A

PROJECT REPORT

ON

VISUALIZATION OF CRIME HOTSPOTS THROUGH ANALYSIS OF ONLINE NEWSPAPER ARTICLES

Submitted in Partial Fulfilment for The Award of Degree of BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

BY

NEHA DINESH PRABHU (160117737011)
&
PRERANA RAJOLE (160117737013)

Under the guidance of

Dr. M. Trupthi
Assistant Professor
Dept. of IT, CBIT.

DEPARTMENT OF INFORMATION TECHNOLOGY
CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY(A)
(affiliated to university; Accredited by NBA (AICTE) and NACC (UGC), ISO Certified 90001:2015)
Kokapet (V) GANDIPET (M), HYDERABAD -500075

Website: www.cbit.ac.in

2020-2021

CERTIFICATE

This is to certify that the work entitled "**VISUALIZATION OF CRIME HOTSPOTS THROUGH ANALYSIS OF ONLINE NEWSPAPER ARTICLES**" submitted by **NEHA DINESH PRABHU (160117737011)** and **PRERANA RAJOLE (160117737013)** in partial fulfilment of the requirements for the award of degree of **BACHELOR OF ENGINEERING** in **INFORMATION TECHNOLOGY** to **CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY(A)**, affiliated to **OSMANIA UNIVERSITY**, Hyderabad is a record of bonafide work carried out by them under my supervision and guidance. The results embodied in this report have not been submitted to any other University or Institute for the award of any other Degree or Diploma.

Project Guide
Dr. M. Trupthi
Assistant Professor, IT Dept.
CBIT, Hyderabad.

Head of Department
Dr. K. Radhika
Professor, IT Dept.
CBIT, Hyderabad.

DECLARATION

We declare that the project report entitled "**VISUALIZATION OF CRIME HOTSPOTS THROUGH ANALYSIS OF ONLINE NEWSPAPER ARTICLES**" is being submitted by us in the department of Information Technology Chaitanya Bharathi Institute of Technology (A), Osmania University.

This is record of bonafide work carried out by us under the guidance and supervision of **Dr. M. Trupthi,** Assistant Professor, Dept. of IT, C.B.I.T.

No part of the work is copied from books/journals/internet and whenever the portion is taken, the same has been duly referred in the text. The report is based on the project work done entirely by us and not copied from any other resources.

Neha Dinesh Prabhu (160117737011)

Prerana Rajole (160117737013)

ACKNOWLEDGEMENTS

It is our privilege to acknowledge with deep sense of gratitude and devotion for keen personal interest and invaluable guidance rendered by our Project Guide **Dr. M. Trupthi**, Assistant Professor, Department of Information, Chaitanya Bharathi Institute of Technology.

We take the opportunity to express our thanks to **Dr K. Radhika**, Professor & Head of IT Department, CBIT for her valuable suggestions and moral support.

We are grateful to our Principal **Prof. G.P.S. Varma**, Chaitanya Bharathi Institute of Technology, for his cooperation and encouragement.

Finally, we also thank all the staff members, faculty of Dept. of IT, CBIT, and our friends, who with their valuable suggestions and support, directly or indirectly helped us in completing this project work.

ABSTRACT

Cities in India are expanding at an unimaginable pace. Growing along with the cities is the crime rate. Crimes like murder, kidnapping, extortion, rape, eve teasing have increased exponentially over the past few years. Based on data available public, through news articles/newspapers it is observed that some locations are more likely to be involved in an incident than others. This data is scattered over various resources and platforms like the local newspaper, national newspapers, digital news media, social media etc. There is no proper compilation of this data which can be referred to by the general public for safety and security purposes.

This project's aim is to create an end-to-end software prototype that mines data from online news stories, extracts important details like the crime location and type of crime, and displays the processed data along with the number of crimes for a particular location on a Heat Map. By using "transfer learning" which is the method used to fine-tune pre-trained models to accommodate any additions in the dataset, the training time will be greatly decreased and the accuracy will be improved by using this methodology. Thus, the output achieved will have lesser error rate and will incorporate the fastest and most intelligent model in the current decade.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API – Application Programming Interface

CUDA – Compute Unified Device Architecture

HTML – HyperText Markup Language

NLP – Natural Language Processing

NLG – Natural language Generation

NLU – Natural language Understanding

NER – Named Entity Recognition

BERT – Bidirectional Encoder Representations from Transformers

CSS – Cascading Style Sheets

CNN – Convolutional Neural Network

RNN – Recurrent Neural Network

MLM – Masked Language Modelling

1. INTRODUCTION

With improvements in technology our country is growing at a fast pace which has led to a great deal of urbanization. Instead of the crime rate decreasing, there has been a surge of crimes in recent years. Citizens come across crimes like theft, murder, etc. being committed on a daily basis in the newspapers.

According to a study, a total of 51.5 lakh cognizable crimes were recorded nationally in 2019, with 32.2 lakh Indian Penal Code crimes and 19.4 lakh Special and Local Laws crimes. The crime rate per 100,000 has risen from 383.5 in 2018 to 385.5 in 2019, reflecting a 1.6 percent (50.7 lakh) annual spike in the number of cases registered. Murder, abduction, attack and death by negligence accounted for more than a fifth of all reported crimes (10.5 lakh), which included violent acts such as murder, kidnapping, assault, and death by negligence.

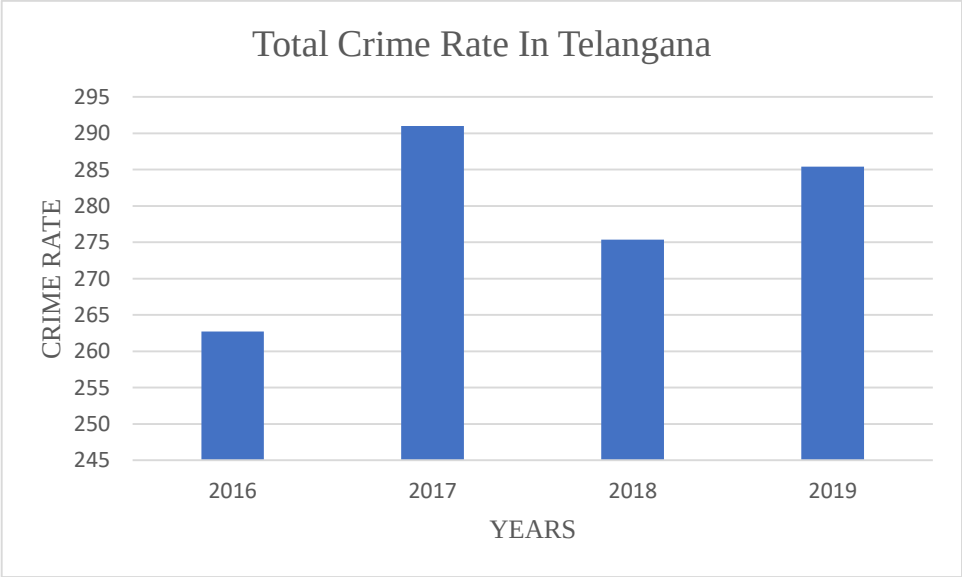


Figure 1.1 Crime rate in Telangana (2016-2019)

In 2019, 18,051 cases were filed under the Hyderabad police commissionerate limits, compared to 16,012 cases in 2018. This information can be very useful when it comes to knowing about the security and safety of a particular location. It is very important for people to know how safe their society is so they can tale the appropriate measures to protect themselves.

As there is no way to find out how secure and safe a place is, parents are always anxious for the safety of their children when they move different cities for higher education or jobs. Parents will also be more relaxed if they know the number of crimes that have happened in a location when their children go out. It is also very important for families moving to new places or going on vacation to know how safe the locality is.

Taking these points as the motivation the aim is to build a software that analyses crimes mentioned in the news articles to educate users about the safety of a location. After analyzing the information, it can be visualized on a heat map for better and easier understanding of how secure the location is.

1.1 Problem Statement

The aim of our project is to build a model which will utilize Natural Language Processing and Deep Learning to process publicly available data like newspaper articles from online sites to recognize the crime type and location where the crime was committed and build a corpus which can be visualized on a map.
The model will ensure the citizens safety by giving them a consolidated image of the crimes in a location so they can take the appropriate measures while visiting that place.

1.2 Applications

There are multiple places where the model can be used:

- Real Estate or housing websites: while renting or buying a house the customers can check how secure the locality and then proceed to buy the house. At the same time if a user is interested in renting his house he can adjust the price of the flat based on how secure or safe the location is.
- The general public can see how safe it is to go to a place and take any necessary measures before heading out or avoid places with higher crime rates altogether.
- The local police stations can increase vigilance in those areas where the crime rate is high and implement measures that will reduce the crime rate.

1.3 Organization of Report

Chapter 1 deals with introduction of the project and explains the purpose of the project.
Chapter 2 deals with the literature survey

Chapter 3 deals with the requirements that are needed in order to execute the project

Chapter 4 deals with the methodology, system design and features of the project

Chapter 5 deals with the implementation of the project

Chapter 6 involves the results of the project

Chapter 7 involves conclusion and future scope of the project.

2. LITERATURE SURVEY

The following research papers have been reviewed based on the problem statement and project requirements:

2.1 Survey Paper 1

Paper Title: Building an Urban Theft Map by Analyzing Newspaper Crime Reports

Authors: Laura Po; Federica Rollo

Publication: IEEE

Description

Newspaper articles are mined using a focused crawler which is a Java application that connects to the section about the thefts on the newspaper website. The application parses the content of each page, detects the useful information given below and stores them in a database

- address where the theft took place (municipality, area and street),
- the date and the time of publication of the news
- the stolen object

By using an automatic text analysis tool, the author was able identify and extract the stolen object from the title and description of the news

The geocoding - the process of deriving the latitude and the longitude coordinates from a textual address, is performed by the Google Maps Geocoding API.

Visualisation on maps is done for the following:

- locating on a map the crimes happened in a specific month - Crime Map analysis
- displaying of a heat map to localize critical - Density Crime Map visualization
- identifying the number of crimes per district within the city - Hot spot detection

The application takes in input the month and the year of interest, connects to the database and retrieves the information on the thefts in the specified period. It employs Polymaps a JavaScript library using Scalable Vector Graphics for creating the maps.

Advantages of the method used in the method

- Extracted data is visualized based on user queries on maps or in form of graphs.
- Includes a feature to compare the number of crimes to get an idea about the growth or decrease of thefts over the time.

Disadvantages of the method used in the method

- Only takes theft as the criminal activity.
- No integration of real time crime news from newspapers or police data.

2.2 Survey Paper 2

Paper Title: Crime news analysis: Location and story detection

Authors: Mehedee Hassan; Mohammad Zahidur Rahman

Publication: IEEE

Methodology

The model implements a crawler using simple tools and libraries like selenium and newspaper to collect plain text from online newspaper. The crawler takes the title, the body contents and the date of a news

The extracted data is then stored in a database. Implements MongoDB as backend for saving text data collected by the crawler.

Data pre-processing is performed. The common techniques like word tokenization, removing stop words and special characters, case conversion, lemmatization is performed is performed using NLTK library on each article before saving in database for processing purpose.

The model in the paper is support vector machine (SVM) classifier to classify the news article as crime-related news and non-crime related news.

A sentence classifier is implemented to classify the sentence which has probable crime location. The sentences with a location in crime news is used as training data. Crime location extraction is performed using named entity recognition (NER) and sentence classification. NER is performed over each sentence to get the sentence with location names.

The accuracy obtained by using SVM for classifying articles as crime and non-crime and for classifying sentences is 95% and 72% respectively.

Advantages of the method used in the paper

- Accuracy of 95% when classifying articles as crime or non-crime articles.
- Uses NER to extract locations

Disadvantages of the method used in the paper

- Entities that can affect the location of the crime have been ignored.

2.3 Survey Paper 3

Paper Title: Location Exploiting Categorization of Online news for Profiling City Areas

Authors: Vignesh Rao; Jayant Sachdev

Publication: IEEE

Methodology

The approach is this paper is to assign an output class to the news articles based on the content.

The process starts with the Data Retrieval module which is the collection of the dataset wherein their self-developed web scraping algorithm is employed to extract the actual text from the webpage.

The dataset so formed is then divided into test data and train data. Train data forms up to 80% whereas test data forms 20% of the dataset collected.

Data pre-processing methods like tokenization, stemming, removal of stop words are applied to the train data which is then used as an input to the classifier for training.

In the paper classification algorithms such as SVM (support vector machine), Naive Bayes, Random forest are used.

The same data pre-processing methods are used for the test data and this acts as an input to the trained classifier which predicts the output class of the test news articles.

The respective algorithms used and accuracies obtained for classifying the articles according to their locations are as follows:

- Naïve Bayes: accuracy of 82%
- SVM (Support Vector Machine): accuracy of 78%

- Random Forest: accuracy of 85%

Advantages of the method used in the paper

- Used customized web crawler
- Tried several classification models and evaluated each one.

Disadvantages of the method used in the paper

- Includes only 5 metropolitan cities for comparison.
- No Named Entity recognition technique used to extract data such as location, time of event, name of victim etc.

2.4 Survey Paper 4

Paper Title: Building an Urban Theft Map by Analyzing Newspaper Crime Reports

Authors: Alessandro Bondielli; Pietro Ducange; Francesco Marcelloni

Publication: IEEE

Methodology

In this paper, they propose an approach to perform profiling of city areas based on articles of local online newspapers, by exploiting information regarding the text as well as metadata such as geo-localization and tags.

In particular, they use tags associated with each article for identifying macro-categories through clustering analysis on tags embedding.

The online newspapers are monitored continuously, thus producing a news stream to be analysed. They show experiments performed on the city of Rome, considering data from 2014 to 2018.

They built a text categorization model that is able to assign one of the macro-categories to a previously unseen and potentially not tagged news article.

The paper employs a text categorization model based on SVM to label online a new article, represented as Bag-of-Words, with one of such categories. The categorization approach has been integrated into a framework recently proposed by the authors for profiling city areas exploiting different web sources of data

The authors chose to represent each article with BOW only considering the title and the summary in order to alleviate problems linked to the sparsity of the representation.

To train the text categorization model, a subset of the articles adopted for the clustering stage was used and were labelled them with the identified macro categories.
They also discuss the results obtained by adopting different classifiers and present that the best classifier, namely an SVM, can achieve an validation accuracy and an f1-score up to 93% and 79%, respectively.

Advantages of the method used in the paper

- Classifies articles into multiple categories.
- Uses different classification methods like Logistic Regression (LR), Decision Tree (DT), and Support Vector Machine (SVM) and proposes the best algorithm.

Disadvantages of the method used in the paper

Although the model has a high accuracy it uses SVM which results in a high training.

3. SYSTEM REQUIREMENTS SECIFICATION

3.1 Functional Requirements

A Functional Requirement (FR) is a statement that describes the service that the program must provide. It refers to a software system or a part of one. A function is nothing more than the inputs, actions, and outputs of a software system. Data manipulation, business processes, user interaction, and any other basic functionality that determines what function a device is likely to perform are all examples of this.

The project requires the following functional requirements:

- To build a software that analyses crimes mentioned in the online news articles to educate users about the safety of a location.
- The input for the model are newspaper articles scraped from the web
- Identify the crimes and crime location with the appropriate tags with highest optimal accuracy after extracting and pre-processing the online newspaper articles.
- A visualization feature, to display location of crimes on a map based on the query given by the user.

3.2 Non-Functional Requirements

Non-functional specifications define a software system's quality attributes. These requirements are specifications that defines the system's operating capabilities as well as the constraints that help it perform better. They define the parameters that can be used to evaluate a system's efficiency.

The non-functional requirements involved in the project are:

3.2.1 Performance Requirements:

To create a balance between the performance and accuracy of the model.

3.2.2 Environmental Requirements:

The development of model does not require any environmental aspect. Since it is developed on Google Colab, change of OS does not create any problem.

3.2.3 Scalability:

The system is first tried and tested on a medium scale i.e. the articles were collected for Hyderabad. It can be scaled using cloud services such as AWS, GCP.

