Low Level Design (LLD)

Malicious URL Detection

Written By	Diana Laveena DSouza
Document Version	0.1
Last Revised Date	05/01/2023

Document Control

Change Record:

Version	Date	Author	Author
0.1	05/01/2023	Diana Laveena DSouza	Introduction & Architecture Defined

Reviews:

Version	Date	Reviewer	Comments
0.1	05/01/2023		Document Content, Version Control and Unit Test Cases to be added

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

Abstract

With the development of Internet technology, network security is under diverse threats. In particular, attackers can spread malicious uniform resource locators (URLs) to carry out attacks such as phishing and spam. The research on malicious URL detection is significant for defending against these attacks. However, there are still some problems. For instance, malicious features cannot be extracted efficiently. Some existing detection methods are easy to evade attackers. We design malicious URL detection using Transformers models to solve these problems.

1 Introduction

1.1 Why is Low-Level Design Document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for News Summarization. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so the programmer can directly code the program from the document.

1.2 Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

2 Technical specifications

2.1 Dataset

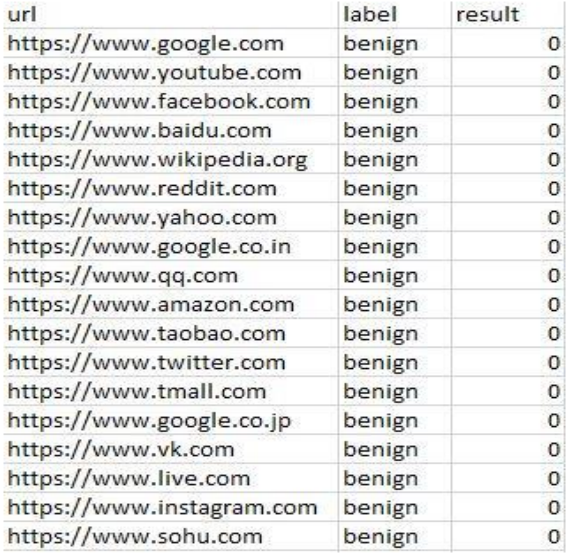
DataSet	Finalized	Source
balanced_urls	yes	https://www.kaggle.com/datasets/samahsadiq/benign-and-malicious-urls

2.1.1 Dataset Overview

The Dataset consists of news articles and their summary.

There are 626182 records in the training set and 6326 records in the validation set for classification.

Urls and their labels



url	label	result
https://www.google.com	benign	0
https://www.youtube.com	benign	0
https://www.facebook.com	benign	0
https://www.baidu.com	benign	0
https://www.wikipedia.org	benign	0
https://www.reddit.com	benign	0
https://www.yahoo.com	benign	0
https://www.google.co.in	benign	0
https://www.qq.com	benign	0
https://www.amazon.com	benign	0
https://www.taobao.com	benign	0
https://www.twitter.com	benign	0
https://www.tmall.com	benign	0
https://www.google.co.jp	benign	0
https://www.vk.com	benign	0
https://www.live.com	benign	0
https://www.instagram.com	benign	0
https://www.sohu.com	benign	0

2.1.2 Input schema

Feature Name	Datatype	Size	Null/Required
url	text	150	Not Required
label	varchar	10	Not Required
result	int	1	Not Required

2.2 Prediction

- The system presents the set of inputs required from the user.
- The user gives the required information.
- The System should detect the malicious URL.

2.3 Database

The database used in this project to store the data is the Cassandra database.

