An End-to-End Framework for Automatic Crime Profiling

A Project Report Submitted in Partial Fulfillment of Requirements for the Degree of

Bachelor of Technology

by

Shailendra Kumar Gupta (2016CSB1059)
Shreyanshu Shekhar (2016CSB1060)

Department of Computer Science & Engineering
Indian Institute of Technology Ropar
Rupnagar 140001, India
June 2020

Abstract

Much effort is being made to ensure the safety of people. One of the main requirements of travellers and city administrators is the knowledge of places that are more prone to criminal activities. To rate a place as a potential crime location, we need the past crime history at that location. Such data is not easily available in public domain, however, it floats around on the Internet in the form of newspaper and social media posts, in an unstructured manner though. Consequently, a large number of works are reported on extracting crime information from news articles, providing piecemeal solutions to the problem. In this work we complement these work by building and end-to-end framework for crime profiling of any give location/area. While customize individual components of the framework and provide a spatio-temporal integration of crime information. We develop an automated framework which crawls online news articles, analyzes them and extracts relevant information to create crime knowledge base which gets dynamically update in real-time. The crime density can be easily visualized in the form of a heat map which is generated by our knowledge base. As a case study, we investigated 345870 news articles published by 6 daily English newspapers collected over a period of approximately two years. Experimental results show that our crime profiling matches with the ratings calculated manually by various organizations.

Acknowledgements

We are grateful to our supervisors, Dr. Neeraj Goel and Dr. Mukesh Kumar Saini, for their motivation, guidance and patience throughout our work.

Further We would like to thank CSE Department for providing us with facilities required for the project.

Last but not the least, we would like to thank our friends and batchmates for their support and many valuable ideas.

Honor Code

We certify that we have properly cited any material taken from other sources and have obtained permission for any copyrighted material included in this report. We take full responsibility for any code submitted as part of this project and the contents of this report.

Shailendra Kumar Gupta (2016CSB1059)

Shreyanshu Shekhar (2016CSB1060)

Certificate

It is certified that the B. Tech. project "An End-to-End Framework for Automatic Crime Profiling" has been done by Shailendra Kumar Gupta (2016CSB1059), Shreyanshu Shekhar (2016CSB1060) under my supervision. This report has been submitted towards partial fulfillment of B. Tech. project requirements.

Dr. Neeraj Goel
Project Supervisor
Department of Computer Science & Engineering
Indian Institute of Technology Ropar
Rupnagar-140001

Dr. Mukesh Kumar Saini
Project Supervisor
Department of Computer Science & Engineering
Indian Institute of Technology Ropar
Rupnagar-140001

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The information of the probability of having crime at a place is essential for travellers, investors and private companies. Police and few government agencies have this information in bits and pieces. However, it is generally not available in the public domain. Besides, the information is mostly static, and the changes are incorporated manually much later after the crime events have occurred, so we can rely much on such data as they may not be accurate or up to date. Due to such situation and unavailability of crime information, nowadays safety became an important issue. We even have different safety apps available in India to handle safety issues at different, but these are not much helpful as most them depends on users feedback/rating about a place, which may not be updated plus many of us don't give importance of giving feedback to such apps until we face some crime issue. Also we have police forces available at our near stations but it they may not be able to reach us in time always. Moreover, due to insufficient crime knowledge of different locations, we may make unsafe plans or decisions which we may have to suffer in the coming future. So we thought of finding a solution to this problem by asking us, What if we can know the real-time crime rate of every location without going there? This awareness about the status of crime will help us identify safe and unsafe places around us so that we can plan accordingly.

In this paper, we propose a framework to model and dynamically calculate crime score of a given location, where location can be a colony, city or a district. The core idea of our approach is to exploit information present in the news articles published in different newspapers on the Internet. We extract date, place

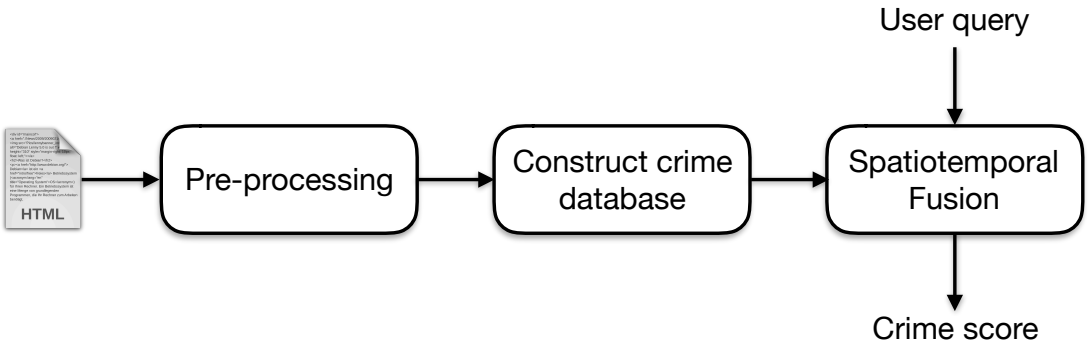

Figure 1.1: Overview of the proposed work. After crawling the articles from the Internet, we pre-process them to separate original crime articles. These crime articles are thn analyzed to build a crime database, which can be queried by users to obtain crime score of a location or a region.

and type of crime from each article, and build an accumulative crime score of a location. The accumulative score of a location depends on the number of crime, their severity and date. The resultant score is dynamic; that is, it changes with time. In this way, we present a real-time crime rate/score of different locations in India. We are first focused on some major cities of India namely Delhi, Mumbai, Hyderabad and Bangalore. This knowledge will help them to plan their trips accordingly. Also, we can combine the result of this framework with Google Maps or any other routing apps which can help users to take the safest path to their destination, which otherwise may lead them users to an unsafe path. This can also provide sufficient information to police forces so that they can accordingly disperse themselves in crime-prone areas.