3.2.4 Maintainability Requirements:

- Same articles should not web-scrape again.
- Location coordinates once acquired must not be queried again using API as API call takes time to return which might lag the system.

3.3 Software Requirements

Table 3.1 Software Requirements

Operating System	Any OS
Programming Language	Python
Platform	Jupyter Notebook/ Google Colab
Libraries	Transformers, Torch, Selenium, Geopy.geocoder, Bokeh, etc.
API's	Google API

3.3.1 Google Colab

Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook. The notebook allows users to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. The notebooks created are stored in your Google Drive account. The notebook can also be shared and connected to your drive to store any input or output files.

A major advantage of the notebook is the ability to connect to a hardware accelerator (GPU or TPU) based on the requirements of your project.

3.3.2 Transformers

The Transformers library offers general-purpose architectures for Natural Language Generation (NLG) and Natural Language Understanding (NLU). The library was first introduced in the paper 'Attention Is All You Need'.

It has over 32 pre-trained models in multiple languages and strong interoperability between TensorFlow 2.0 and PyTorch. It is a sub-library of the HuggingFace library. For all of the models available, the library currently contains PyTorch implementations, pre-trained model weights, usage scripts, and conversion utilities. A major advantage of using the models offered by the library is the easy customizability of pre-trained models.

Of the many models offered by the library the project uses the pre-trained BERT model for the NER task. The following functions – 'BertTokenizer', 'BertConfig', BertForTokenClassification and AdamW are used from this library.

3.3.3 Geopy.geocoder

geopy is a Python client which is used by many geocoding web services. It allows developers to locate the coordinates of an address, city, country or landmark across the globe using data sources or third-party geocoders.

Geolocation services like Google Maps and Nominatim, have their own class in geopy.geocoders abstracted from the API of the service. A geocode method, is defined by geocoders to resolve a location from an input string. In this project the Nominatim geolocation service is used.

Nominatim is built on the osm2pgsql which is a software used to import OpenStreetMap data into a PostgreSQL database. using the alternative gazetteer output option. The indexing and searches are performed using the combination of php, plpgsql and C.

3.3.4 Selenium

Web scraping, also known as "crawling" or "spidering," is a technique used for the automatic gathering data from an online source.

Selenium is a web-based automation platform that is open-source. Although often used in the industry for research, it is an excellent tool can also be used for web scraping. The WebDriver protocol is used by the Selenium API to control web browser such as Chrome, Firefox, or Safari.

Locating data on a website is one of Selenium's key use cases, whether for checking (to ensure that a particular feature is absent or present on the page) or extracting data and saving it for further processing (web scraping).

In Selenium, CSS Selectors are string patterns that are used to recognize an element based on the combination of an HTML tags, ids, classes, and attributes. Even though locating using CSS Selectors is more complicated than any methods that exist, it is the most common strategy of advanced users because it can access those elements that have no name or ID.

3.3.5 Bokeh

Bokeh is a Python library that allows us to make interactive visualizations. It allows us to create stunning graphics ranging from basic plots to complex dashboards with live data.

It also provides high-performance interactive charts and plots. The output can be obtained on various mediums such as HTML pages, notebooks (Colab, Jupyter) and servers. It is also possible to embed the bokeh plots in bothe Flask and Django apps. The bokeh.palettes module.an

Bokehheat provides a bokeh based plotting implementation that uses either interactive boolean data, categorical data or numerical data to plot heatmaps or dendrograms. Bokeh provides us with multiple color palettes in the bokeh.palettes module.

Bokeh's underlying concept is that graphs are built up one layer at a time. User begins by creating a figure, which they then embellish with elements known as glyphs. Glyphs come in a variety of forms, depending on the intended application: circles, lines, patches, bars, arcs, and so on.

Some advantages of using the bokeh library over the seaborn plots are the options given for zooming into the plot created and the hover tool. This hover tool is used to display the tooltip of each of the glyphs created.

3.3.6 Google API

Google APIs are application programming interfaces (APIs) created by Google that enable users to communicate with Google Services and integrate them with other services. Authentication, authorization, and auditing are all part of Google Cloud API access control. To create a API key, a user first needs to create a project under the editor role. After that they must navigate to the ‘APIs and Services’ in the cloud console. Next, select the create credentials followed by the ‘API key’. The API key created is a long

string consisting of a combination of lower and upper case letters, dashes and numbers. Depending on the type of service a credit card might be asked for payment purposes.

Restrictions can also be added on which websites will be allowed to use the key. The Google APIs use the OAuth 2.0 protocol for authentication and authorization of the API key generated. The linchpin of this industry-standard protocol for authorization is the client developer simplicity. At the same time, it provides specific authorization flows for desktop applications, web applications and mobile phones.

3.4 Hardware Requirements

Table 3.2 Hardware requirements

Processor	Intel CORE i5
RAM	8 GB
Hard Disk space	200 GB
GPU	Minimum of 8GB

3.4.1 Graphics Processing Unit (GPU)

A graphics processing unit (GPU) is a specialized processor that was created to speed up the rendering of graphics. They can process a large amount of data at once, making them ideal for machine learning, video editing, and gaming.

As mentioned above the Colab environment gives us an opportunity to change the accelerator of our notebook. Google Colab, an online browser-based platform that allows us to train our models on machines for free, provides a single GPU that can be used up to 12 hours continuously. After that, the virtual machine is fully cleared, and the session must begin again.

While training the model, three of the GPU's which were assigned at random were tested. After connecting to the GPU, a user is given access to 13 GB RAM and 64 GB disc space.

Table 3.3 GPU comparisons

GPU Model	Tesla T4	Tesla K80	Tesla P4
Manufacture Year	2018	2014	2016
Core clock speed	585 MHz	562 MHz	886 MHz
Training Time	59 minutes	120 minutes	93 minutes

3.4.2 CUDA

Nvidia's CUDA is a parallel computing framework and (API) model. It enables software engineers and developers to use the GPU with CUDA support for any general-purpose processing. The CUDA platform is a software layer that allows compute kernels to have direct access to the GPU's virtual instruction set and parallel computational elements.

The CUDA Toolkit is a software development environment for GPU-accelerated applications. The Toolkit enables users to develop, customize and deploy applications on GPU-accelerated desktop workstations and cloud platforms. To develop and deploy your application on major architectures, it includes GPU-accelerated libraries, optimization and debugging tools, a C/C++ compiler, and a runtime library.

3.5 Dataset

For our project, a dataset containing newspaper articles was used. Custom changes were made in the dataset by increasing the labels for the entities to be recognized. The dataset was labelled according to the BIO scheme. BIO stands for Beginning, Inside and Outside of a sentence segment. The dataset had 4 columns:

- Sentence # - holds the sentence number
- Word – holds the word
- POS – holds the parts of speech tag for the corresponding word
- Tag – contains the BIO tag of the word

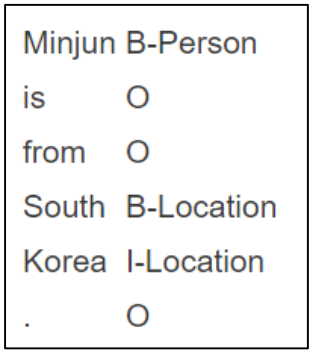


Figure 3.1 Example of BIO scheme

The following are all the unique tags in the dataset:

O	B-nat	I-art
B-gpe	B-org	B-eve
I-geo	I-eve	I-nat
B-per	B-tim	I-org
I-gpe	B-art	I-per
I-tim	B-geo	

4. METHODOLGY

4.1 Introduction

The aim of the system is to take newspaper articles as input, process these sentences to identify the location of the crime and also what kind of crime has occurred. For performing the NER task the BERT model is used.

4.1.1 System Architecture

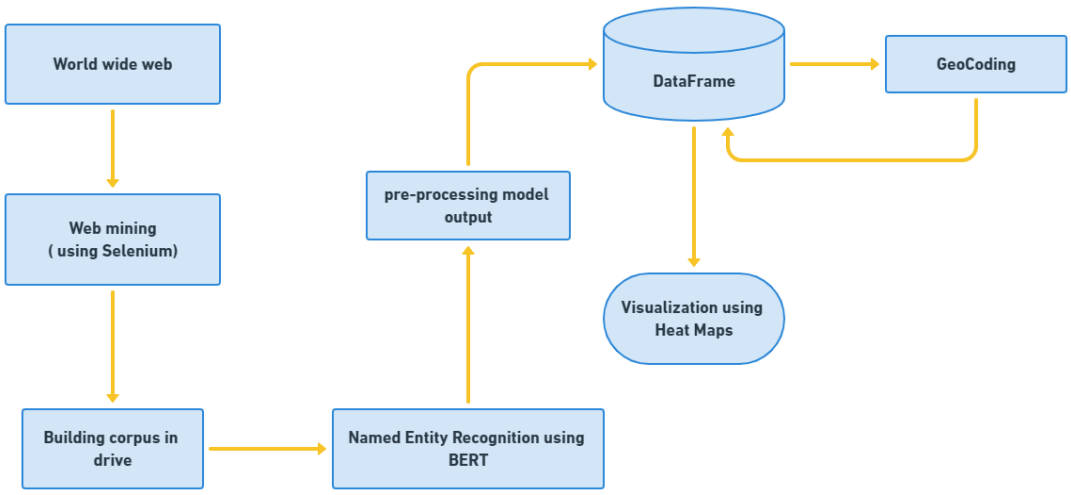


Figure 4.1 Architecture of proposed system

The flow of the proposed methodology is as follows:

- Web scrape news articles from online sources like the Times of India, Siasat to extract the title, content and any necessary meta data of the article. Build a corpus, which is a large collection of text. The mined articles were all saved in a folder in the google drive.
- Feed the pre-processed data to the NER model (built by fine tuning the pre-trained BERT model) in order to obtain features such as location, time, crime type like murder, sexual assault, kidnapping, etc. Any un-necessary locations mentioned in the article were eliminated by customizing the dataset.
- Store the location of crime and type of crime in a csv file. If the location obtained is not pre-existing in the database obtain the longitude and latitude of the location using geo-coding and create a new entry in the csv file.
- Display all the information in the csv file in the form of a heat map to show the recent most crime for that location along with the number of crimes that have occurred.

4.2 BERT

It's a Google-developed Transformer-based machine learning technique for natural language processing (NLP) pre-training. The model is pre-trained on unlabeled data extracted from Wikipedia having 2,500 million words and BooksCorpus having 800 million words.

Unlike other previous models BERT is bi-directional and also takes into account the context for each occurrence of a given word. For instance, although the vector for "tie" will have the same word2vec vector representation for both of its occurrences in the sentences " The game ended in a tie." and " I need to tie my hair back.", BERT will provide a contextualized embedding that will be different according to the sentence.

The language representation model was pre-trained using the following unsupervised prediction tasks:

- Next Sentence Prediction
- Masked Language Modeling

The bi-directional characteristic of the model comes from the concept of MLM. 15 percent of the words in each word series are replaced with a [MASK] token before being fed into BERT. Based on the meaning given by the other, non-masked words in the sequence, the model then attempts to predict the original value of the masked words. In technical terms, to predict the output requirement is:

- To add a classification layer above the encoder output.
- Transform the output vectors into the vocabulary dimension by multiplying them by the embedding matrix.
- To use a Softmax function to calculate the likelihood of each word in the vocabulary.

The model can be modified to perform NER tasks by adding a classification layer that will predict the NER label after feeding the output vector of each token to it.

4.2.1 BERT Architecture

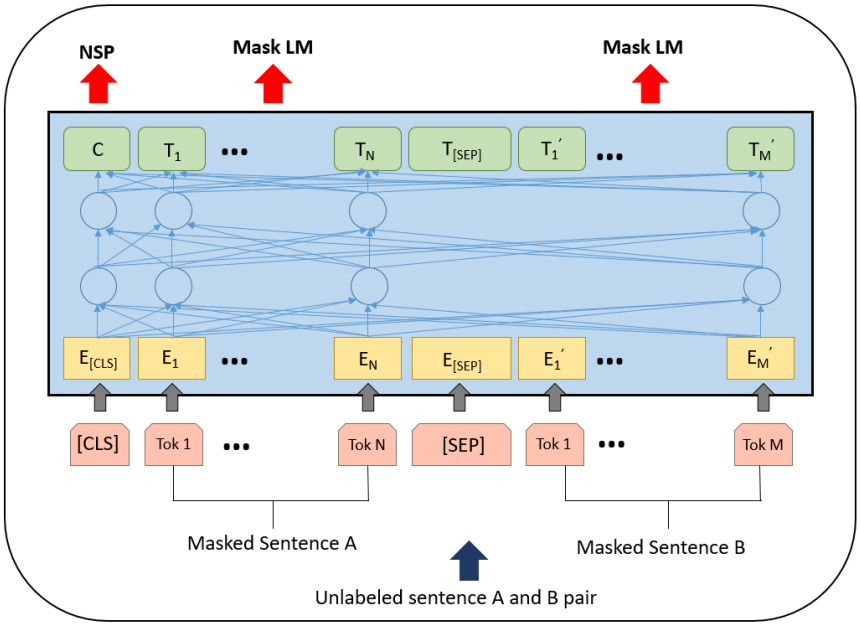


Figure 4.2 BERT architecture (pre-trained)

The arrows in the above image (Figure 4.1) shows the information flow from one layer to the next. The final contextualized representation of each input word which was obtained by tokenizing the sentence is indicated by the green boxes at the top.

As mentioned in the paper 'Attention Is All You Need' the transformer is a model that depends completely on the concept of self-attention without relying on RNNs or CNNs to compute the output. The model will use attention to concentrate on other terms in the input that are closely connected to the word in question.

The model is made up of staked up encoders and decoders. The encoder is made up of a layer of Multi-Head Attention and a Feed Forward neural network. The decoder has an extra Masked Multi-Head Attention Layer also known as encoder-decoder attention layer. The input to the decoder stack is the output of the last encoder layer. The encoder-decoder attention layer combines the two sources of input – the self-attention layer below it and the input of the encoder layer.

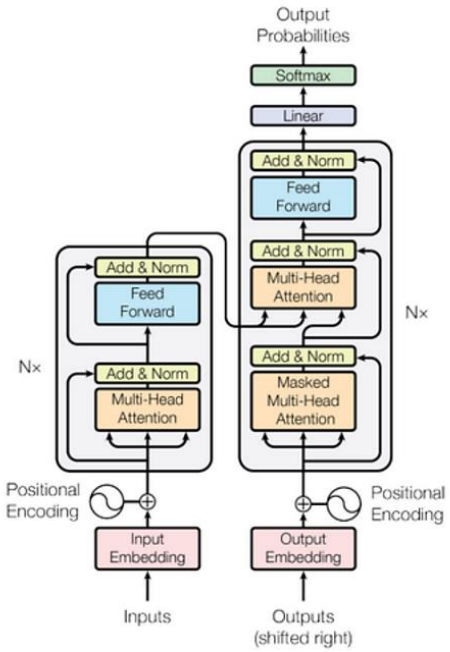


Figure 4.3 Structure of encoder and decoder

4.2.2 Attention in BERT

The input of the attention layer is taken from three parameters: query, key and value. In the transformer the attention layer computes multiple times in parallel. Each computation is termed as an attention head. The query, key and value are split into n-sets with each being passed through a separate head. All the calculations are combined to produce a single attention score. This process is called multi-head attention and allows the transformer to be able to encode multiple relationships for each and every word.

The output of the final encoder in the stack is passed onto the value and key parameters in the Encoder-Decoder attention. This attention computes the interaction between each input word with each target word. In Masked Multi-Head Attention, a part of the output sequence for every parallel operation is hid or masked.

4.2.3 Fine tuning BERT

NER or Named Entity Recognition is an NLP task of being able to locate and classify the named entities that are mentioned in the unstructured text into the pre-defined categories like person names, places, organizations, time etc. By customizing our dataset, our own pre-defined labels can be included.

The BERT model can be modified to perform NER tasks by adding a classification layer that will predict the NER label after feeding the output vector of each token to it.

After downloading the pre-trained model for the library, fine-tuning is done using the AdamW function as the optimizer.