2.5 Deployment

Google Cloud

3 Technology Stack

Front End	HTML/CSS
Backend	Python Flask
Database	Cassandra
Deployment	GCP

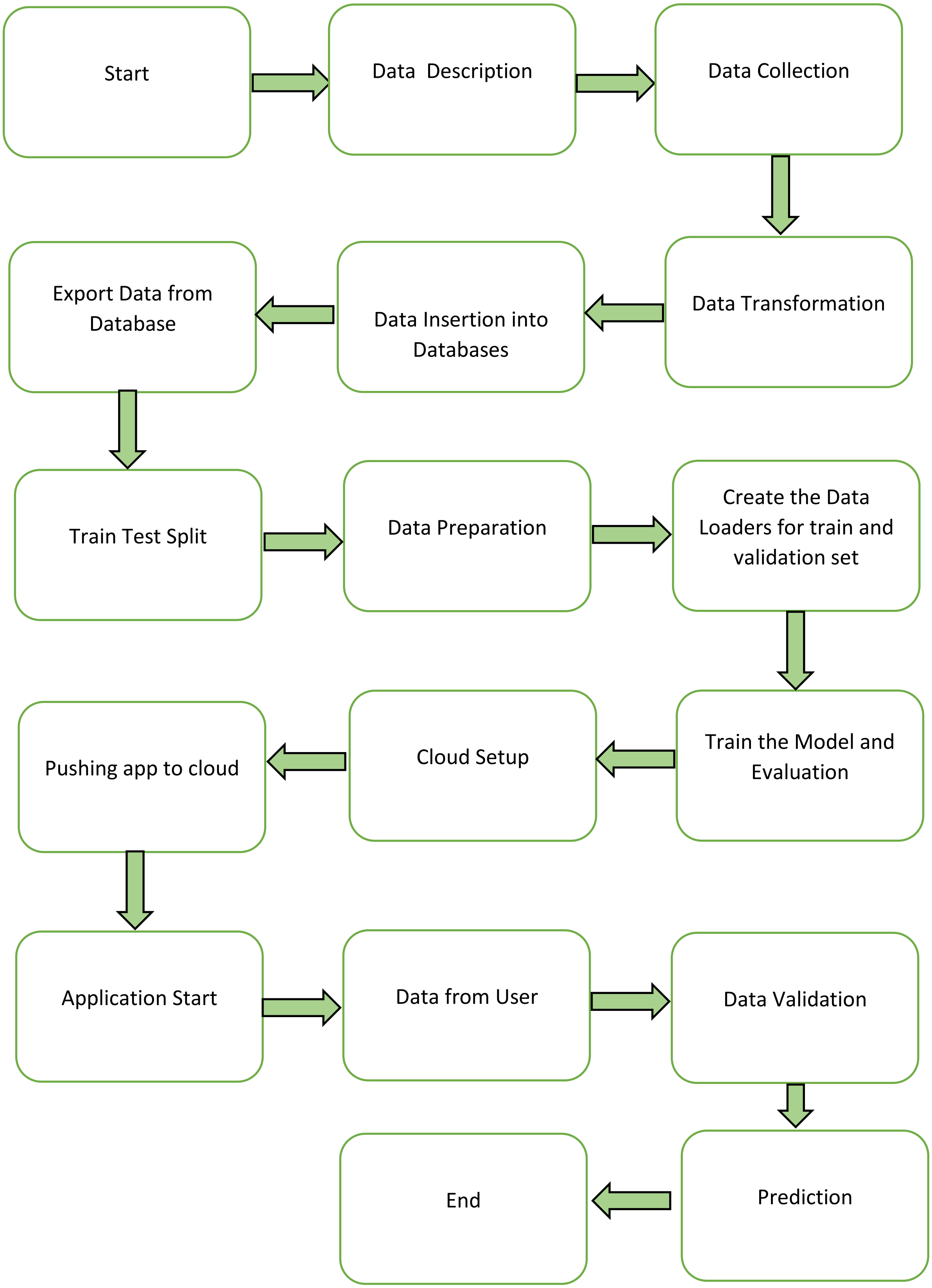
4 Proposed Solution

Refer: https://deepai.org/publication/a-transformer-based-model-to-detect-phishing-urls