Several challenges need to be addressed to extract crime information from news articles. First, we have to separate crime-related news articles from other types. Once we have a crime article, we need to identify the location and time of the crime. Extracting date, place and type of crime from news articles is challenging because the information in news articles is context-specific and may not be complete. Further, multiple-meaning of the same word, multiple news articles on the same incident and ambiguity between places, names and organizations makes the extraction more challenging. It is also possible to have more than one news articles of the same crime; hence detecting duplicate articles is essen-

tial. In this work, we provide solutions to these individual problems and finally integrate these solutions to build a framework to calculate crime density. This framework requires two sets database, first one is to store all the crawled articles (raw data) and the second one is to store all the processed data like locations, Geo-coordinates, crime score, crime type, etc. We collected 345870 news articles published by six daily newspapers on the Internet over approximately two years. To demonstrate the efficacy of the proposed solution, we take a case study of the city of Delhi, and the results are promising.

1.1 Background and Related Work

Only a few disparate research attempts have been been done in the past to extract relevant information from online available news articles. Arulanandam et al. [2014] propose a method to detect crime location in theft related news articles. After detecting the location using NER, the authors use conditional random fields (CRF) to classify whether the detected location is related to the crime or not. The method is evaluated only on 70 theft related articles in New Zealand. Jayaweera et al. [2015] analyse news articles published in Sri Lanka to extract crime information. The articles are classified as crime or no crime using an SVM classifier trained on TF-IDF representation of the articles. Location is extracted using NER. To find duplicate articles, the authors calculate simhash value of the entities extracted from the article. The overall goal of the work is to build a database handler that supports viewing crime statistics on a map.

The crime information is also available with various government organizations. Tayal et al. [2015] use KNN based data mining technique cluster data according to crime type and detect criminal attributes. Although this information is more reliable, the information extracted from the news articles is more up-to-date. Joshi et al. [2017] take a supervised approach to detect different types of crimes like thefts, homicide and various drug offences in a crime dataset of North Wales region, Australia. Yadav et al. [2017] also implement a system to derive state-wise crime statistics by analyzing government records of 14 years.

Researcher have explored crime analysis in various other safety related applications as well. Sharma et al. [2015] also transform the documents into a lower

dimensional space and apply KNN to detect creme related articles. KNN requires a large number of labelled documents, also, it is not scalable with the number of articles. The author employ NER based method to detect locations in the document and calculate crime score as the crime count. This crime score is used to find safe path between two points. Similarly, Goel et al. [2017] use the crow-sourced safety score to find safe routes. Hassan and Rahman [2017], find crime articles using SVM classifier and then cluster these articles to group according to the crime. Documents in each cluster are used to build a crime story. NER is used to find location term. The crime locations are found by classifying the whole sentence as crime or no crime, using SVM classifier.

Work	**Focus**	**Crime Article**	**Crime Type**	**Duplicate detection**	**Location detection**	**Location classification**	**Crime score**
Arulanandam et al. [2014]	Theft detection	No	No	No	Yes	No	No
Jayaweera et al. [2015]	Database handler	Yes	No	Yes	Yes	No	No
Tayal et al. [2015]	Data mining	No	No	No	No	No	No
Sharma et al. [2015]	Safe navigation	Yes	No	No	No	No	Yes
Hassan and Rahman [2017]	Crime story	Yes	No	Yes	Yes	Yes	No
Joshi et al. [2017]	Crime classification	No	Yes	No	No	No	No
Proposed	**Crime density**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

Table 1.1: Comparison with the related work.

Table 1.1 shows a summary of the related work. We can see that there has been only piecemeal work on crime density calculation. We have integrated and

customized these works to develop and end to end framework, which is currently running. In terms of individual components, we have improved document classification and location extraction. In terms of novelty, we are the first to consider spatio-temporal fusion to calculate crime score.

Work	**Focus**	**Crime Article**	**Crime Type**	**Duplicate detection**	**Location detection**	**Location classification**	**Crime score**
Arulanandam et al. [2014]	Theft detection	No	No	Not done	Using NER	No	No
Jayaweera et al. [2015]	Database handler	SVM + TF-IDF	No	simhash	Using NER	No	No
Tayal et al. [2015]	Data mining	No	No	No	No	No	No
Sharma et al. [2015]	Safe navigation	SVD + KNN	No	No	NER + LDA	No	Yes (crime count)
Hassan and Rahman [2017]	Crime story	SVM + TF-IDF	No	NER	SVM	No	No
Joshi et al. [2017]	Crime classification	No	K-means	No	No	No	No
Proposed	Crime density	Yes*	Yes*	Yes*	Yes*	Yes*	Yes*

Table 1.2: Comparison with the related work.

1.2 Experimental Setup and Dataset

We have selected six popular English news websites of India, given in Table 1.3. The crawler is running daily to collect new articles from these websites since May 2018. Title, body, date, time and URL of each article is stored in a database. Our database has around 345870 news articles (both crime and non-crime) collected over 2 years. The break-up of these articles, which are used for analysis, is shown

Source	Articles	Time Duration
TOI - Times of India	124623	Sep 18 - Nov 19
Hindustan Times	35562	Jan 18 - Nov 19
India Today	33960	Jan 18 - Nov 19
The Hindu	85035	Dec 18 - Nov 19
News18	36232	Dec 18 - Nov 19
NDTV	30036	March 18 - Nov 19

Table 1.3: Popular News Websites

in Table 1.3. The collected data was unlabelled. For verification and accuracy studies, we labelled part of the data. We labelled around 1400 articles for ground truth regarding crime type, location, and date. Around 50% of the labelled articles were crime related. The remaining data is used for verification of the framework. More details discussed during evaluation.

1.3 Methodology

The objective of this work is to create an automated framework that can find a reliable crime score for all possible locations by extracting information from newspaper articles. Fig 1.1 shows the flow of overall method. The proposed method consists of three main steps. The first step is pre-processing. In this step, we crawl a news article from the Internet, determine whether or not it is a crime related article, and check for existing duplicate articles. From each original crime article, in the second step, we extract location and crime type information and build a crime database. Finally, in the third step, we calculate the final crime score according to the user query using spatiotemporal information fusion. To make a heatmap of crime, the system can be queried repeatedly. In this section we provide technical details of each block.