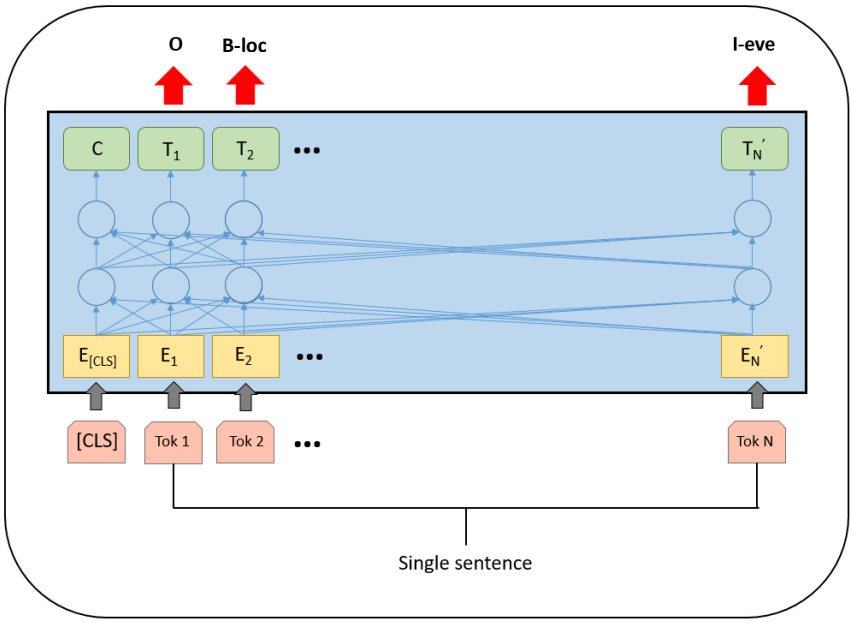


Figure 4.4 BERT architecture for performing NER

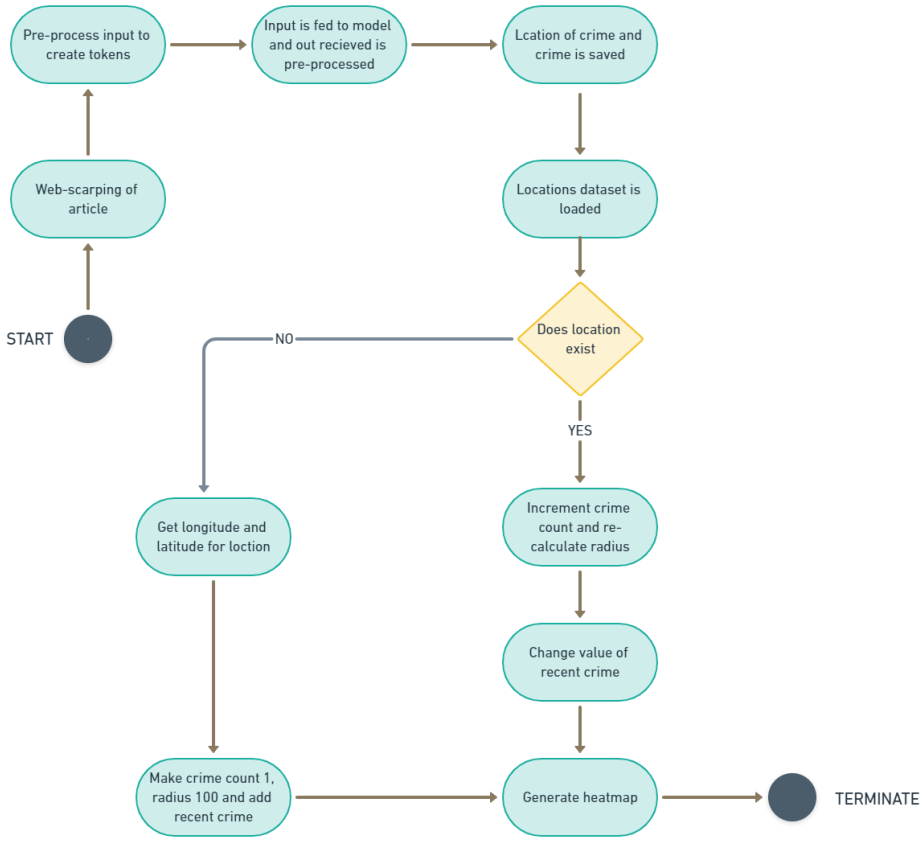


Figure 4.5 Activity diagram of system

5. IMPLEMENTATION

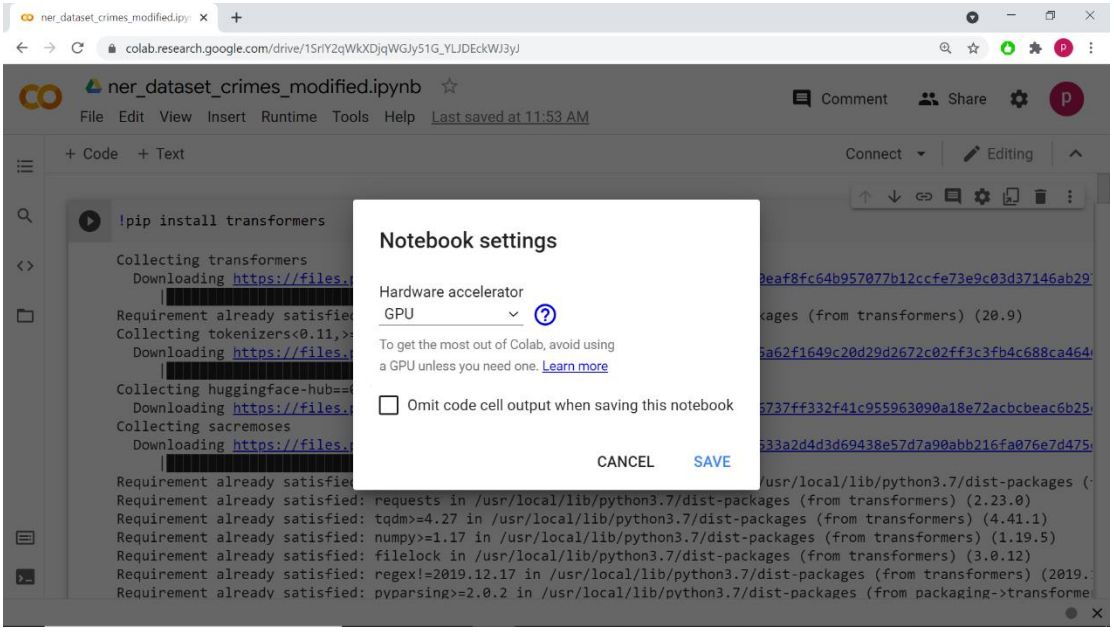


Figure 5.1 Screenshot depicting the hardware accelerator of the notebook

To enable the GPU on the Colab environment go to the edit option. After that choose the notebook settings and select the GPU option in the dropdown for hardware accelerator.

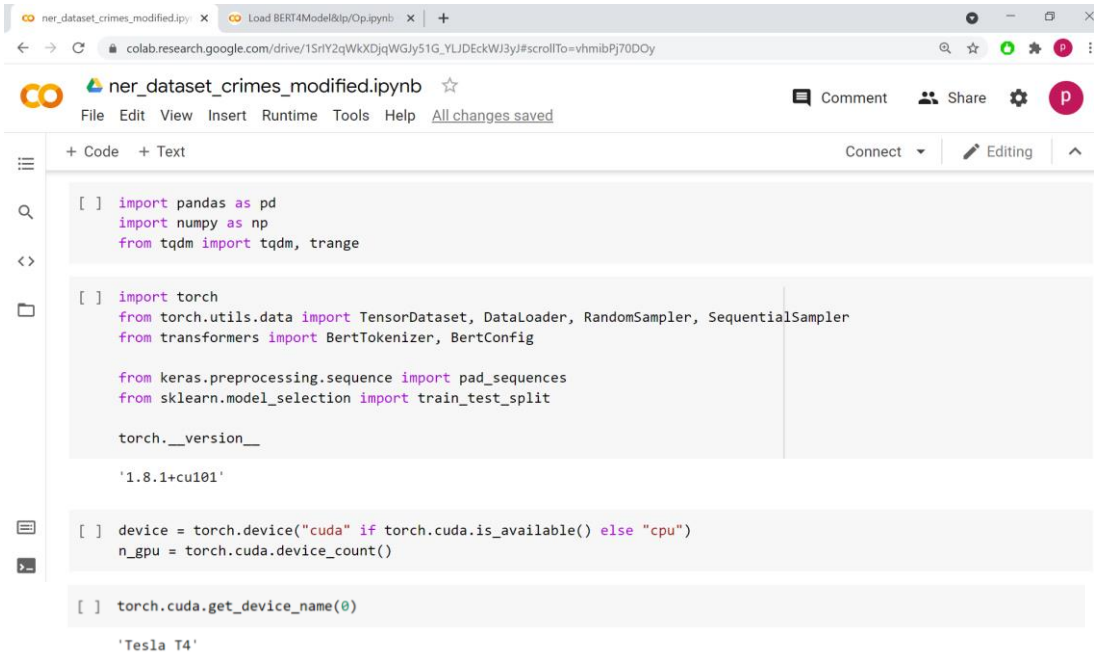


Figure 5.2 Importing important libraries and activating the GPU

The torch.device is an object representing the device on which a torch.Tensor will be allocated. torch.device contains a device type either ‘cpu' or 'cuda' and an optional device ordinal for the device type. In case the device ordinal is not present, the object will represent the current device for the device type.

torch.cuda is the function used to set up and run any CUDA operations available. It keeps track of the currently selected GPU. All the CUDA tensors allocated will be created on that device by default. A torch.cuda.device context manager can be used to change the selected device. A torch.device is an object representing the device on which a torch.Tensor is or will be allocated.

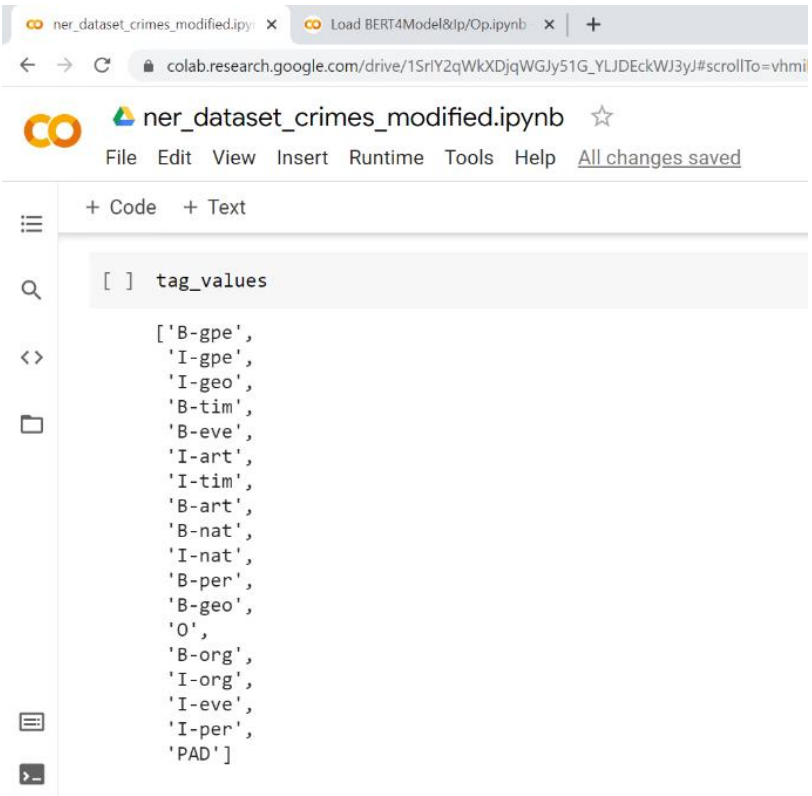


Figure 5.3 Unique tag values existing in the dataset

tag_values is a list of all the unique tags that exist in our dataset. The order of the tags should be maintained when the model is used for inference.

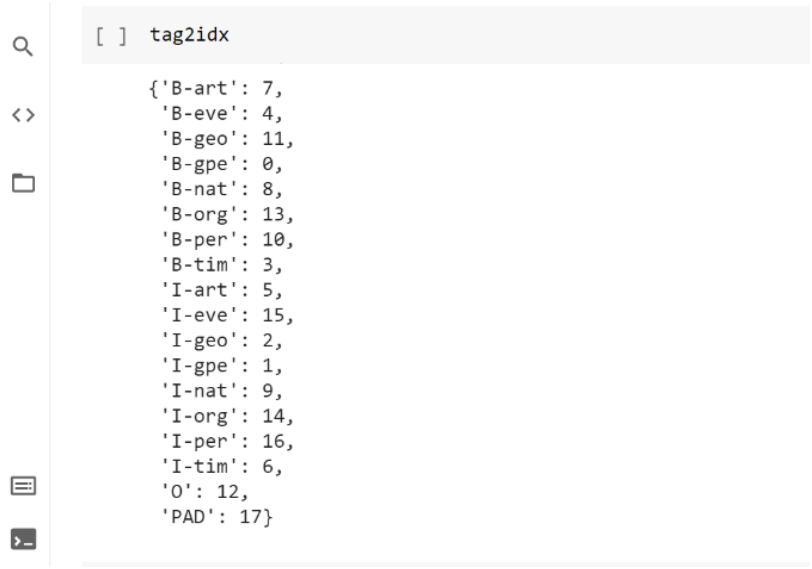


Figure 5.4 Numerical values assigned to each tag

The tag2idx is a dictionary that has index values assigned randomly for all the tags including the PAD tag. Since the aim is not to predict the next sentence indexes for the CLS and SEP tags have not been added.

Tokenization

Tokenization is the process of splitting a phrase, sentence, paragraph, or an entire text document into smaller units like individual words or terms. Each of these smaller units are called tokens.

In the model to tokenize the sentences the BertForTokenClassification() function was used. Given below is the syntax and arguments for the bertTokenizer function. It keeps track of tokenization methods as well as the downloading and loading pre-trained tokenizers.

```
class transformers.BertTokenizer (vocab_file,
do_lower_case=True, do_basic_tokenize=True,
never_split=None, unk_token='[UNK]', sep_token='[SEP]',
pad_token='[PAD]', cls_token='[CLS]',
mask_token='[MASK]', tokenize_chinese_chars=True,
strip_accents=None, **kwargs)
```

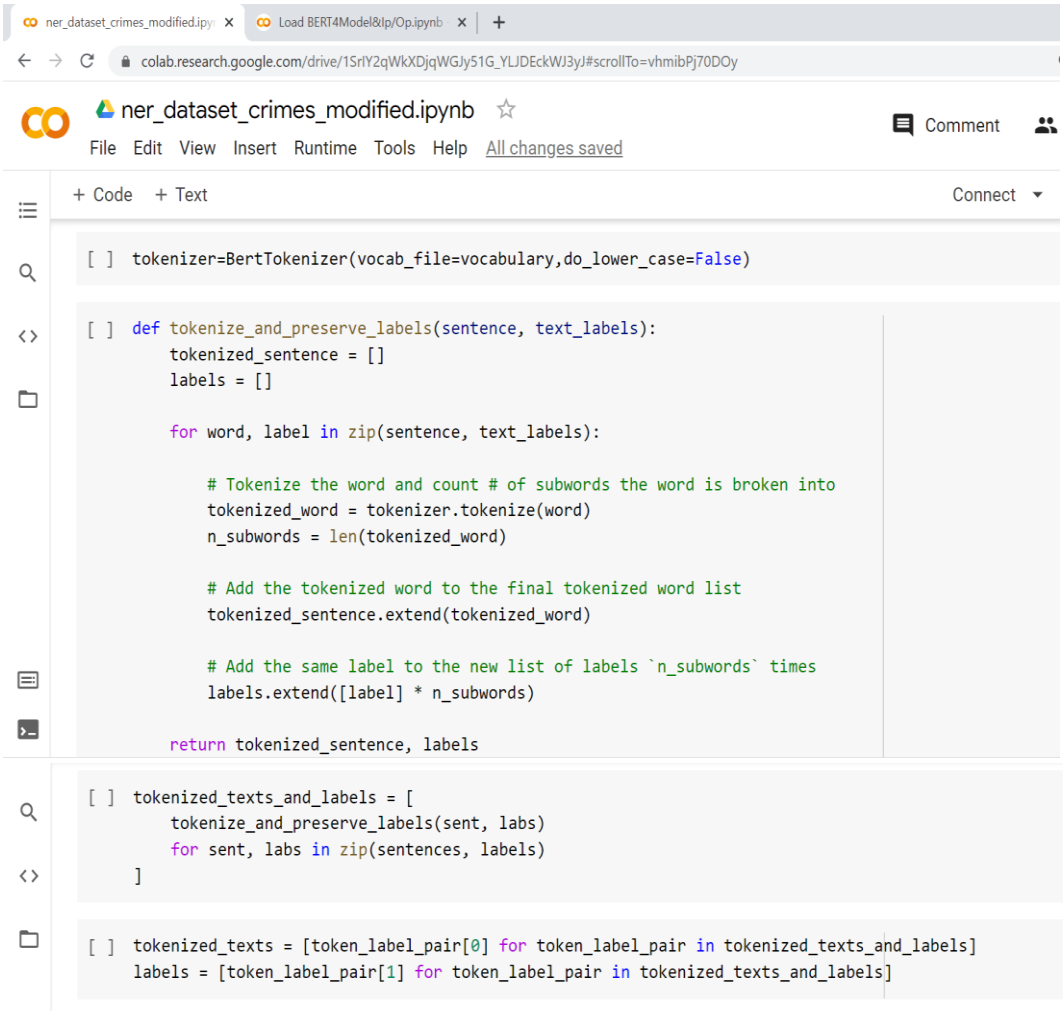


Figure 5.5 function for tokenizing the input sentences

The following are the arguments for the function:

- vocab_file (str) – File containing the vocabulary.
- do_lower_case (bool, optional, defaults to True) – Whether or not to lowercase the input when tokenizing.

