ABV-IIITM, Gwalior

Problem Statement Report

Bug Report Priortization

Submitted By:
Anurag Yadav
2020BCS-013

Supervisor :
Dr. Santosh Singh Rathre

May 29, 2023

Contents

Chapter 1

Problem Statement Report

1.1 Background

With the increase in the development and usage of software and apps, the numbers of bugs associated has also increased. These bugs are report in a document format as a bug report. A triager is a person who reads the reports and manually classifies and assigns the priorities of the bugs based on its severity or urgency etc.

Prioritising bug reports is a crucial step in both software development and bug management. It entails deciding which reported defects should be fixed first based on severity, impact, and other considerations. Prioritising bugs enables teams and developers to allocate resources wisely and quickly address urgent problems. It also helps to manage large amount of bug reports, reduce the manual labour and speedup the processing ultimately cutting down the cost.

The process of prioritising bug reports can be automated using Natural Language Processing (NLP) methods. The system may more effectively prioritise issues by extracting useful data and trends from the textual content of bug reports using NLP models.

Considering the importance of this real world problem, I chose this problem to apply my knowledge of NLP and different models to improve the performance of current state of the art models. Try to use different architecture and improve the accuracy as the result.

This is a important problem to solve a real world issue, moreover its helps to progress

the existing state of the art models for further improvement and development so that they can be applied to other classification or NLP problems. Considering this the above problem should be of the common interest of others.

1.2 Project Objectives

The goal is to create a model, and test various models, that can assign priorities to problem reports and produce accurate results with acceptable consistency. The model should also be able to handle new bug reports consistently.

The following objectives are intended to be achieved by the proposed project :

- To create a model that can automatically prioritise bug reports using deep learning approaches after preprocessing the data using NLP techniques.
- To compare the outcomes of the proposed method to the state-of-the-art methodology after training the proposed model on open-source bug reports from various available datasets.

1.3 Requirements

In this section we will the about the software and hardware requirement for the proposed problem i.e to develop a bug report priortizer using NLP models.

The software and hardware requirements are as follows :

Availability : Since all of the software resources are free or open source, there is restriction in the availability. The hardware requirements are also satisfied.

1.4 System Overview

This section describes overall workflow process in development of the model. Transformer model are chosen, basic transformer architecture is mentioned.

1.5 Timeline

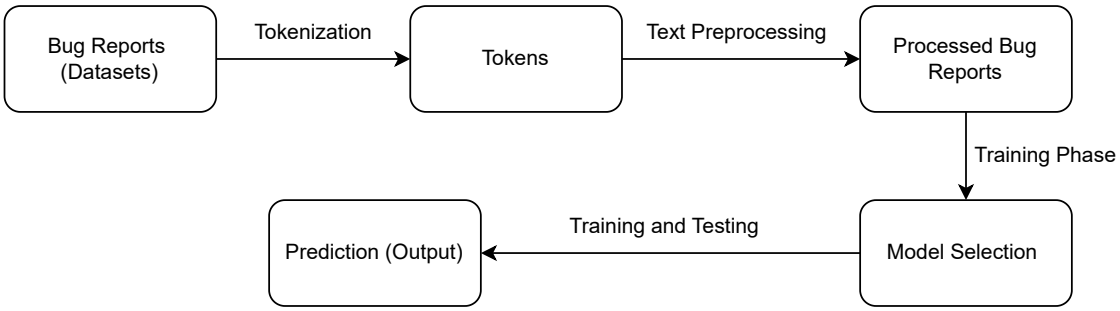

Figure 1.1: Workflow Model

Software	**Description**
Python	The programming language commonly used for NLP tasks.
Deep Learning Frameworks	TensorFlow, PyTorch, or Keras for building and training neural networks.
NLP Libraries	NLTK, spaCy, or Gensim for text preprocessing, tokenization, and other NLP tasks.
Pre-trained Models	Libraries or APIs to access pre-trained models like BERT or GPT.
Development Environment	Jupyter Notebook, PyCharm, or any IDE for coding and experimentation.

Table 1.1: Software Requirements

Hardware	**Description**
CPU	A modern multi-core processor to run the code and perform computations.
GPU (optional)	For training deep learning models efficiently, a dedicated GPU like NVIDIA GeForce or Tesla can significantly speed up the training process.
Memory (RAM)	Depending upon the datasaet size, 16GB or 32GB RAM should be sufficient.
Storage	To store the dataset, pre-trained models, and any intermediate or output files.
Internet Connection	Required for downloading pre-trained models, datasets, and any necessary libraries.