1.4 Our Contributions

Following are the main contributions of our the proposed work:

- First end-to-end framework for automatic crime score calculation from news articles. The framework provided crime ratings are in sync with manual agency ratings.
- A more granular crime profiling of articles including crime detection and crime type weighing.
- A spatiotemporal fusion framework to calculate overall crime score of locations, independent of whether or not crime is reported at that place.
- Not all locations extracted from the crime article are related to the crime. We propose a novel method to classify a location as crime or no crime location.

Chapter 2

Pre-processing

We start with the collection of news articles from online news websites. Each article is classified as crime or non-crime based on the body and title. Crime articles are further classified are original or duplicate. Only original articles are added into the crime database. Duplicate articles are rejected. The overview of the pre-processing step is given in Fig. 2.1.

2.1 Crime Article Detection

Intuitively, if crime words appear in an article, it can be classified as crime-article. However, due to context dependent multiple meanings of the same word, this process becomes very difficult. We first explain a baseline method and then an improved version being used in our system.

2.1.1 Baseline Method: Crime word-based Classification

The input of the algorithm is the *articleText* and *crimeList*, a list of all possible crime words. We have also assigned severity score for each word based on the seriousness of the crime. Example murder will have a high severity score than theft. Algorithm 1 shows the details of the steps. From *crimeList*, first, we create *synonymList*, a list of all related words using NLTK library. Wu and Palmer [1994] algorithm is used to find similarity. Modified severity score for a word is multiplication of similarity score and severity score of the main word. Article

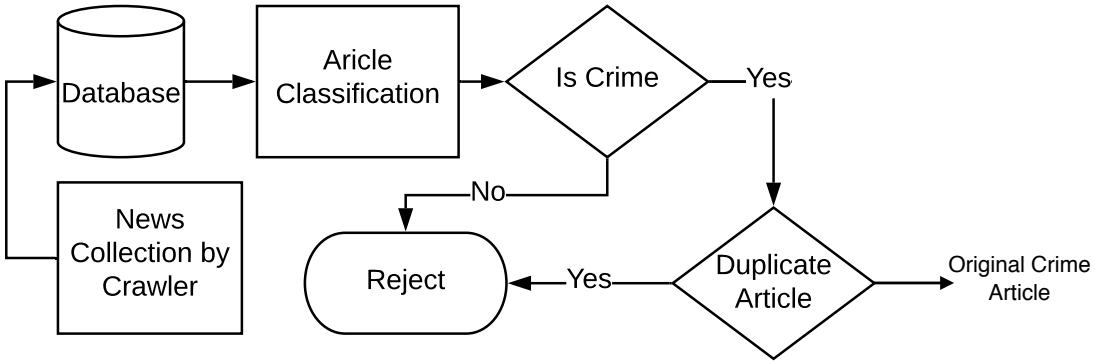

Figure 2.1: In the above shown pre-processing step, we perform taks to segregate out original crime articles

score is the sum of severity score of all the words present in the article. Based on a threshold, we classify the article as crime or non-crime.

For an experimentally selected optimal threshold, we run this algorithm on our labelled dataset and results are shown in Table 2.1. The results show that the naive algorithm has a very high recall, but accuracy and precision are low. Analysis of the results showed that low accuracy was due to the ambiguity of words since one word can be used for several meanings. Further, we observed that treating the title as a separate entity could enhance accuracy. Most related works suffer from this limitation Jayaweera et al. [2015].

2.1.2 Improved Method: Ambiguity score based classification

The previous methods either train a standard classifier, or rely on crime word list on the whole text. To improve the baseline methods, we make two important changes. First, we consider the title and body as two distinct entities of a news article. Second, we introduce ambiguity score to mitigate the effect contextual ambiguity. Title of a new is an essential part of the news and is a reflection of the complete article. If there is a crime word present in the title we classify it as crime article. Results (Table 2.1 show that if we use the only title for classification, accuracy is good, but the recall is not effective. Similarly, if we use only the body of an article as a source and classify it as crime article if atleast one

Algorithm 1: Baseline Crime Classification

Result: Crime or Non-Crime lables
Input: articleText, crimeList
Input: List Of Major Crime Words
Initialization;
$crimeList \leftarrow listOfMajorCrimeWords$;
$synonymList \leftarrow synonyms\ of\ crimeList$;
$SeverityScore(synonymWord) \leftarrow SeverityScore(mainWord)$ * $SimilarityScore$;
$crimeWordList \leftarrow crimeList + synonymList$;
$articleScore \leftarrow 0$;
for $word \in preprocess(article)$ **do**
    if $word \in crimeWordList$ **then**
        $articleScore+ = SeverityScore(word)$;
    end
end
if $articleScore > threshold$ **then**
    Crime Article;
else
    Non-Crime Article;
end

Methods	Naive	Title Only	Body Only	Improved
Accuracy	0.749	0.810	0.669	0.845
Precision	0.652	0.822	0.584	0.794
Recall	0.982	0.752	0.994	0.942

Table 2.1: Results of Crime Classification Models

crime word is present, then recall is 99.6% but precision and accuracy are poor. Therefore, we propose to treat title and body as separate features, and overall article score is determined as a combination of both.

In the naive approach, each crime word was weighted by severity score and weights were based on subjectivity. To remove subjectivity, we propose to give equal weight to all the crime words. However, if a crime word can be used in different contexts, then its weight is calculated as the number of crime context possible divided by the total number of contexts for that word. We call this as

ambiguity score for that word. For example, "hit" can be used in two contexts, "Hit the road", or "Hit a man". Thus, the ambiguity score for "hit" is 1/2. Based on the ambiguity score, we calculate the overall score of the article.

