (ARES)*, pages 284–289. IEEE, 2016.
- [18] Tingwen Liu, Yang Zhang, Jinqiao Shi, Jing Ya, Quangang Li, and Li Guo. Towards quantifying visual similarity of domain names for combating typosquatting abuse. In *Military Communications Conference (MILCOM)*, pages 770–775. IEEE, 2016.
- [19] Anirban Banerjee, Md Sazzadur Rahman, and Michalis Faloutsos. Sut: Quantifying and mitigating url typosquatting. *International Journal of Computer and Telecommunications Networking*, 55:3001–3014, 2011.
- [20] Lansing E. Brandbyge S. Pedersen J. M. Kidmose, E. Detection of malicious and abusive domain names. In *Data Intelligence and Security (ICDIS), 2018 1st International Conference on*, pages 49–56, 2018.
- [21] Singh M. M. Shariff A. R. M. Samsudin A. Zulkefli, Z. Typosquat cyber crime attack detection via smartphone. *Procedia Computer Science*, 24:664–671, 2017.
- [22] Onder Demir Buber, Ebubekir and Ozgur Koray Sahingoz. Typosquat cyber crime attack detection via smartphone. *Feature selections for the machine learning based detection of phishing websites*, pages 1–5, 2017.
- [23] <https://github.com/elceef/dnstwist>, 2018. [Online; accessed 04 November-2018].
- [24] <https://github.com/ring0lab//catphish>, 2017. [Online; accessed 11 November-2017].
- [25] <https://www.morningstarsecurity.com/research/urlcrazy>, 2012. [Online; accessed July 2012].
- [26] <https://github.com/nccgroup/typofinder>, 2016. [Online; accessed 22 November-2016].
- [27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

- [28] Frederic P Miller, Agnes F Vandome, and John McBrewhster. Levenshtein distance: Information theory, computer science, string (computer science), string metric, damerau? levenshtein distance, spell checker, hamming distance. 2009.
- [29] M Umanol, Hirotaka Okamoto, Itsuo Hatono, HIROYUKI Tamura, Fumio Kawachi, Sukehisa Umedzu, and Junichi Kinoshita. Fuzzy decision trees by fuzzy id3 algorithm and its application to diagnosis systems. In *Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the Third IEEE Conference on*, pages 2113–2118. IEEE, 1994.
- [30] B Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.
- [31] Tin Kam Ho. Random decision forests. In *Document analysis and recognition, 1995., proceedings of the third international conference on*, volume 1, pages 278–282. IEEE, 1995.
- [32] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.
- [33] David W Aha, Dennis Kibler, and Marc K Albert. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991.
- [34] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines*, 2000.
- [35] François Chollet et al. Keras. <https://github.com/fchollet/keras>, 2015.
- [36] *Bangladesh Telecommunications Company Ltd. (BTCL)*. Website: <http://www.btcl.com.bd>. Last accessed 11:51 am, January 27, 2019.
- [37] TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [38] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.