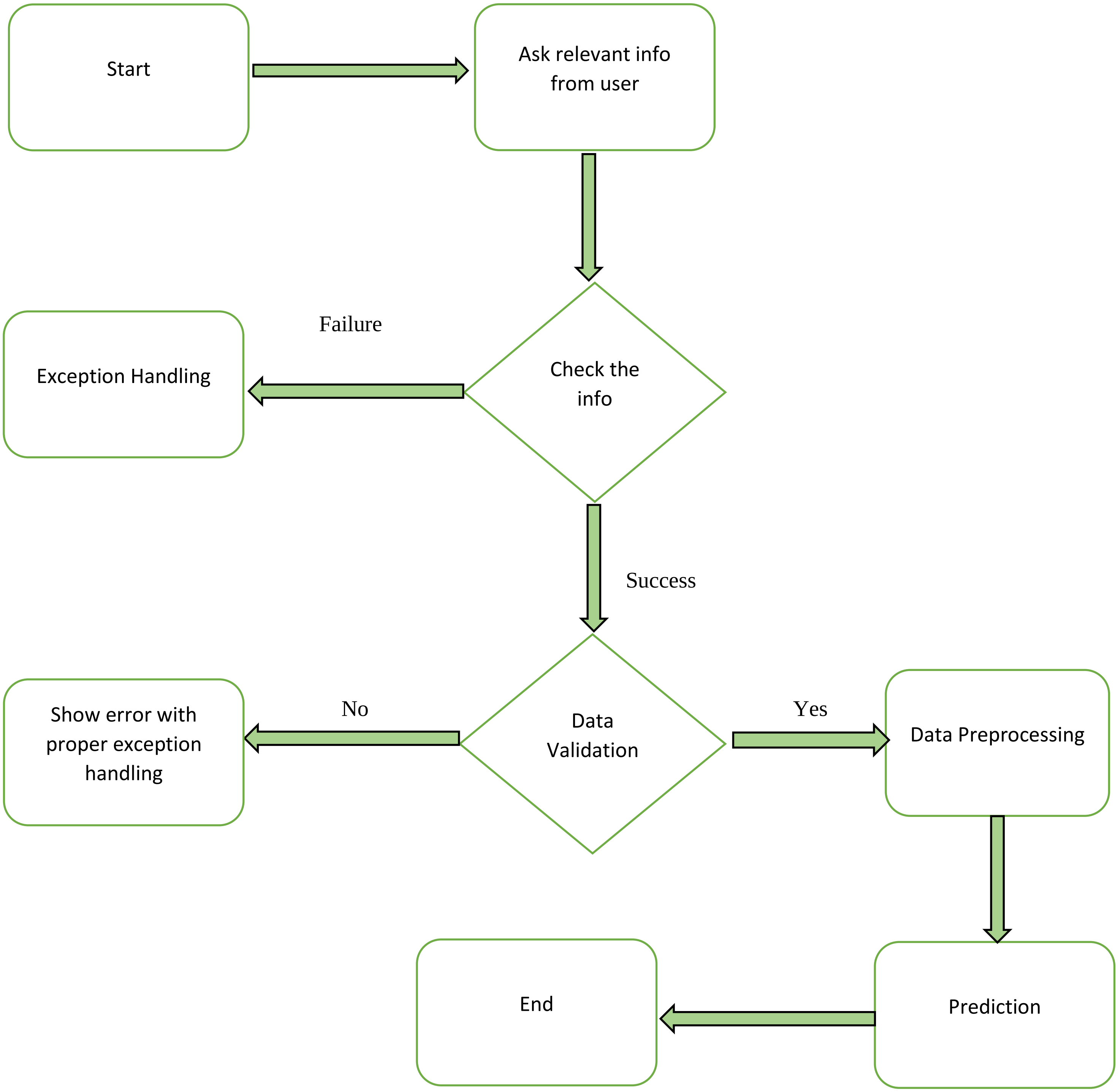
The research paper proposed TransformerEncoder with MultiheadAttention model for Classification. Finally, we selected the proposed model.

Baseline Model: TransformerEncoder with MultiheadAttention

5 Model Training/Validation Workflow



6 User I/O workflow



7 Test Cases

Test Case Description	Pre-Requisite	Expected Result
Verify whether the Application URL is accessible to the user	1. Application URL should be defined	The application URL should be accessible to the user.
Verify whether the Application loads entirely for the user when the URL is accessed	1. Application URL is accessible 2. Application is deployed	The Application should load entirely for the user when the URL is accessed.
Verify whether the user can input the text in all input fields	1. Application is accessible	The user should be able to input the text in all input fields.
Verify whether the user gets Submit button to submit the inputs.		The user should get Submit button to submit the inputs.
Verify whether the user is presented with results on clicking submit.		The user should be presented with results on clicking submit