In our improved approach, we first classify based on title and body of the article. If both titles as well as body give the same classification result, the resulting classification is given as output. However, if there is a contradiction in classification, then we used ambiguity score of crime words and treat both body and title as a single entity and then classify the news article. The results (Table 2.1) show significant improvement in accuracy and precision with a slight degradation in recall in comparison to all other approaches.

2.2 Duplicate Article Detection

Duplicate article detection is an important part of the overall algorithm. It is general that a single incident will be covered by more than one news website, also with time new articles are added to update more information about an incident. Such cases may inflate the crime score for a location to an inappropriate proportion. To avoid that, we need to detect and discard, all the duplicate articles. First, we used tf-idf Juan et al. [2003] to find the similarity of the articles. If the most important words in two article are same, we classify them as the same article. However this is not sufficient as this method decides the importance of words based on how frequently those words are occurring in the article, whereas for our case importance of words may or may not depend on frequency of words. Basically if we talk about two duplicate crime articles, both should have same crime type, same persons(victim, accused or investigators) and same crime location. So to justify this we used crime type and entities(PERSON and LOCATION/GPE), which includes location, person details involved in the crime, to determine the similarity of two news articles. So in final near duplicate detection method, we calculated two scores, S1, which is decided based on the crime type and common entities involved in both articles using simhash technique as mentioned in Jayaweera et al. [2015], and S2, which is generated from td-idf method. Using these two scores the final result is calculated. We used the tf-idf factor in the final method to achieve good results in case of one large and one small duplicate

document comparison. It may be possible that a small document does not have sufficient entity information in that case it may be possible that these two documents may be classified as non-duplicate. In such cases, tf-idf will help to identify the similarity between two articles, also as tf-idf uses normalization which avoids biasing towards short or long documents. This is an important case as the framework will face this case most of the time because we first get basic news info(short document) and after in near future we get updated news with more details(long document) on the website. We tested this algorithm by creating a new dataset. We enumerated two sets of news articles A and B. Both set contains 20 news articles. All articles of set A is based on same story whereas, all articles of set B are all from different stories. Finally, combining all 40 articles we randomly generated 770 pairs, which are used to produce results of this algorithm as shown in Table [2.2] and Table [2.3].

	Predicted	
Actual	Duplicate	Not Duplicate
Duplicate	188	15
Not duplicate	30	537

Table 2.2: Duplicate Detection algorithm results

Metrics	**Values**
Accuracy	94.15%
Precision	86.23%
Recall	92.61%
F1-Score	89.31%

Table 2.3: Duplicate Detection algorithm results

For our framework to detect whether an incoming news article is duplicate or not, we need to compare this news article with all other news articles present inside our database. It is obvious to observe that initially this cost of comparison will be very low due to less number of articles in database, however as the database increases the number of comparison will also shoot up. Hence this will

slow down our framework. In order to solve this problem we article information to prune out some unnecessary comparisons. First, Fixing the time span for comparison i.e. article published today will only be compared to articles published in last N months. This value of N is flexible and can be changed as per requirement. Second, there is no need to compare articles from two different locations because generally crime happened at a location L will be published only under that location section of news website. Using this way the overall complexity of duplicate detection will be reduced to number of articles published in the given location for past N months. Table [2.4] shows duplicate detection result for 10 articles. The result is calculated for four different time spans. Table [2.5] show time taken by system to execute duplicate detection algorithm for 50 articles. First column shows the time required for comparing articles with all articles published in past X days from current article's date. Column 2 shows the time required for comparing articles with only those articles which are published in same location as current article in past X days. It can be observed that there is a high difference between the execution time of both scenario. We have taken 30 days as our fix time span for the framework as it balances both the execution time and false negatives.

15 Days		**30 Days**		**60 Days**		**90 Days**	
ID	**Dup ID**	**ID**	**Dup ID**	**ID**	**Dup ID**	**ID**	**Dup ID**
1001	None	1001	28402	1001	28402	1001	28402
1002	26961	1002	26961	1002	26961	1002	26961
1013	12948	1013	12948	1013	12948	1013	12948
1021	6710	1021	6710	1021	6710	1021	6710
1031	6663	1031	6663	1031	6663	1031	6663
1035	2327	1035	2327	1035	2327	1035	2327
1050	9503	1050	9503	1050	9503	1050	9503
1062	None	1062	None	1062	5698	1062	5698
1078	None	1078	None	1078	8586	1078	8586
1088	None	1088	7852	1088	7852	1088	7852

Table 2.4: Results of duplicate detection by fixing the time span for comparison as X days, where X is 15, 30, 60 and 90 days repectively. ID refers to Article ID and Dup ID refers to respective Duplicate Article ID.

Days	Without Location(mins)	With Location(mins)
15	67.11	21.74
30	104.99	28.69
60	146.57	37.18
90	171.20	44.87

Table 2.5: Time taken by the system to run duplicate detection algorithm over 50 articles. With Location means comparing only those articles which has same crime location. Days indicates that current article will be compared to articles which are published within X days before current article.

Chapter 3

Building Crime Database

The overview of the process of building a crime database is given in Fig. 3.1. Location, crime type, date and named entities of each article are extracted and stored in the database.

3.1 Crime Location Extraction

Extracting location information from an article is very tricky because in many cases, location name matches precisely with the person or organization's name. To address this challenge, we have divided this task into three parts. First to extract all entities from the article. Second, we extract potential location entities. And finally, we classify these potential locations as crime or non-crime locations. The details of these three steps are given next.