The vocabulary file is downloaded from the bert-based-cased pre-trained model. The size of Bert Vocabulary is set to 30K tokens. Subwords and characters are used to represent words that are not in the lexicon.

```
[ ]  input_ids = pad_sequences([tokenizer.convert_tokens_to_ids(txt) for txt in tokenized_texts],
                               maxlen=MAX_LEN, dtype="long", value=0.0,
                               truncating="post", padding="post")

[ ]  tags = pad_sequences([[tag2idx.get(l) for l in lab] for lab in labels],
                          maxlen=MAX_LEN, value=tag2idx["PAD"], padding="post",
                          dtype="long", truncating="post")

[ ]  attention_masks = [[float(i != 0.0) for i in ii] for ii in input_ids]
```

Figure 5.6 creating input_ids, tags and attention_masks for the model

In the above figure the input_ids, tags and attention_masks are generated.

input_ids – have the list of indices of the input sequence tokens in the vocabulary.

attention_mask –the padding tokens are masked so that attention is not performed on these tokens.

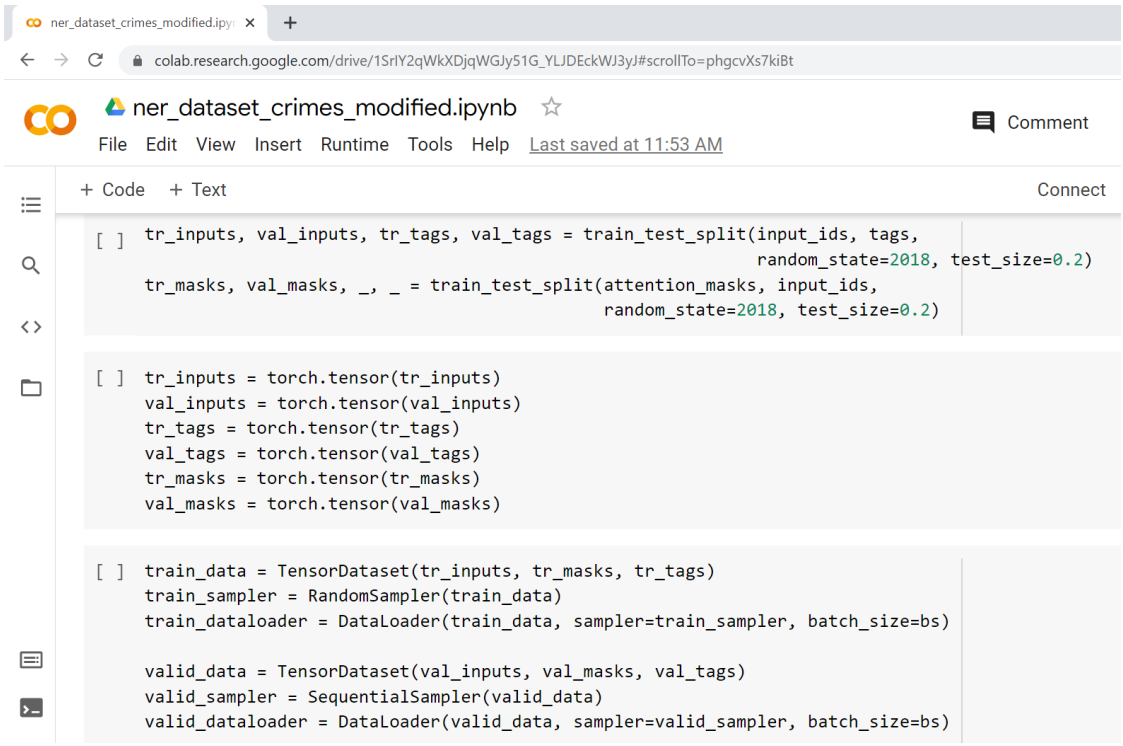


Figure 5.7 Creating inputs, tags and masks for training and testing datasets

The inputs tags and masks are converted into a multi-dimensional matrix using the torch.tensor() function.

The TensorDataset() function creates tensor dataset from a vector containing tensors and the Dataloader() combines the dataset and the sampler to provide an iterable over the resulting dataset.

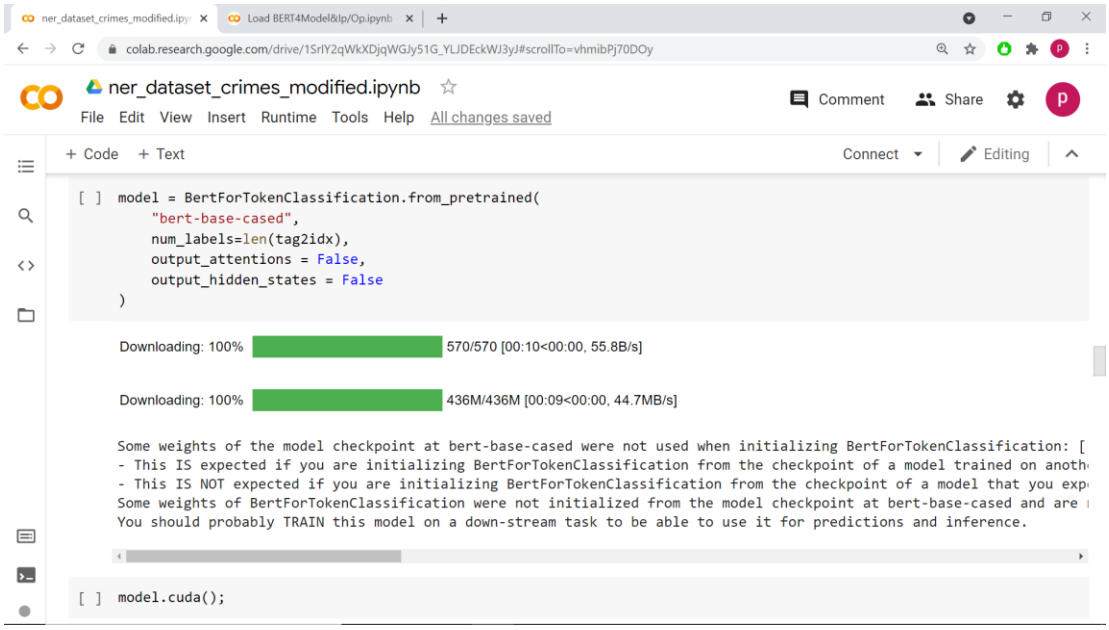


Figure 5.8 Downloading the model and loading it onto the device

To download the pre-trained model the BertForTokenClassification() function is used.

```
classmethod from_pretrained
(pretrained_model_name_or_path: Optional[Union [str,
os.PathLike]], *model_args, **kwargs)
```

The above function is used to instantiate a pre-trained pytorch model from the existing pre-trained model configuration. As a combination of the above two functions, the following function is obtained:

```
BertForTokenClassification.from_pretrained("bert-base-
cased", num_labels, output_attentions = False,
output_hidden_states = False)
```

The following parameter must be be passed through the function

- bert-base-cased – the path for the existing pre-trained model
- num_labels – pass the tag2idx
- output_attentions (bool, optional) – Whether or not to return the attentions tensors of all attention layers.
- output_hidden_states (bool, optional) – Whether or not to return the hidden states of all layers.

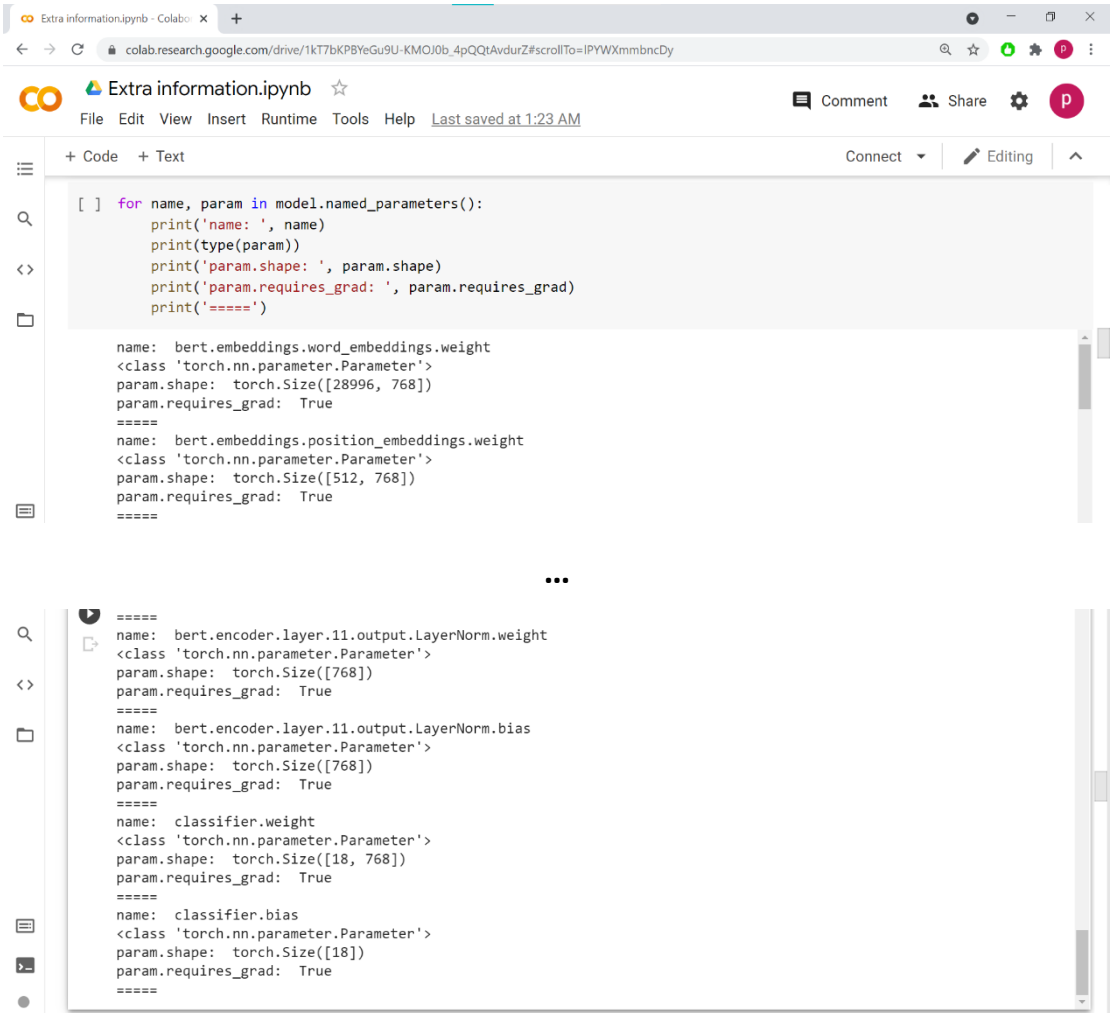


Figure 5.9 Layers in the pre-trained model

In the above figure you get the layer number and the parameters of each layer and their default weight.

The layer numbering starts from 0. The model on a total has 201 parameters.

The param.requires_grad option is TRUE if it is possible to manipulate that specific parameter during trining. It is false if it is necessary to freeze the parameter and no changes must be made during training.

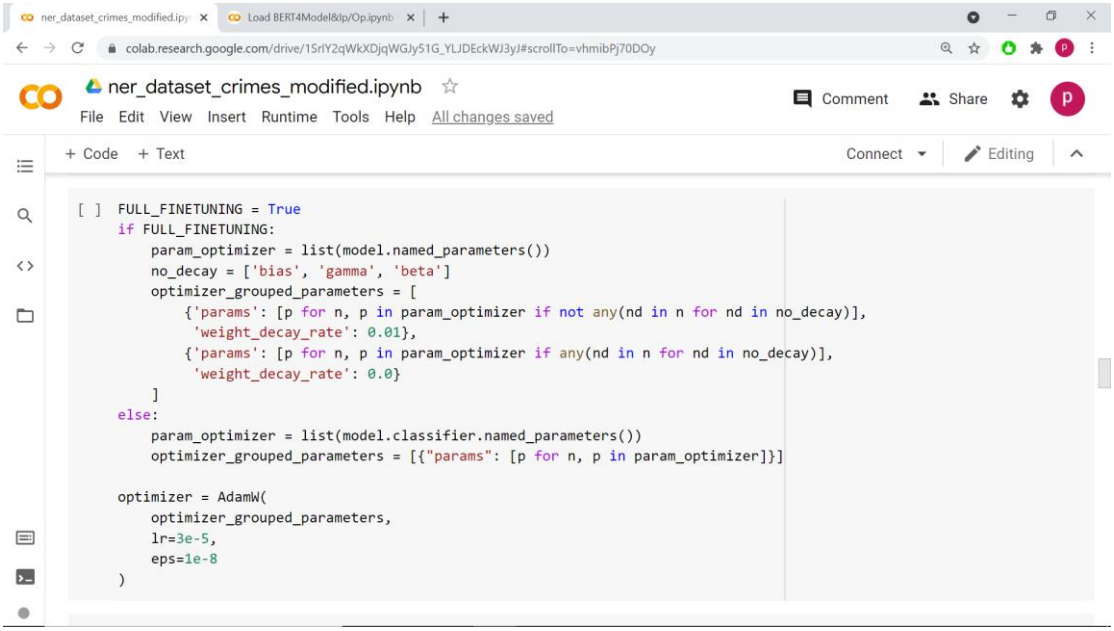


Figure 5.10 Fine-tuning the model and creating optimizer

The pre-trained model has 12 layers. The weights for various layers are obtained and are added into a single list. After that weight parameters are separated into the bias, gamma, and beta parameters.

Following the separation, two groups are created – one having these values and the other one without the values. Each of the groups also has a different decay rate 0.0 and 0.01 respectively. Weight decay is used to prevent the weights from becoming too large after each weight update.

The learning rate and epsilon values in the optimizer are 3e-5 and 1e-8 respectively. The epsilon value is a parameter that is used for numerical stability.