Table 1.2: Hardware Requirements

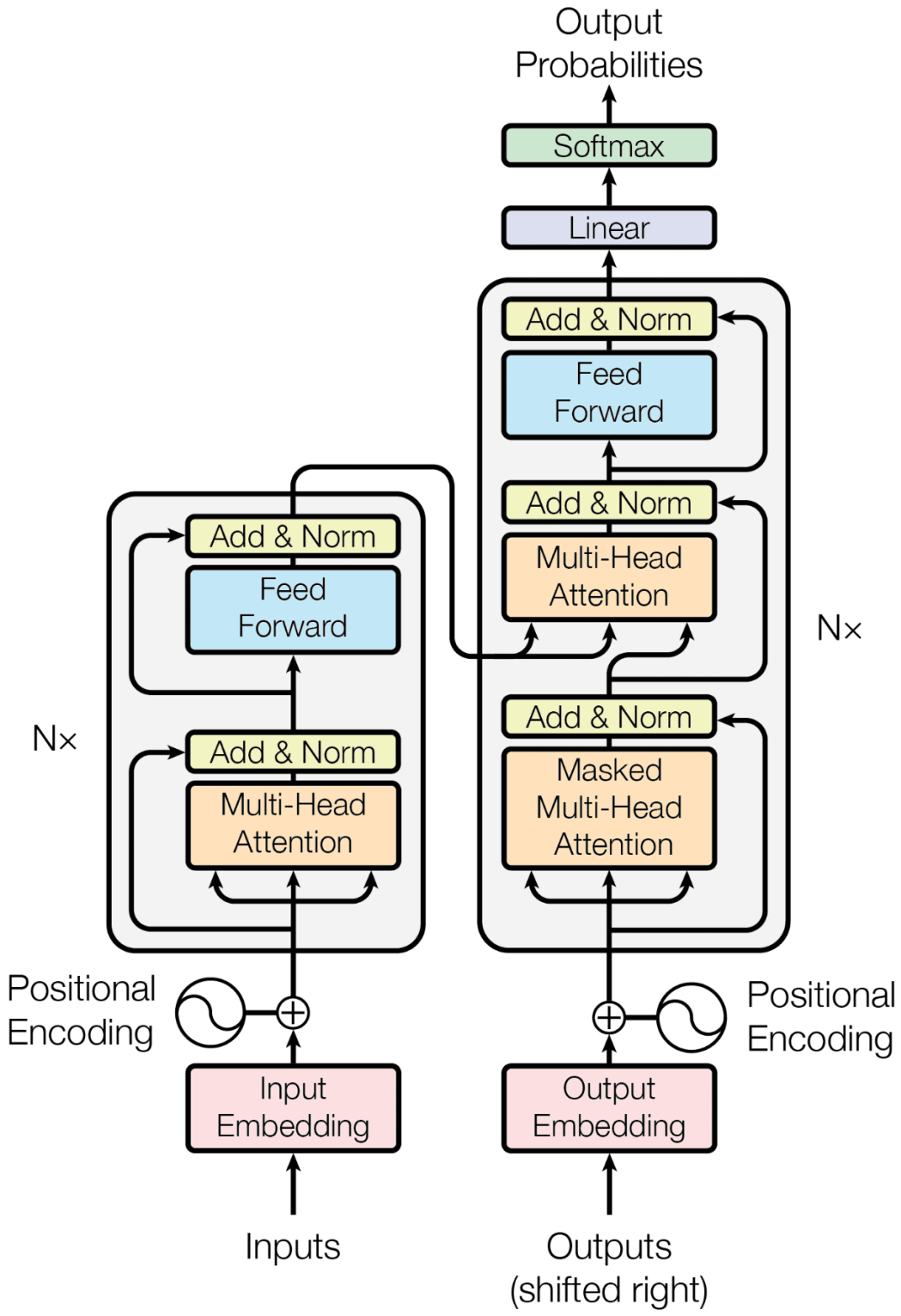

Figure 1.2: Transformer Model Architecture

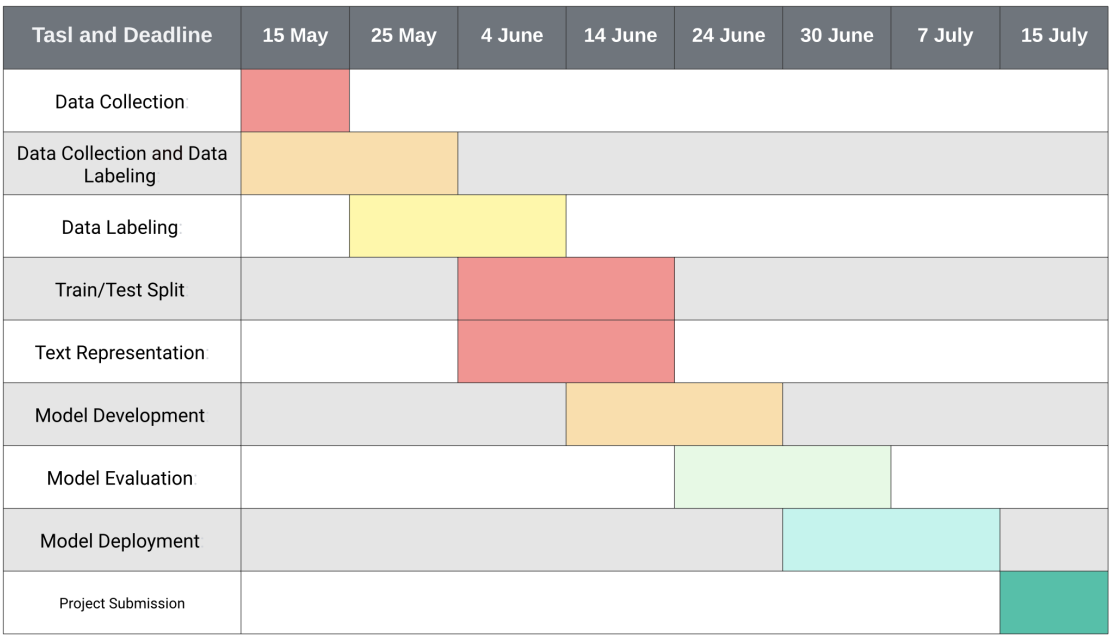

Figure 1.3: Gantt Chart