3.2 Entity Extraction

To resolve ambiguity between name, locations and organizations, we first extracted all named entities in the article with tags ORGANIZATION, LOCATION/GPE or PERSON. We used NLTK NE chunker Johnson [2016] and Stanford NER tagger Manning et al. [2006] to tag these entities. NLTK NE chunker is a supervised Maximum Entropy classifier trained on Automatic Content Extraction (ACE) data. Stanford NER tagger is computationally expensive than

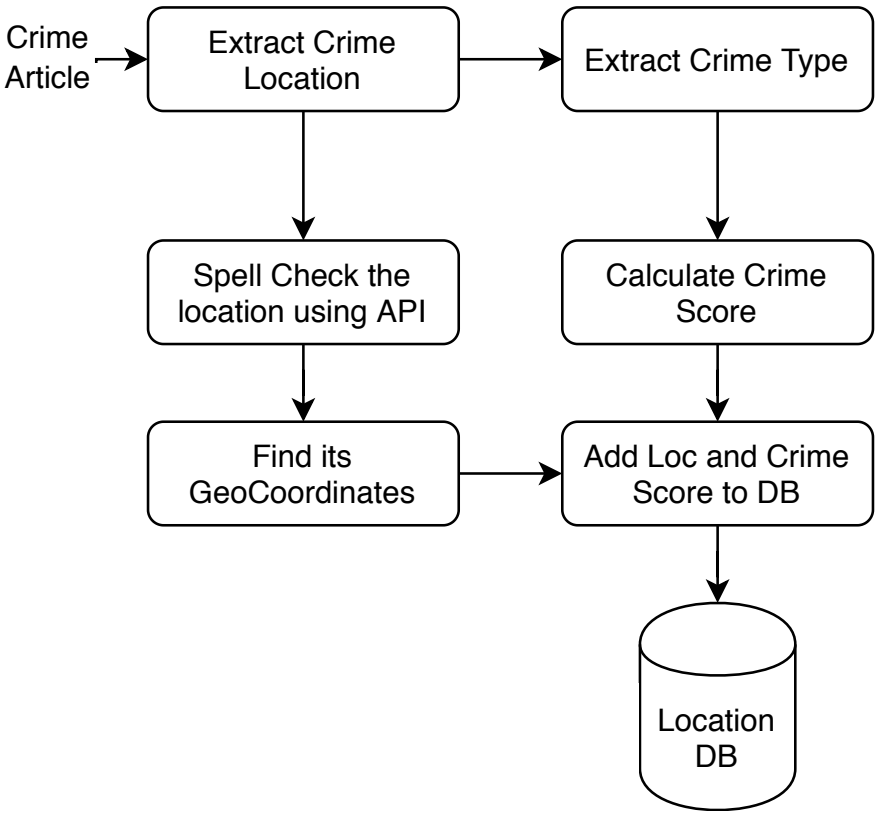

Figure 3.1: Important steps to build a crime database from the original crime articles.

NLTK NE chunker as it uses the advanced statistical algorithm. It is also a machine learning-based trained Conditional Random Field (CRF) model. We observe that, in both these taggers, the name was classified as locations and vice-versa. In the next subsection, we propose an algorithm to find locations from the named entities.

3.3 Classification of entities as location or non-location

As we are focusing on extracting location information from news articles, we have considered the following tags from the tagger's predefined list - GPE/LOCATION, ORGANIZATION, and PERSON. The major challenge that we faced in this task is how to differentiate between location name and person's name. Here are some examples where both NLTK and Stanford Tagger goes wrong, "A man found murdered in Anand Vihar, Delhi.". For this sentence, NLTK predicts Anand Vihar as GPE(Location), whereas it predicts Delhi as a PERSON entity. Talking about Stanford Tagger, it predicted Anand Vihar as a PERSON entity and Delhi as a LOCATION entity. So to achieve this classification and remove such ambiguities to some extent, we have added some extra checks. If you go through the location names in India, you can observe some commonly used words in location names like Vihar, Nagar, Pradesh, Chowk, etc. Using this knowledge, we have created a list of commonly used words in Indian location names as $Common\_Used\_Words$ and a check is performed that ensures no PERSON entity contains any tags from this list because it is very infrequent that a person's name contains such tags. We ran this algorithm over tagged set of news articles for both NLTK and Stanford Taggers. There were 577 location tagged news articles. When we added this check along with NLTK tagger for separating location entities from all tagged entities we observed an increase of 11.06% in the accuracy of NLTK tagger and 3.81% increase in the accuracy of Stanford Tagger as shown in table [3.1].This check is basically a comparison from list of common used words in location names of any country which can be added with any tagger and can be used for identifying the location names of that country from any given

text.

Method	Without Check	With Check
NLTK	52.15%	63.21%
Stanford Tagger	78.96%	82.77%

Table 3.1: Accuarcy improvement results for Location Separation from all entities by performing the check, presence of $Common\_Used\_Words$ in entities

Algorithm 2 shows the steps involved in the classification of extracted entities as a location or non-location. From $articleText$, entities are extracted with their respective tags using NLTK or Stanford taggers. Entities containing any word from the $Common\_used\_word$ list are classified as locations and finally, the algorithm returns a list of all classified location entities.

Algorithm 2: Method to classify entities as location

```
Result: Locations
Input: articleText
initialization;
Sentences ← sent_tokenize(articelText);
Entities ← EmptyList;
Locations ← EmptyList;
for sentence ∈ Sentences do
    entities ← EntityTagger(sentence);
    AppendToEntities(entities)
end
for entity ∈ Entities do
    if any(common_used_word) is substr of entity then
        AppendToLocations(entity)
    end
    else if any(preposition_list) before/after entity then
        AppendToLocations(entity)
    end
end
```

3.4 Classification of extracted locations as crime or non-crime location.

Next task is to classify extracted locations as crime or non-crime location. We considered two main factors by observing the writing of reporters to classify locations as crime locations. First, distance of location from the beginning of article. It is observed that in many cases, the writer mentions the name of crime location at the start of the article. For example, *"Sadar Raikot police on Wednesday booked two persons for the murder of an electrician after a fight over the phone in village Brahampura late on Tuesday evening."* and *"A 36-year old man was found murdered in Anand Vihar, Delhi."* are the first sentences of news articles. Second, how close the location is mentioned from a crime word, as by observation crime locations are mentioned closer to crime words. For example, in *"Ramesh, a resident of Sultanpura, was arrested in the case of murder in Anand Vihar, Delhi."*, it can be observed that the location *"Anand Vihar"*, where the victim was murdered, is closer to crime word *"murder"* compared to the location *"Sultanpura"*, which is the place of residence of accused.