The AdamW is a stochastic optimization method that modifies the implementation of weight decay in Adam, by decoupling weight decay from the gradient update. It adjusts the weight decay term to appear in the gradient update.

```
class transformers.AdamW(params:Iterable[torch.nn.parameter.
Parameter], lr: float = 0.001, betas: Tuple[float, float] =
0.9, 0.999, eps: float = 1e-06, weight_decay: float = 0.0,
correct_bias: bool = True)
```

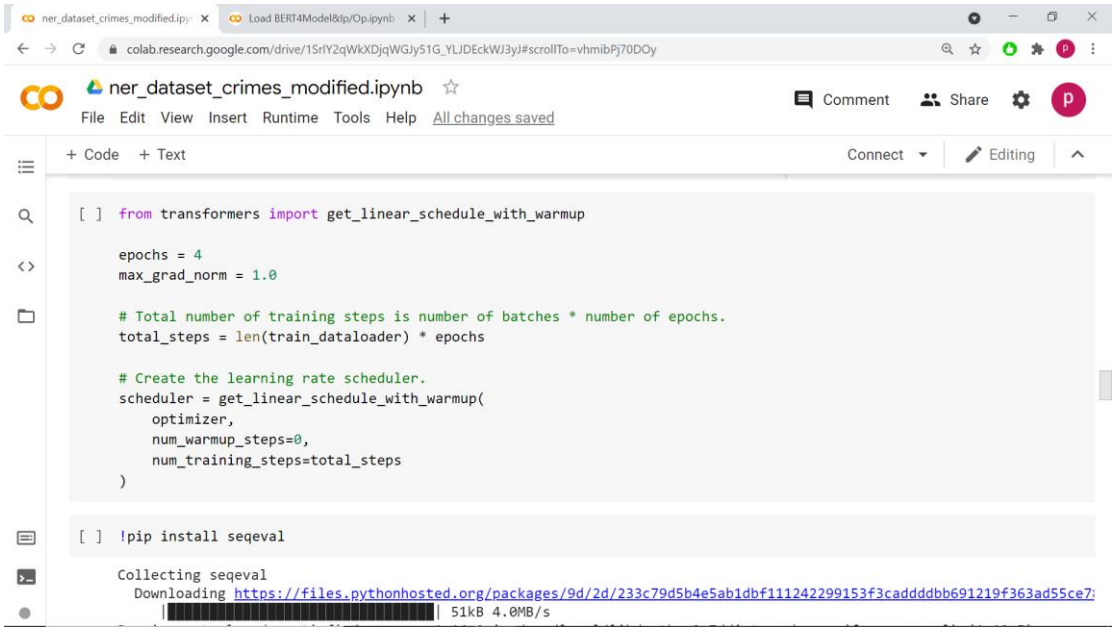


Figure 5.11 Creating a learning rate scheduler

As the number of training epochs grows, it is beneficial to reduce the learning rate. This is predicated on the assumption that a deep learning model with a high learning rate would have a lot of kinetic energy. Hence, a learning rate scheduler is used to reduce the learning rate for each epoch.

This is done using get_linear_scheduler_with_warmup function which uses the previously created optimizer as an argument.

***transformers*.get_linear_schedule_with_warmup**(*optimizer, num_warmup_steps, num_training_steps, last_epoch=- 1)*

- optimizer (Optimizer) – The optimizer for which to schedule the learning rate.
- num_warmup_steps (int) – The number of steps for the warmup phase.
- num_training_steps (int) – The total number of training steps.

Warm up steps is a parameter that lowers the learning rate to lessen the impact of the model deviating from learning when it is exposed to a new data set for the first time.

The max_norm_grad is used to control the learning rate growth.

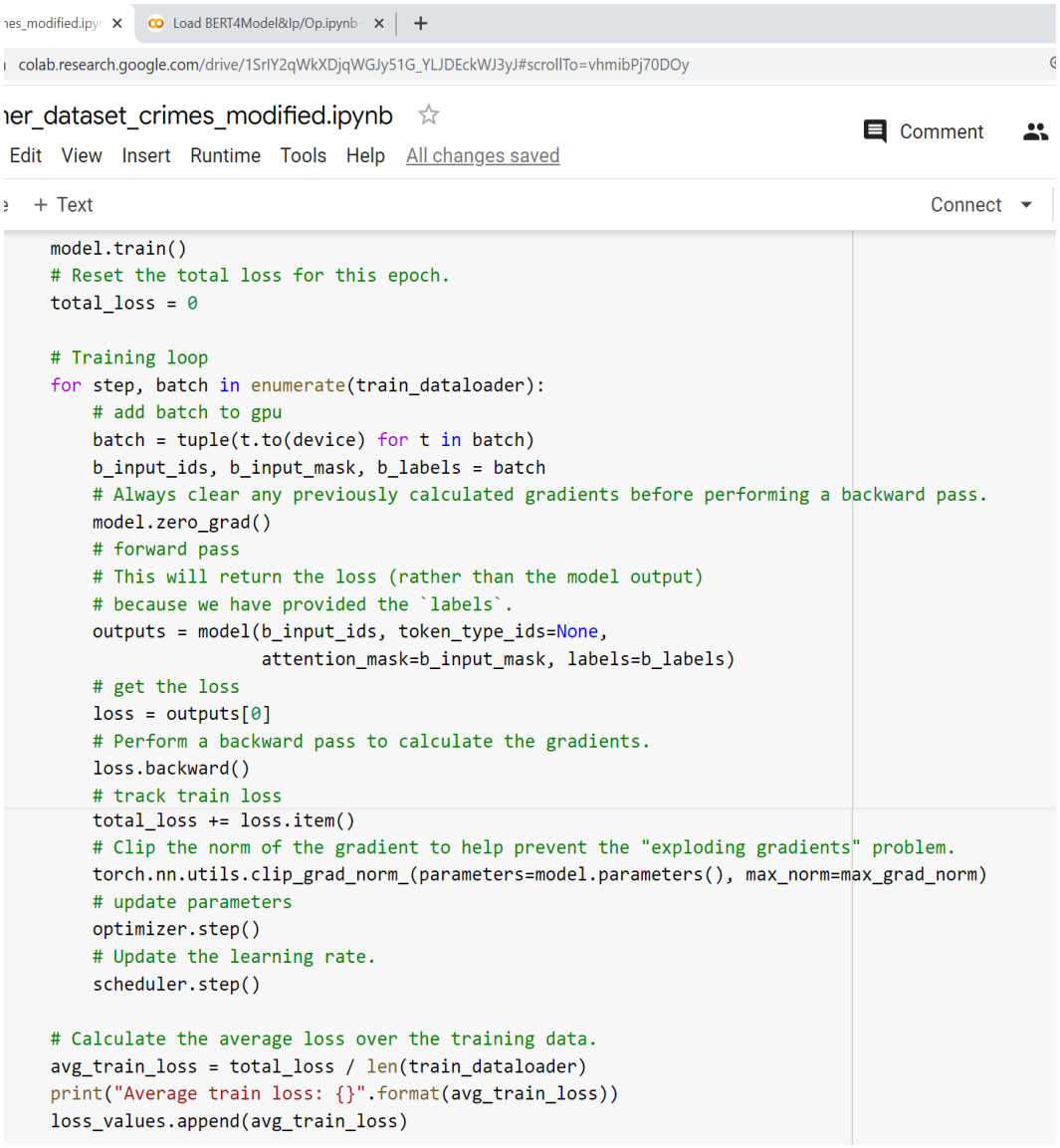


Figure 5.12 Training the model

As seen in the above figure the model is also made to deal with the exploding gradients problem. In a network having n hidden layers the derivatives for each layer i.e. n derivatives will be multiplied. If the derivatives are huge, the gradient will grow exponentially as the model propagates downward until it explodes, this is known as the exploding gradient problem.

To eliminate the above problem in our model gradient clipping has been used. The main aim of gradient clipping is to rescale the gradient if it becomes too large.

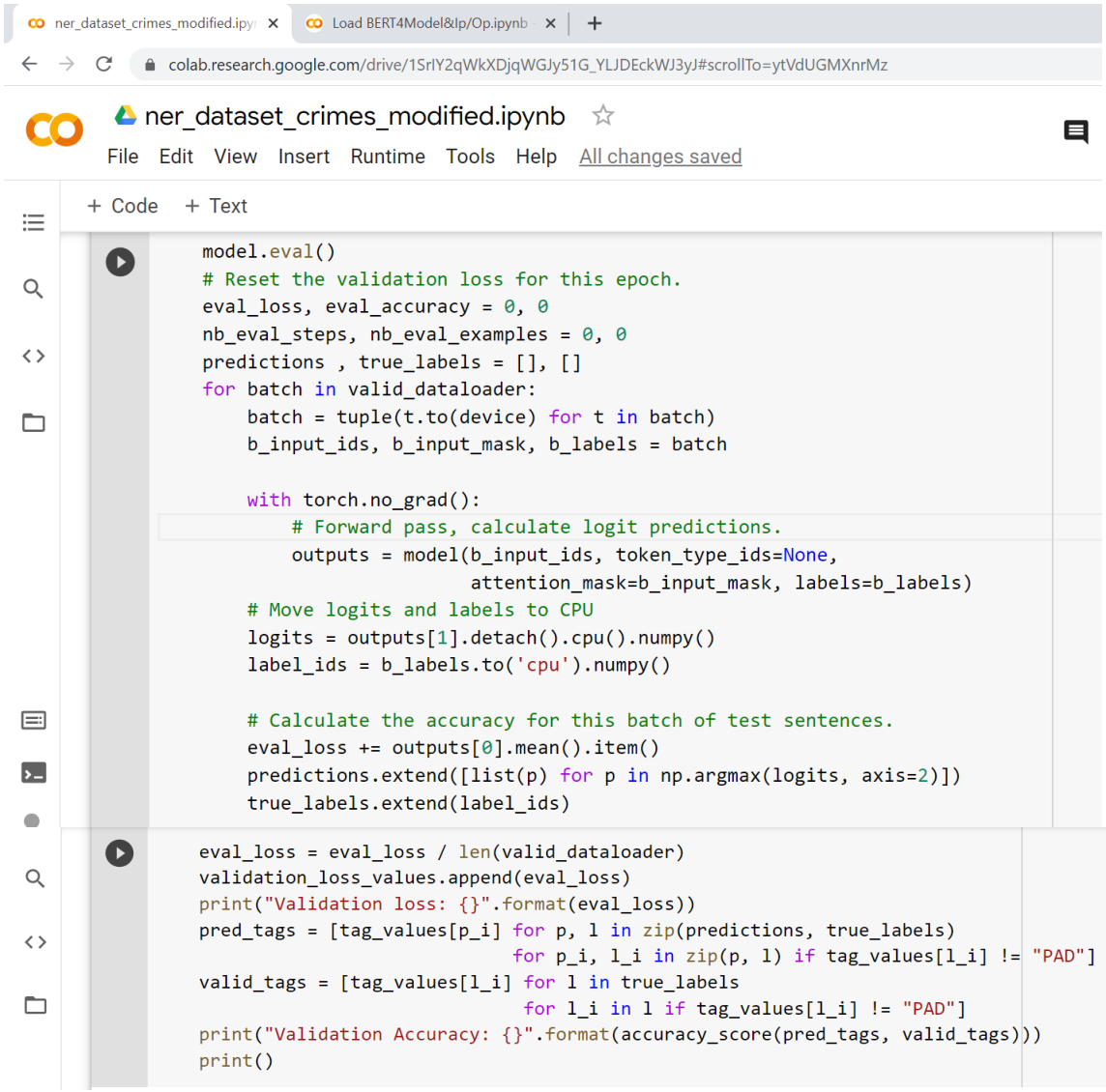


Figure 5.13 Validating the model

Logits are the output of the BERT Model before a softmax activation function is applied to the output of BERT. After passing the input encoding to the BERT Model, the logits can be obtained by specifying output.logits function which returns a tensor.

By applying a softmax activation to this tensor, the probabilistic distributions for every one of the words in BERT's vocabulary is obtained.

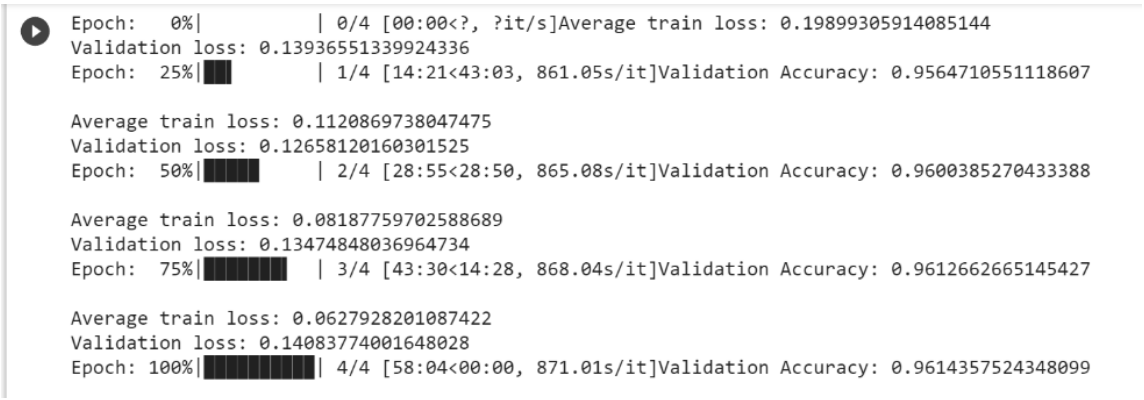


Figure 5.15 Validation for each epoch

Validation accuracy is used to validate the model being built by using the validation dataset which is the data used to validate the generalization ability of the model during the training process.

```
[ ]  pred_tags = [[tag_values[p_i] for p, l in zip(predictions, true_labels)
                                   for p_i, l_i in zip(p, l) if tag_values[l_i] != "PAD"]]
     valid_tags = [[tag_values[l_i] for l in true_labels
                                    for l_i in l if tag_values[l_i] != "PAD"]]
     print("Validation F1-Score: {}".format(f1_score(pred_tags, valid_tags)*100))

     Validation F1-Score: 83.8776123245934
```

Figure 5.14 F1 score

The f1 score is calculated as follows:

$$\mathrm{F1} = 2 * \left(\frac{(P * R)}{(P + R)} \right) \qquad \textbf{5.1}$$

Precision is the number of True Positives divided by the sum of True Positives and False Positives. Precision can be thought of as a measure of a classifiers exactness

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \qquad \textbf{5.2}$$

Recall is the number of True Positives divided by the number of True Positives and the number of False Negatives. Recall can be thought of as a measure of a classifiers completeness

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad \textbf{5.3}$$

Where,

- P - precession
- R - recall
- TP- number of true positives
- FP – number of false positives
- FN – number of false negatives

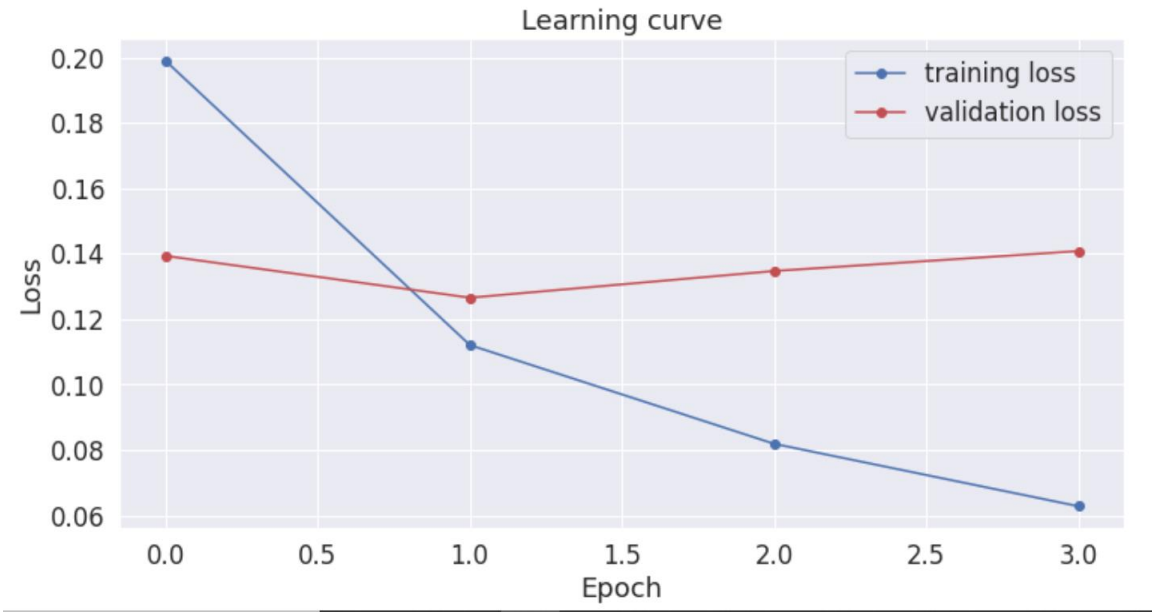


Figure 5.16 Learning curve

The learning curve is plotted between the loss and epoch number. The error on the training set of data is known as training loss. The error after running the validation set of data through the trained network is known as validation loss.