Algorithm 3 shows the step wise procedure of crime location classification. It takes set of locations, returned from Algorithm 2, and article text as input and returns a list of crime locations. First, a $SentDistScore$ is assigned to each sentence based on the distance from the beginning of article. Next, each location entity gets a $CrimeWordDist$ score, sum of distance of all crime words from location entity in a sentence. Here only those sentences are considered which contains that location entity. Finally, a total score is assigned to each location which is the sum of reciprocal of SentDistScore and reciprocal of CrimeWordDist. finally we have decided a threshold empirically for separating out crime locations from non-crime locations. More on experiments is discussed in evaluation section.

Algorithm 3: Proposed method for Location Classification as Crime or Non-crime Location

Result: $CrimeLocations$
Input: $Locations, articleText$
initialization;
$Sentences \leftarrow sent\_tokenize(articelText)$;
$CrimeLocations \leftarrow EmptyList$;
$numOfSents \leftarrow length(Sentences)$;
$LocScore \leftarrow EmptyDict$
for $sent \in enumerate(Sentences)$ **do**
  $SentDistScore[sent] \leftarrow 1.0/(distanceFromBeginning)$;
end
for $loc \in Locations$ **do**
  for $sent \in Sentences$ **do**
    if loc is substr of sent **then**
      $finalcrimeLocScore[loc] \leftarrow$
       $SentDistScore[sent] + CrimeWordDist[loc]$;
    end
  end
end
for $loc \in Locations$ **do**
  if $LocScore[loc] < threshold$ **then**
    $AppendToCrimeLocations(loc)$
  end
end

Chapter 4

Crime Score Calculation

This information of all the articles is used to calculate overall crime score for a particular location.

We considered ten distinct crime classes. These classes are shown in X-axis of Fig 4.1. Each crime class may consist of different sub-crimes. For example, 'robbery' includes 'theft', 'snatching', 'stealing' etc. One word can be associated with more crime classes. For example, we associate Homicide with Murder, and Death with Murder, Suicide, Accident and Terrorism. From the crime article, we extract all the crime words. These crime words votes for their respective classes. Class with maximum votes is chosen for that article. This model has **76.12%** accuracy in 10 classes.

Next we assign severity to each crime class. For example, murder is more severe than robbery, so its severity score should be high. However, the judgement of severity score is subjective. It varies from one person to other and also depends on the context. Therefore, we did a survey, asking respondents how unsafe would they feel on the scale of 1 to 10 if a particular crime has occurred at a place they are planning to visit. The response of the survey was post processed and is normalised to a scale of 5. The survey was responded by 186 people and after post processing of the data, severity score of each crime class is shown in Fig. 4.1. We may observe that suicide got minimum severity score and terrorism has maximum score.

As discussed before, crime score of a location is function of crime type, their frequency and when they occurred. For each crime news article, we calculate

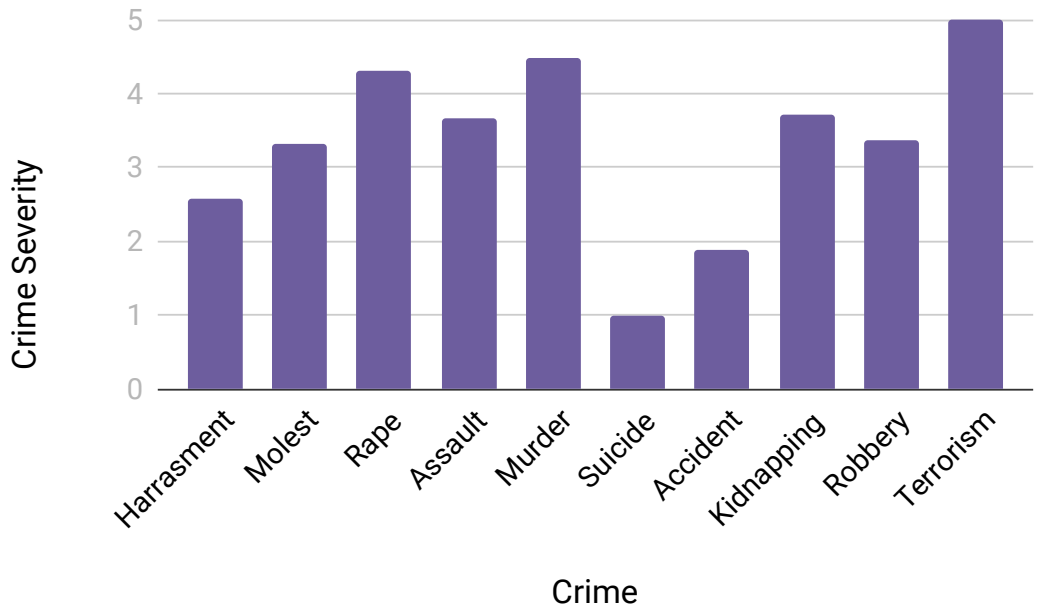

Figure 4.1: Crime Severity Score

crime score, α, based on crime type and its severity. Since effect of a crime diminishes exponentially with time, therefore, we use half-life concept to model current crime score of that article. Current crime score for a past article is given by β as defined by following equation:

$$\beta = \alpha * e^{-(\lambda * t)} \tag{4.1}$$

$$\lambda = \frac{ln2}{T}, \tag{4.2}$$

where, T is half life constant, and t is the age of the article in number of days. Thus, if T is 180 days, crime score of the article will be half after 180 days.