From the graph, it is observed that the second epoch has the least difference between the training and validation loss. Our aim is to make sure that both the loss has the least difference with the training loss to be the lesser value.

```
[ ]  from google.colab import drive
     drive.mount('/content/drive', force_remount=True)

     Mounted at /content/drive

[ ]  model_to_save = model.module if hasattr(model, 'module') else model

[ ]  output_model_file = "./drive/MyDrive/BERT4Model/pytorch_model.bin"
     output_config_file = "./drive/MyDrive/BERT4Model/config.json"

[ ]  torch.save(model_to_save.state_dict(), output_model_file)
     model_to_save.config.to_json_file(output_config_file)
     tokenizer.save_vocabulary("./drive/MyDrive/BERT4Model")

     ('./drive/MyDrive/BERT4Model/vocab.txt',)
```

Figure 5.17 Saving the model and its configuration

To save the model the torch.save() function is used.

Using this function, a serialized object is saved to disc using this feature. It serializes using Python's pickle feature. The function can be used to save models, tensors, and dictionaries of various objects.

A torch.nn.Module model's learnable parameters (i.e. weights and biases) are stored in the model's parameters (accessible with model.parameters()) in PyTorch. A state_dict is a Python dictionary object that is used to map each layer to the tensor of its parameter. Because state dict objects are Python dictionaries, they can be readily saved, updated, changed, and restored, giving PyTorch models and optimizers a lot of modularity.

Given below is the syntax

```
torch.save(obj, f, pickle_module=pickle,
pickle_protocol=2, _use_new_zipfile_serialization=True)
```

- obj – saved object
- f –a string or os.Path like object containing a file name
- pickle_module – module used for pickling metadata and objects
- pickle_protocol – can be specified to override the default protocol

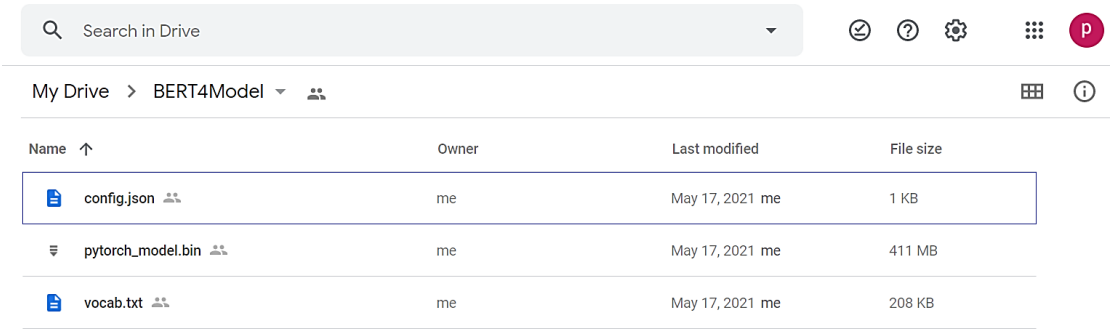


Figure 5.18 Saved model in drive

As shown in the above figure the model is the file with the .bin extension and has a size of 411 MB. The configuration of the model is saved in the config.json file and has all the weights after fine-tuning the model. As a precaution, the vocab.txt file is also saved.

6. RESULTS

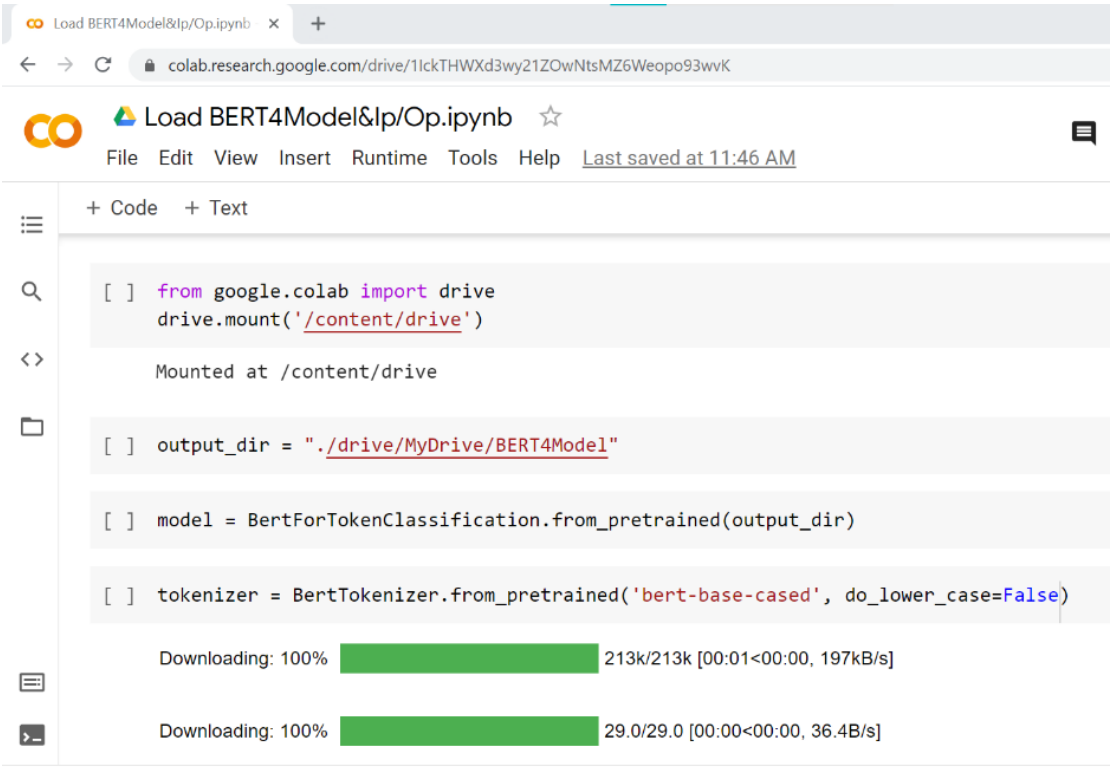


Figure 6.5.1 Loading the pre-saved model

To download the pre-saved model the BertForTokenClassification() function is used.

The above function is used to instantiate a pre-trained pytorch model. As a combination of the above two functions the following function is obtained:

The following parameter must be be passed through the function

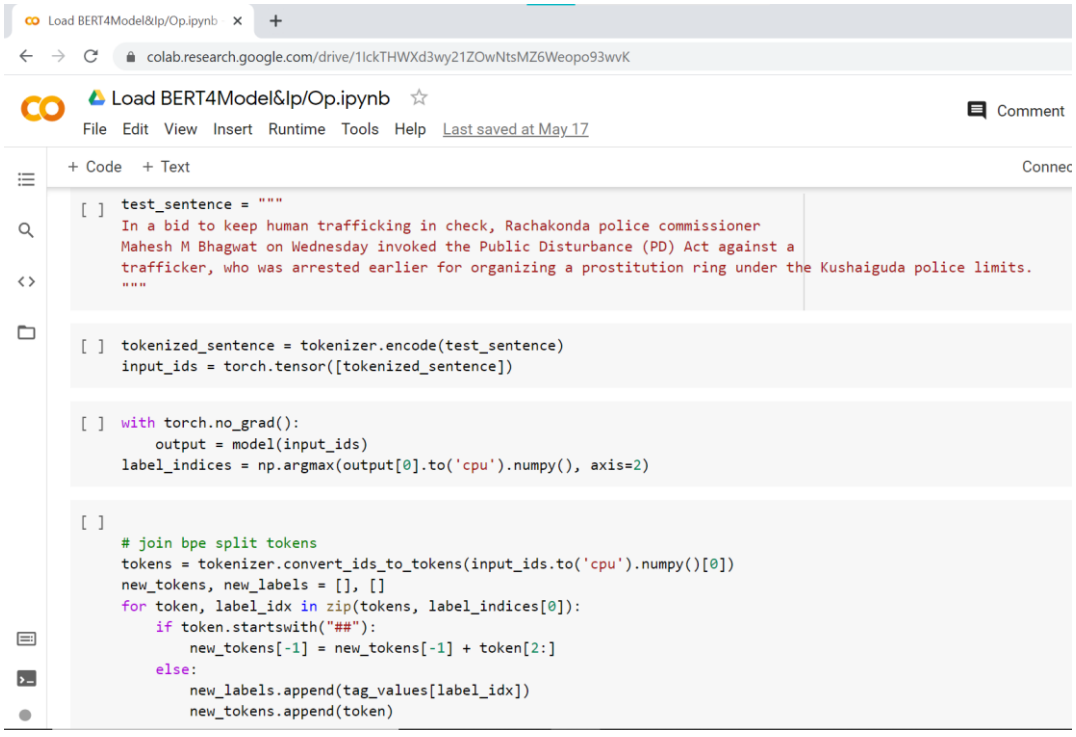
`BertForTokenClassification.from_pretrained`*`(path)`*

The path leads to a folder having the saved model, its configurations i.e. the weigths of the model for each of the parameters of all the 12 hidden layers and classification layer.

```
[ ]  tag_values=['B-gpe',
      'I-gpe',
      'I-geo',
      'B-tim',
      'B-eve',
      'I-art',
      'I-tim',
      'B-art',
      'B-nat',
      'I-nat',
      'B-per',
      'B-geo',
      'O',
      'B-org',
      'I-org',
      'I-eve',
      'I-per',
      'PAD']
```

Figure 6.3 Setting the tag_values before running the model

Tag values are to be saved before running the model as without saving them they are assigned randomly again which fails to produce the expected results.



Load BERT4Model&Ip/Op.ipynb

colab.research.google.com/drive/1IckTHWXd3wy21ZOwNtsMZ6Weopo93wvK

Load BERT4Model&Ip/Op.ipynb

File Edit View Insert Runtime Tools Help Last saved at May 17

Comment

+ Code + Text

Connec

```
[ ] test_sentence = """
    In a bid to keep human trafficking in check, Rachakonda police commissioner
    Mahesh M Bhagwat on Wednesday invoked the Public Disturbance (PD) Act against a
    trafficker, who was arrested earlier for organizing a prostitution ring under the Kushaiguda police limits.
    """
```

```
[ ] tokenized_sentence = tokenizer.encode(test_sentence)
    input_ids = torch.tensor([tokenized_sentence])
```

```
[ ] with torch.no_grad():
        output = model(input_ids)
    label_indices = np.argmax(output[0].to('cpu').numpy(), axis=2)
```

```
[ ]
    # join bpe split tokens
    tokens = tokenizer.convert_ids_to_tokens(input_ids.to('cpu').numpy()[0])
    new_tokens, new_labels = [], []
    for token, label_idx in zip(tokens, label_indices[0]):
        if token.startswith("##"):
            new_tokens[-1] = new_tokens[-1] + token[2:]
        else:
            new_labels.append(tag_values[label_idx])
            new_tokens.append(token)
```

Figure 6.2 Passing input (1) into the model

The torch.no_grad() function is useful the model is to be used for inference. It sould be used only when the torch.backward() function will no longer be called. It save memory and also reduces the computation time.

```
[ ]  t=[]
     l=[]
     for token, label in zip(new_tokens, new_labels):
         t.append(token)
         l.append(label)
         print("{}\t{}".format(label, token))
```

```
O       [CLS]
O       In
O       a
O       bid
O       to
O       keep
B-eve   human
B-eve   trafficking
O       in
O       check
O       ,
B-geo   Rachakonda
B-org   police
O       commissioner
B-per   Mahesh
I-org   M
I-per   Bhagwat
O       on
B-tim   Wednesday
O       invoked
O       the
O       Public
O       Disturbance
O       (
O       PD
O       )
O       Act
O       against
O       a
O       trafficker
O       ,
O       who
O       was
O       arrested
O       earlier
O       for
O       organizing
O       a
B-eve   prostitution
O       ring
O       under
O       the
B-geo   Kushaiguda
B-org   police
O       limits
O       .
O       [SEP]
```

Figure 6.4 Output(1) for input(1)

As seen in the above figure the model is able to recognize police as an organization which will enable us to eliminate the cases where the location of the police station is also mentioned in the article.

```
for i in range(0,len(t)-1):
  if l[i] == 'B-geo' and l[i+1] == 'B-org':
    if t[i+2] == 'limits':
      if l_count == 0:
        l_count = l_count +1
        loc = t[i]
      else:
        l_count = l_count + 1
        loc = loc + " , " + t[i] + " " +t[i+1]
    else:
      i=i+1
    continue
  elif l[i] == 'B-geo' and l[i+1] == 'I-geo':
    if l_count == 0:
      l_count = l_count +1
      loc = t[i] + " " + t[i+1]
    else:
      l_count = l_count + 1
      loc = loc + " , " + t[i] + " " +t[i+1]
  elif l[i] =='B-geo':
    if l_count == 0:
      l_count = l_count +1
      loc = t[i]
    else:
      l_count = l_count + 1
      loc = loc + " , " + t[i]
```

Figure 6.5 Function to process output from model (1)

The above function is used to save the location and crime type from the output of the model.

If the location is followed by a police station the location is not necessary. On the other hand, if the location is followed by police station limits the identified location is the correct one. If the location has a whitespace in between i.e. Banjara Hills the tokens recognized as ‘B-loc’ and ‘I-loc’ are separated with a space.

The two variables ‘cri_count’ and ‘l_count’ are used to make sure that for the second location or crime recognized it will be possible to make a continuous string including the first location or crime.

```
elif l[i] == 'B-eve' and l[i+1] == 'B-eve':
  if cri_count == 0:
    cri_count = cri_count +1
    cri = t[i] + "-" + t[i+1]
  else:
    cri_count = cri_count + 1
    cri = cri + " , " + t[i] + "-" +t[i+1]
elif l[i] == 'B-eve' and l[i+1] == 'I-eve':
  if cri_count == 0:
    cri_count = cri_count +1
    cri = t[i] + "  " + t[i+1]
  else:
    cri_count = cri_count + 1
    cri = cri + " , " + t[i] + " " +t[i+1]
elif l[i] == 'B-eve':
  if l[i-1] == 'B-eve':
    continue
  if cri_count == 0:
    cri_count = cri_count +1
    cri = t[i]
  else:
    cri_count = cri_count + 1
    cri = cri + " , " + t[i]
else:
  continue
```

Figure 6.6 Function to process output from model (2)

Similarly, we make a string of all the crimes committed.

```
[ ]  loc

     'Kushaiguda'

[ ]  cri

     'trafficking , prostitution'
```

Figure 6.7 Location and crime for input (1)

	Location	Number	lat	lon	radius	crime
0	Ameerpet	5	17.437501	78.448251	500	theft
1	Kukatpally	18	17.493084	78.405441	1800	accident
2	vanasthalipuram	8	17.330318	78.568278	800	murder
3	Ambedkar Nagar	17	17.514669	78.500382	1700	roberry
4	Kothapet	1	17.367550	78.555810	100	roberry
5	S.R Nagar	6	17.443120	78.440724	600	theft
6	Attapur	9	17.367224	78.430728	900	kidnap
7	Mehdipatnam	16	17.394263	78.434251	1600	rape
8	Borabanda	17	17.451670	78.415421	1700	smuggling
9	Nalgonda	24	17.053743	79.265870	2400	theft
10	Jubilee hills	9	17.430836	78.410288	900	kidnap
11	sainikpuri	11	17.489215	78.542489	1100	smuggling
12	Ramanthapur	19	17.388504	78.537168	1900	roberry
13	Gandipet	19	17.402291	78.397841	1900	kidnap
14	Khairtabad	8	17.416049	78.461892	800	molest
15	Miyapur	20	17.498161	78.356763	2000	theft
16	Anand Nagar	25	17.479950	78.320553	2500	kidnap
17	Narayanguda	21	17.398116	78.488501	2100	accident
18	Asif Nagar	22	17.383623	78.446144	2200	roberry
19	Ram Nagar	18	17.411423	78.507608	1800	smuggling
20	Bandlaguda	15	17.371488	78.572727	1500	accident
21	Saroor Nagar	17	17.356402	78.531895	1700	roberry
22	Panjagutta	2	17.425424	78.451752	200	kidnap
23	Moula Ali	25	17.466816	78.559216	2500	theft

Figure 6.8 Snapshot of dataset before inserting output(1)

```
if res:
  b=df.loc[df['Location'] == loc, 'Number'].iloc[0]
  b = b+1
  r = 100*b
  df.loc[(df.Location == loc), 'Number'] = b
  df.loc[(df.Location == loc), 'radius'] = r
  df.loc[(df.Location == loc), 'crime'] = cri
else:
  loca = loc+', Hyderabad'
  geolocator = Nominatim(timeout=10, user_agent = "dlab.berkeley.edu-workshop")
  location = geolocator.geocode(loca)
  lat = location[-1][0]
  lon = location[-1][1]
  df2 = {'Location': loc, 'Number': 1, 'lat': lat, 'lon': lon, 'radius': 100, 'crime': cri }
  df = df.append(df2, ignore_index = True)
```

Figure 6.9 Function to check and insert location and crime

In the above figure under the if the location exists the crime count and recent crime is updated. If the location does not already exist the longitude and latitude for the location are extracted using the nominatim package and the added along with location and recent crime as a new row in the data set.