Now that we have the crime score, we need to store them into the database, but before we need to process the extracted location. First, we use the spellcheck API on the extracted crime location to check if the spelling is correct. Then we use another API that returns the geocoordinates given the location. Finally, we

store the location with its geocoordinates and crime score into the database. If the location is already present in the database, then we update it by adding the calculated crime score. These crime scores in the database are not normalized, so we are also storing a max crime score in a singleton table and keep updating it whenever a new location is added or updated in the database. When a query comes for a crime score of any location, we return by normalizing it using the max score. Therefore, the returned crime score always lies between 0-1 so that any user can understand the severity of it. The crime score stored in the database is also decayed periodically (say after one month), and for that, we also store the date when the crime score was last updated.

The whole point of this framework is to get the crime score of any location. But in our database, we won't have all the locations in the world. So when we get a query of a new location which is not present in the database, we use the crime score of its surrounding crime locations to predict its crime score. We take the location lies inside the 3σ radius where σ is the variance and use their crime score say γ, take their Gaussian factor and sum them up as shown in equation 4.3. It will return the crime score that will be returned to the user and added to the database with its geocoordinates.

$$crimeScore = \sum_{i=location}^{inRange} e^{\frac{-x^2}{2\sigma^2}} * \gamma_i \tag{4.3}$$

Method	**Potential Locations**	**Crime Locations**
NLTK	63.21%	60.08%
Stanford Tagger	82.77%	79.24%

Table 4.1: Accuarcy results for Location Extraction

Chapter 5

Evaluation

We have tested the above framework on our dataset crawled from various news web sites, mentioned in the table [1.3]. As already specified in section 1.2, we have divided the dataset into two parts. We used the first part(tagged manually) to verify various methods of the framework.

We tried multiple methods for crime classification, but in our final method, we achieved 85.4% Accuracy with 79.4% Precision and 94.2% Recall. We were more focused to achieve high Recall because for our implementation it wouldn't be good if we are missing some crime articles. Various experiments were carried out in Crime location extraction using both taggers. Using the NLTK tagger alone, we have achieved an accuracy of 75.92% in extracting all potential locations and 69.88% in separating out all possible crime locations. For Stanford Tagger, we obtained an accuracy of 81.02% and 78.24%, respectively. Finally, we have executed our overall framework on the remaining portion of the dataset to build a database that contains updated crime scores and other details of possible crime locations. The results thus obtained were represented on an interactive map of India. For reference, we show here the heat map of Delhi in Fig. 5.2. This is a heat map to blue circles represents the colder regions, i.e. low crime occurring areas and red circles represents the hotter regions, i.e. high crime occurring areas. The radius of the circle represents the coverage area of crime region. To prove that our results are adequate, we have cross-verified our crime score results with crime reports available on trusted websites. We have compared our results with `https://www.numbeo.com/crime/rankings.jsp`, an online re-

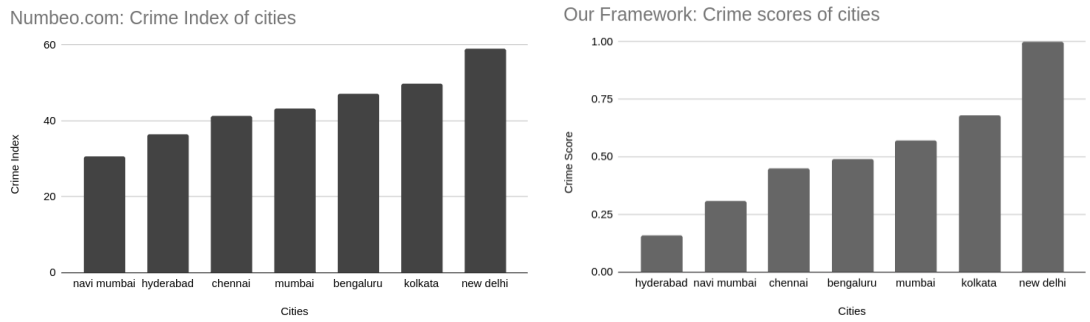

Figure 5.1: Crime index comparison

source that maintains crime index periodically every year. We have collected the crime index of cities for the year 2018 and 2019 from `Numbeo.com` and taken their average as the final crime index. Figure 5.1 shows the city-wise bar graph of crime score/index. left part of Fig. 5.1 shows the web results, and right part of Fig. 5.1 shows our framework results. One can observe that the order of crime score/index of cities are similar and follows almost similar pattern. There crime index is out of 100 whereas our crime score of cities lies between 0 and 1. Further, we have also compared state wise results of our framework with online resources. Due to the unavailability of accurate crime reports of states on the internet, we compared results with some previous results. We have made a comparison with these two websites: https://gigadom.in/2015/01/16/a-crime-map-of-india-in-r-crime-against-women/ and https://www.moneycontrol.com/news/india/10-indian-states-that-have-the-highest-number-of-crimes-against-women-4572681-8.html. The results were almost alike.