- timeout (int) – time (in seconds), to wait for the geocoding service to respond before raising a geopy.exc.GeocoderTimedOut exception.

```
class geopy.geocoders.Nominatim (*,
timeout=DEFAULT_SENTINEL, proxies=DEFAULT_SENTINEL,
domain='nominatim.openstreetmap.org', scheme=None,
user_agent=None, ssl_context=DEFAULT_SENTINEL,
adapter_factory=None)
```

- user_agent (str) – User-Agent header to send with the requests to geocoder API.

The geolocator.geocode function is used to resolve the location given.

```
geolocator.geocode(location)
```

location (str) – the string for which the location is resolved.

	Location	Number	lat	lon	radius	crime
0	Ameerpet	5	17.437501	78.448251	500	theft
1	Kukatpally	18	17.493084	78.405441	1800	accident
2	vanasthalipuram	8	17.330318	78.568278	800	murder
3	Ambedkar Nagar	17	17.514669	78.500382	1700	roberry
4	Kothapet	1	17.367550	78.555810	100	roberry
5	S.R Nagar	6	17.443120	78.440724	600	theft
6	Attapur	9	17.367224	78.430728	900	kidnap
7	Mehdipatnam	16	17.394263	78.434251	1600	rape
8	Borabanda	17	17.451670	78.415421	1700	smuggling
9	Nalgonda	24	17.053743	79.265870	2400	theft
10	Jubilee hills	9	17.430836	78.410288	900	kidnap
11	sainikpuri	11	17.489215	78.542489	1100	smuggling
12	Ramanthapur	19	17.388504	78.537168	1900	roberry
13	Gandipet	19	17.402291	78.397841	1900	kidnap
14	Khairtabad	8	17.416049	78.461892	800	molest
15	Miyapur	20	17.498161	78.356763	2000	theft
16	Anand Nagar	25	17.479950	78.320553	2500	kidnap
17	Narayanguda	21	17.398116	78.488501	2100	accident
18	Asif Nagar	22	17.383623	78.446144	2200	roberry
19	Ram Nagar	18	17.411423	78.507608	1800	smuggling
20	Bandlaguda	15	17.371488	78.572727	1500	accident
21	Saroor Nagar	17	17.356402	78.531895	1700	roberry
22	Panjagutta	2	17.425424	78.451752	200	kidnap
23	Moula Ali	25	17.466816	78.559216	2500	theft
24	Kushaiguda	1	17.482545	78.572644	100	human-trafficking , prostitution

Figure 6.10 Snapshot of dataset after inserting output(1)

```
[64] test_sentence = """
     Two persons were injured after armed robbers
     opened fire while looting an ATM at Kukatpally.
     Among the injured persons was a security personnel
     who was attacked during the robbery
     """
```

Figure 6.11 Input (2) for the model

```
O       [CLS]
O       Two
O       persons
O       were
O       injured
O       after
O       armed
O       robbers
O       opened
O       fire
O       while
O       looting
O       an
O       ATM
O       at
B-geo   Kukatpally
O       .
O       Among
O       the
O       injured
O       persons
O       was
O       a
O       security
O       personnel
O       who
O       was
O       attacked
O       during
O       the
B-eve   robbery
O       [SEP]
```

```
loc
```

```
'Kukatpally'
```

```
cri
```

Figure 6.12 Output for the input(2) given in fig 6.9

	Location	Number	lat	lon	radius	crime
0	Kushaiguda	1	17.482545	78.572644	100	trafficking , prostitution
1	Ameerpet	5	17.437501	78.448251	500	theft
2	Kukatpally	19	17.493084	78.405441	1900	robbery
3	vanasthalipuram	8	17.330318	78.568278	800	murder
4	Ambedkar Nagar	17	17.514669	78.500382	1700	roberry
5	Kothapet	1	17.367550	78.555810	100	roberry
6	S.R Nagar	6	17.443120	78.440724	600	theft
7	Attapur	9	17.367224	78.430728	900	kidnap
8	Mehdipatnam	16	17.394263	78.434251	1600	rape
9	Borabanda	17	17.451670	78.415421	1700	smuggling
10	Nalgonda	24	17.053743	79.265870	2400	theft
11	Jubilee hills	9	17.430836	78.410288	900	kidnap
12	sainikpuri	11	17.489215	78.542489	1100	smuggling
13	Ramanthapur	19	17.388504	78.537168	1900	roberry
14	Gandipet	19	17.402291	78.397841	1900	kidnap
15	Khairtabad	8	17.416049	78.461892	800	molest
16	Miyapur	20	17.498161	78.356763	2000	theft
17	Anand Nagar	25	17.479950	78.320553	2500	kidnap
18	Narayanguda	21	17.398116	78.488501	2100	accident
19	Asif Nagar	22	17.383623	78.446144	2200	roberry
20	Ram Nagar	18	17.411423	78.507608	1800	smuggling
21	Bandlaguda	15	17.371488	78.572727	1500	accident
22	Saroor Nagar	17	17.356402	78.531895	1700	roberry
23	Panjagutta	2	17.425424	78.451752	200	kidnap
24	Moula Ali	25	17.466816	78.559216	2500	theft

Figure 6.13 Snapshot of dataset after adding output(2)

In the above figure change is seen in the crime count and crime for the location Kukatpally.

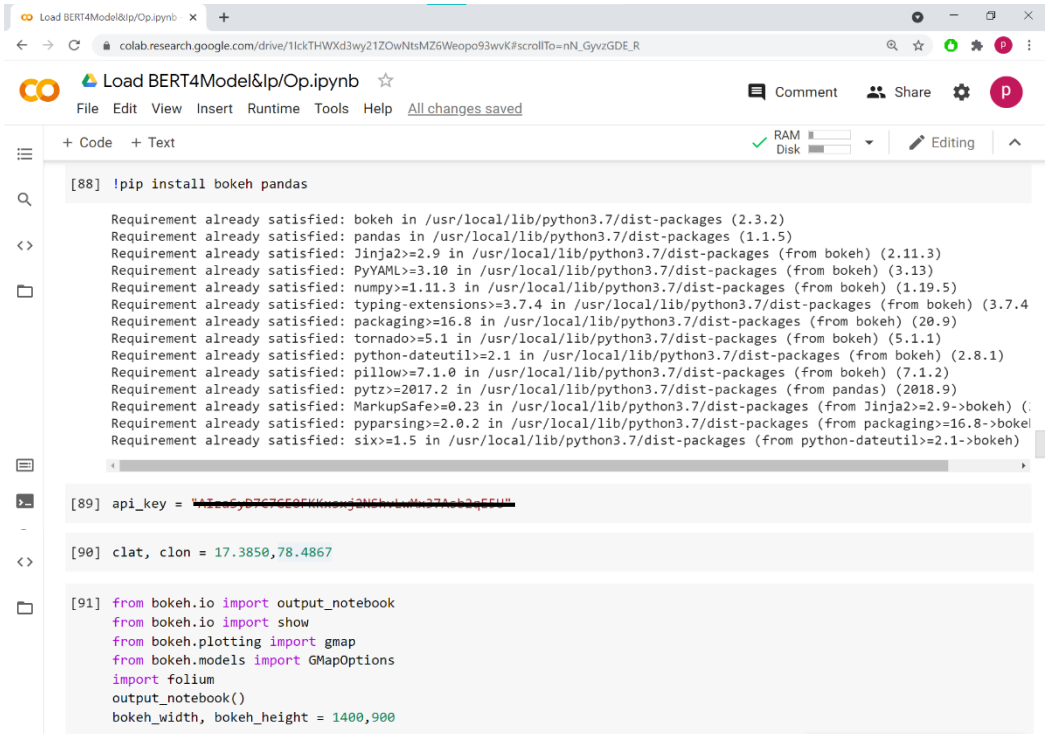


Figure 6.14 Setting API key, central latitude and central longitude

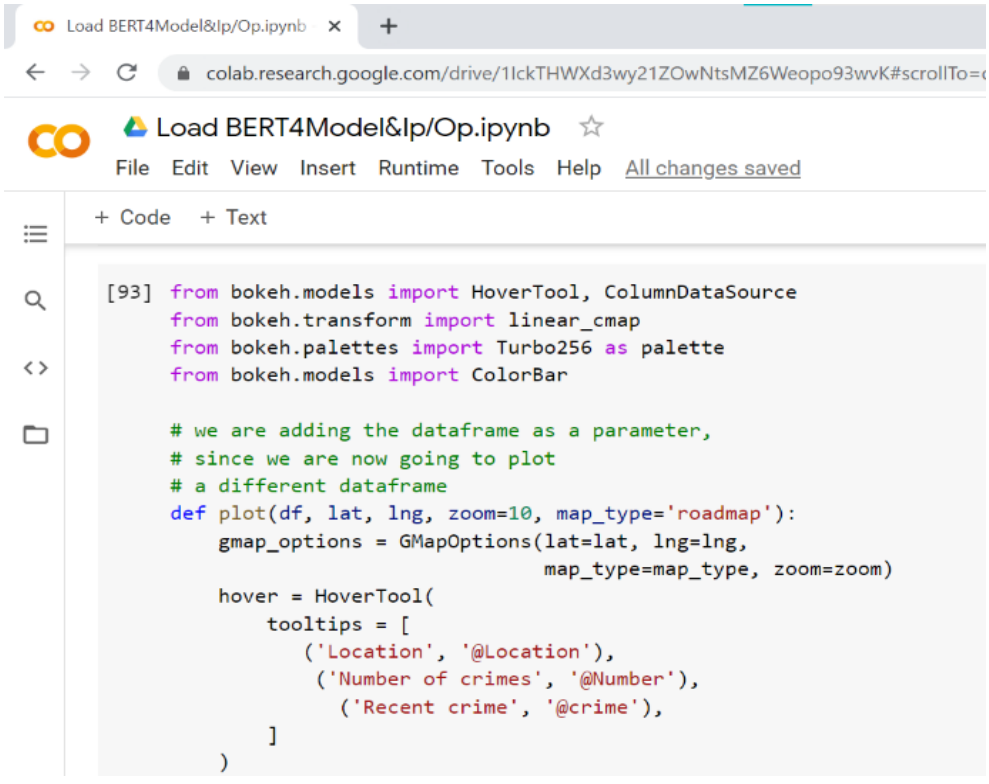
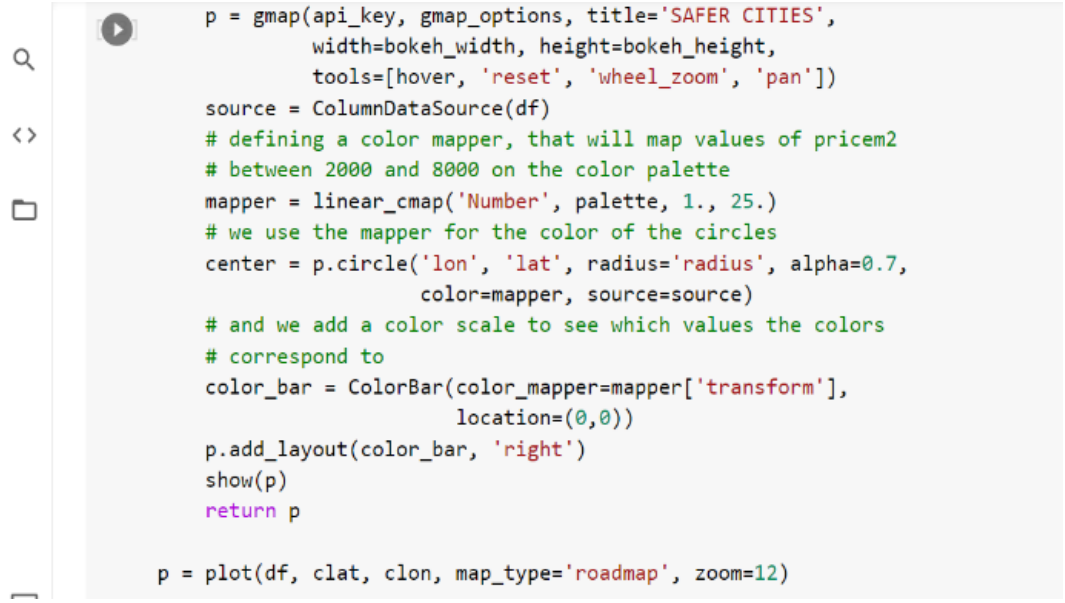


Figure 6.15 Function to plot heat map (1)



```
    p = gmap(api_key, gmap_options, title='SAFER CITIES',
             width=bokeh_width, height=bokeh_height,
             tools=[hover, 'reset', 'wheel_zoom', 'pan'])
    source = ColumnDataSource(df)
    # defining a color mapper, that will map values of pricem2
    # between 2000 and 8000 on the color palette
    mapper = linear_cmap('Number', palette, 1., 25.)
    # we use the mapper for the color of the circles
    center = p.circle('lon', 'lat', radius='radius', alpha=0.7,
                      color=mapper, source=source)
    # and we add a color scale to see which values the colors
    # correspond to
    color_bar = ColorBar(color_mapper=mapper['transform'],
                         location=(0,0))
    p.add_layout(color_bar, 'right')
    show(p)
    return p

p = plot(df, clat, clon, map_type='roadmap', zoom=12)
```

Figure 6.16 Function to plot heat map(2)

to plot the map the GMapoptions() function is used

```
class GMapOptions(*args, **kwargs)
```

- Lat – The latitude along which the map should be centered.
- Lng – The longitude along which the map should be centered
- Map_type – The map type to be used for the GMapPlot
- Zoom – The initial zoom level to use when displaying the map.

To use the google map as the background for plotting the glyphs, the gmap() function is called. As given in the documentation , the function is used to “Retrieve a 'Google Map' that can be used as a background for plotting points and other spatial data.”

```
gmap(x, exp=1, type='terrain', filename='',
style=NULL, scale=1, zoom=NULL, size=c(640, 640),
rgb=FALSE, lonlat=FALSE, map_key, geocode_key, ...)
```

- map_key – Google API key for geocoding
- width – the width of the map to be displayed
- height – the height of the map to be displayed
- tools – the tools which will displayed for easier useage of the map
- gmap_options – the above defined options using the GMapOptions

In Figure 6.14, Modules such as “HoverTools” and “ColumnSourceData” are used to make the graph interactive. The Graph is plotted by initiating an Object of class “GMapOptions” from Bokeh. It creates a google Map option object with the given central longitude, latitude, maptype and zoom options.

Hover Tool enables features such as zoom-in, zoom-out, scroll effect enabling and disabling etc. It uses tool tip to choose columns from given data frame which are to be represented on Map to display relevant information.

A palette is a basic Python array of (hex) RGB colour strings. The module contains a lot of options. Magma, Inferno, Plasma, Viridis, and Cividis are among the Matplotlib palettes used in Bokeh.

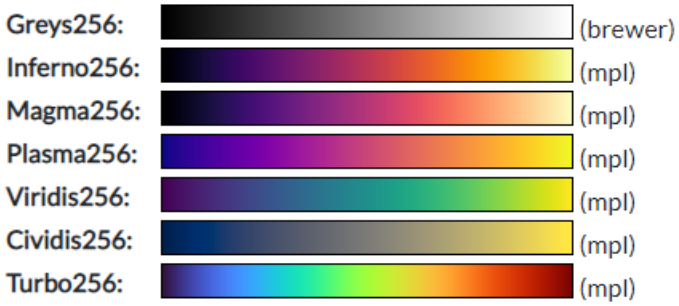


Figure 6.17 Bokeh palette options

The ‘Large palette’ in this module has 256 colors which is useful to differentiate between different locations based on crime rate. “Turbo256” was used in this project as the color range from start to end had suited the context appropriately.

As seen in the above figure the dark red color is used to denote the location with the highest crime rate and dark blue is used to denote the location with least crime rate on the heat map.

The GoogleMapOption object is now passed into gmap which renders a Google Map via Google Map API internally with given options.

A Heat Layer is added using circles and given palette. The radius of the circle and color is determined by the number of crimes as shown below in Figure 6.15.

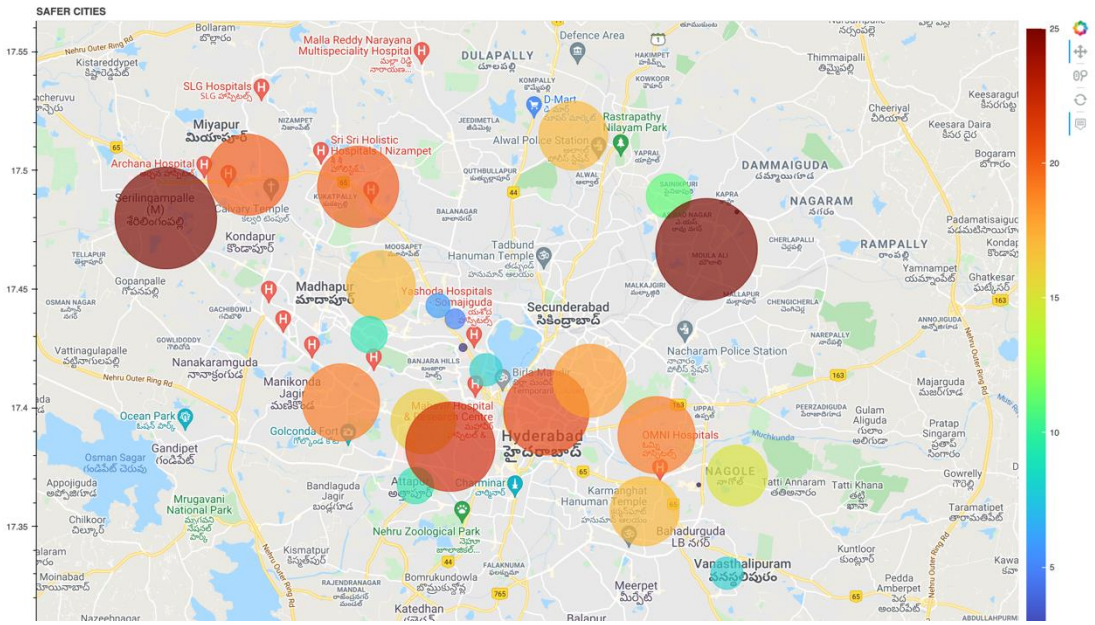


Figure 6.18 Heat Map representing crime locations

On Top right of the Figure 6.15, The Tools to interact with map are displayed. The color bar represents the number of crimes ranging over various colors.

For Example, locations with <5 crimes are marked in shades of blue, a location with crime rate ranging from 10<n<15 where n is number of crimes, are marked with the shades of green and locations with crimes > 20 are marked in shades of red.

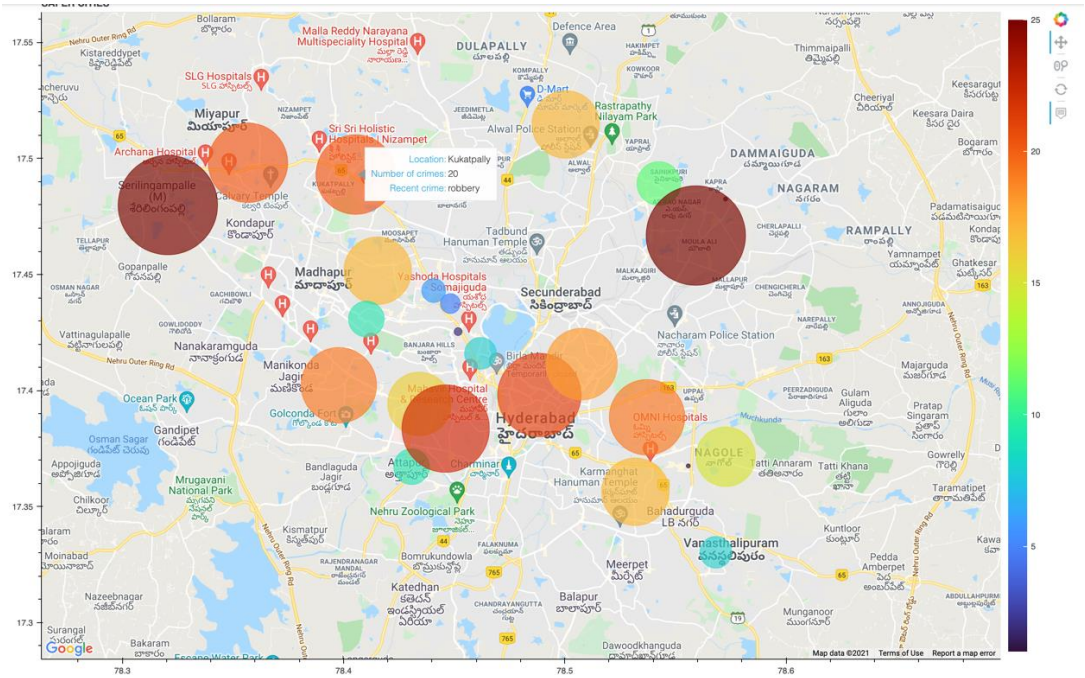


Figure 6.19 Heat Map with Hover Feature

In Figure 6.16, a small information box can be seen when the pointer/ mouse is hovered over a certain location. This shows the Location Name, Total number of crime recorded and the most recent crime.

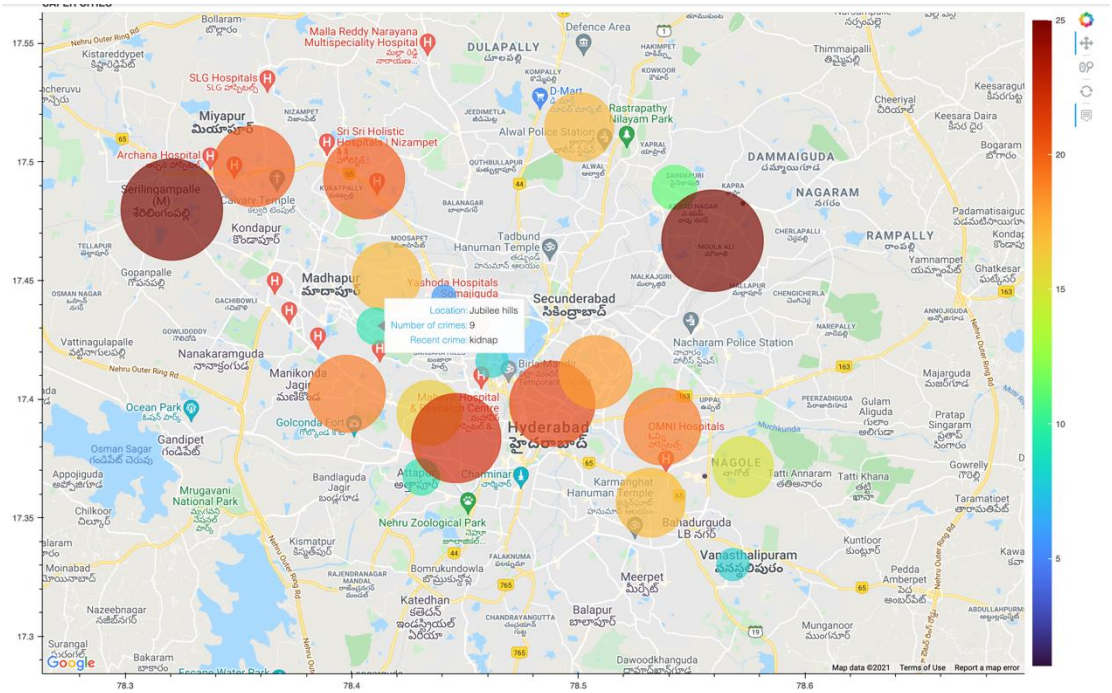


Figure 6.20 Heat Map with Details

7. CONCLUSION AND FUTURE SCOPE

This project was built to help citizens to be aware of the crime rate around them. It will help the users who seek to move into a new location, buy a property or who want to go for a vacation to an unknown area.

The model in this prototype extracts important details such as the location of crime and type of crime using NER and visualizes the processed information on a Heat Map. The model will ensure the citizens safety by giving them a consolidated image of the crimes in a location so they can take the appropriate measures while visiting that place. BERT which is a state of the art model was used in this project to perform the NER tasks.

The Accuracy of the model in this project was greater than any other model in the papers referred. The validation accuracy and F1-score of our model is 96.00% and 83.87% respectively when compared to other models used in our base paper which has an accuracy and f1-score of 93% and 79% respectively.

The training time for the model was also reduced to one hour and 5 minutes after using the Tesla T4 GPU.

The project implements a Heat Map to help users interpret the results more precisely. On the heat map along with the location name the number of crimes for the location along with the recent crime that has happened in that particular location is displayed when the pointer is made to hover over the circle.

FUTURE SCOPE

This implementation of “Visualization of Crime Hotspots by analyzing Crime Newspaper Articles” is a just a protype and can be converted into a full-fledged product for the welfare of the Indian citizens.

The Model can be connected to a backend with database for transactional purposes which will help in deleting older and repeated records. Most committed crime can be added to the Map if the data is stored in a database as the redundancy will be reduced due to several tables. APIs can be developed for querying the parameters in the data frame.

The current model only uses newspapers related to Hyderabad. This can be extended to several cities/states based on availability of authentic and verified local newspaper in English. A Native Mobile App can be created for quick access and user-friendly frontend.

BIBLOGRAPHY

[1] Laura Po; Federica Rollo; "**Building an Urban Theft Map by Analyzing Newspaper Crime Reports**", 2018 13th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP) 2018;

[2] Mehedee Hassan; Mohammad Zahidur Rahman; "**Crime news analysis: Location and story detection**"; 2017 20th International Conference of Computer and Information Technology (ICCIT)

[3] Vignesh Rao; Jayant Sachdev; "**A Machine Learning Approach to classify Articles based on Location**"; 2017 International Conference on Intelligent Sustainable Systems (ICISS)

[4] Alessandro Bondielli; Pietro Ducange; Francesco Marcelloni; "**Location Exploiting Categorization of Online news for Profiling City Areas**"; 2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)

[5] Chi Sun; Xipeng Qiu; Yige Xu; Xuanjing Huang; "**How to Fine-Tune BERT for Text Classification**"; 2019 arXiv preprint arXiv:1905.05583v2

[6] Rexy Arulanandam; Bastin Tony Roy Savarimuthu; Maryam A. Purvis; "**Extracting Crime Information from Online Newspaper Articles";** 2014 Proceedings of the Second Australasian Web Conference

[7] Priyanka Das; Asit Kumar Das; "**Crime Analysis against Women from Online Newspaper Reports and an Approach to apply it in Dynamic Environment**" 2017 International Conference On Big Data Analytics and computational Intelligence (ICBDACI)

[8] S.M. Mazharul Hoque Chowdhury; Zerin Nasrin Tumpa; Fatema Khatun; S.K. Fazlee Rabby; "**Crime Monitoring from Newspaper Data based on Sentiment Analysis**"; 2019 International Conference on System Modeling & Advancement in Research Trends

[9] Samiullah Shah; Vijdan Khalique; Salahuddin Saddar; Naeem A. Mahoto; "**A Framework for Visual Representation of Crime Information**" 2017 Indian Journal of Science and Technology

[10] Linkun Cai; Yu Song; Tao Liu; Kunli Zhang; " **A Hybrid BERT Model That Incorporates Label Semantics via Adjustive Attention for Multi-Label Text Classification"** 2020 IEEE Access

[11] Xu Zhang; Lin Liu; Luzi Xiao; Jiakai Ji; "**Comparison of Machine Learning Algorithms for Predicting Crime Hotspot**s"; 2020 IEEE Access

[12] Jacob Devlin; Ming-Wei Chang; Kenton Lee; Kristina Toutanova; "**BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"** arXiv preprint arXiv:1810.04805

[13] The Vanishing Exploding Gradient Problem in Deep Neural Network; Towards Data Science; https://towardsdatascience.com/the-vanishing-exploding-gradient-problem-in-deep-neural-networks-191358470c11

[14] Local newspaper for crime in Hyderabad - https://www.siasat.com/crime/crime-hyderabad/

[15] Named Entity Recognition NER With Bert In Spark NLP; Towards Data Science; https://towardsdatascience.com/named-entity-recognition-ner-with-bert-in-spark-nlp-874df20d1d77

[16] Creating Own Name Entity Recognition using Bert and Spacy with Tourism Dataset; Analytics Vidhya; https://medium.com/analytics-vidhya/creating-own-name-entity-recognition-using-bert-and-spacy-tourism-data-set-c5ee1c2955a2

[17] Multi-Label Text Classification using Bert The Mighty Transformer; Hugging Face; https://medium.com/huggingface/multi-label-text-classification-using-bert-the-mighty-transformer-69714fa3fb3d

[18] Understanding Transformers NLP State Of The Art Models; Analytics Vidhya; https://www.analyticsvidhya.com/blog/2019/06/understanding-transformers-nlp-state-of-the-art-models/?utm_source=blog&utm_medium=demystifying-bert-groundbreaking-nlp-framework

[19] Bert Explained State Of The Art Language Model For NLP: Towards Data Science; https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

[20] What is Gradient Clipping; Towards Data Science; https://towardsdatascience.com/what-is-gradient-clipping-b8e815cdfb48

[21] Learning Rate Scheduler; Towards Data Science; https://towardsdatascience.com/learning-rate-scheduler-d8a55747dd90

[22] Bokeh Pallets; https://docs.bokeh.org/en/latest/docs/reference/palettes.html?highlight=palettes#module-bokeh.palettes

[23] Bokeh Gmap using python; https://docs.bokeh.org/en/latest/docs/reference/plotting.html? highlight=gmap

[24] Show your Data in a Google Map with Python; The Data Frog; https://thedatafrog.com/en/articles/show-data-google-map-python/

[25] Google API key; https://developers.google.com/maps/documentation/embed/get-api-key

[26] BERT Explained: A Complete Guide with Theory and Tutorial; https://towardsml.com/2019/09/17/bert-explained-a-complete-guide-with-theory-and-tutorial/

[27] BERT Explained – A list of Frequently Asked Questions; https://yashuseth.blog/2019/06/12/bert-explained-faqs-understand-bert-working/