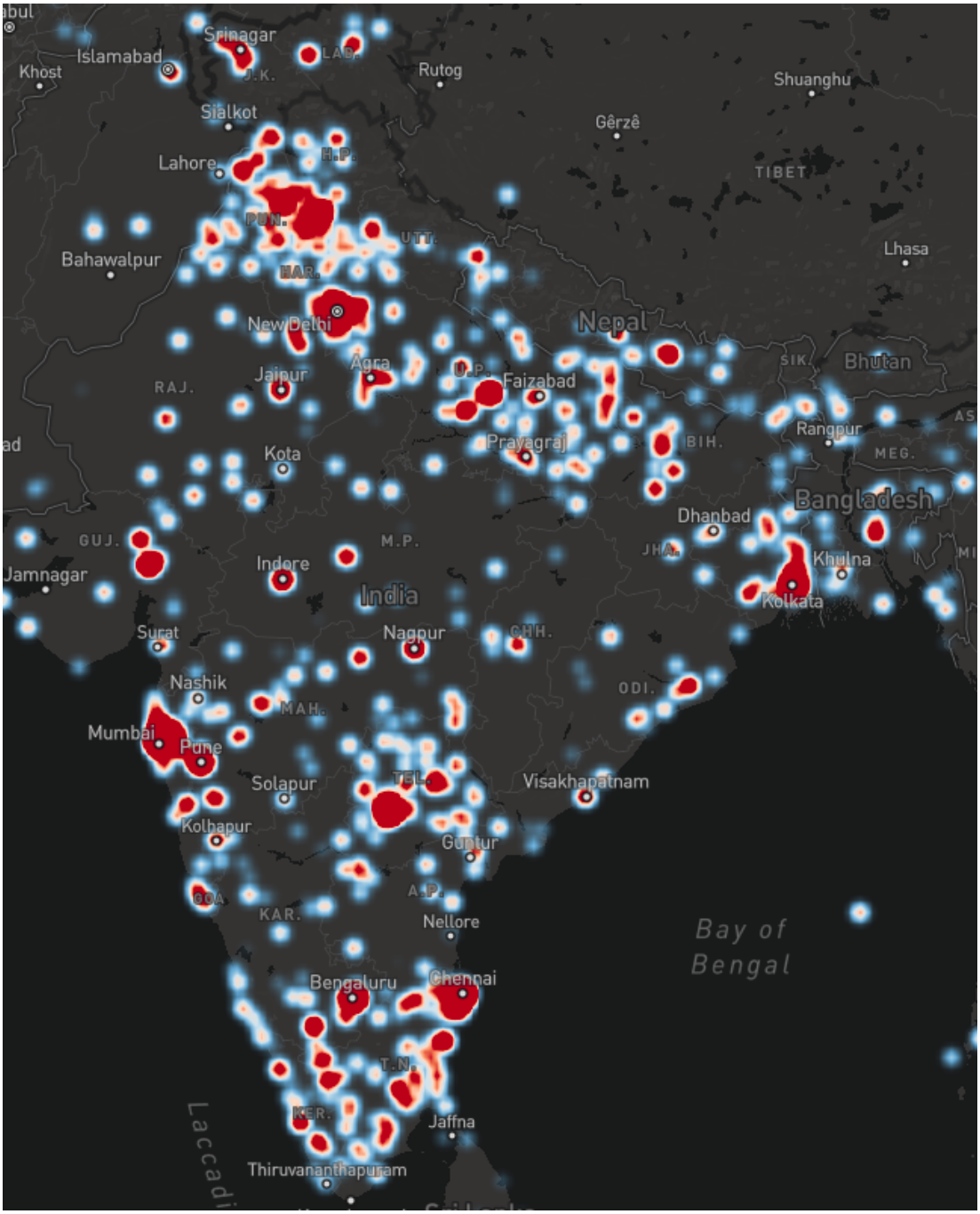

Figure 5.2: Crime Heat Map of India

Chapter 6

Future Work

Having information about crime of locations is very helpful as it can help us to make various decisions like Whether to visit a place for holidays or not, decide a good place to buy house, finding the safest routes, etc. This information can also help companies to decide whether to put their new office/factory.

This framework can provide sufficient information to Police forces which can help them to dynamically organise police force in more crime-prone areas in order to reduce crime rates at those locations. Also if we have sufficient information about crime at different locations we can use other advance methods to predict future crimes at different locations.

For now our database is only storing the final crime score of a location and it gets updated when a new crime article comes but if the crime score of the locations are frozen with time stamp and stored in a different database, then we can generate a graph which can show the history of a crime at any location over time.

The framework that we developed is generalised and flexible, hence it can be integrated with many other applications. For example, the framework can be integrated with Google Maps to display safest path between two places along with fastest, shortest paths.

Chapter 7

Conclusion

Crime information of a location is very useful from safety perspective. It is difficult for a normal person to get this information from any source, so this paper proposes an automated framework that provides this information. In this paper we have build a automated framework, that collects news from online websites, classify news articles as crime or non-crime, finds the duplicate articles, identify crime locations and build a model for dynamic crime score for a location or city. Subset of collected data was tagged with ground truth and all the proposed algorithms were tested on this data. Experiments show that our crime classification accuracy is 84.5%, crime location extraction accuracy is 79.24% and 76% accuracy for identifying correct crime class. We ran the framework for the whole data-set(stats about data-set is mentioned in table [7.1]) and got the results which we verified with online available crime rates and finally we presented the result as a heat map layered over the map of India. The framework is generalised and flexible, it can be integrated with many other application, for example safest route finding application.

Total Articles	345870
Non-crime Articles	266624
Crime Articles	79246
Duplicate Crime Articles	12096
Crime Locations	3311

Table 7.1: Dataset Stats

Appendix A

A.1 Data Collection

The main focus of this project was to obtain a crime score of different locations using online resources. There are a lot of resources available online from where we can get the crime information about any place. However, we aimed to use News Articles as they are trust-able(which is very important when one aims to provide such reliable platform), contains an adequate amount of information about crime incidents, and also widely available on different News websites. Now to collect a good amount of News Articles from News websites, we wrote a script that to crawl News Articles from 6 six News websites, as mentioned in table 1.3. Using Crontab we scheduled to run the script daily at 2 AM IST, which forks six threads each handling crawling of News Articles from one website. We collected News Articles from these websites for around two years.

A.2 Data Tagging

To verify the correctness of our algorithms for crime classification and crime location extraction, we required some set of tagged data. To achieve this requirement, we created an interface for tagging data, as shown in figure A.1. The interface asks to classify the article as Crime or Non-Crime, further it asks the user to provide more information about the article if it is a Crime article.

Figure A.1: Data Tagging Interface

A.3 Crime Severity Survey

The crime rating of a location depends on the number of crime incidents occurring at that location and their severity. For example, no one would like to visit or reside at a place that has high murder/rape incidents compared to a location having high accident/theft cases. This framework provides us the number and type of crime that happened at a location, but we required a method to decide a final crime score that should be assigned to that location given the stats. To fulfill this requirement we did a small survey, asking each user to rate the severity of different types of crimes. Figure A.2 shows a part of the surveyed form and figure 4.1 shows the graphical report of the survey.

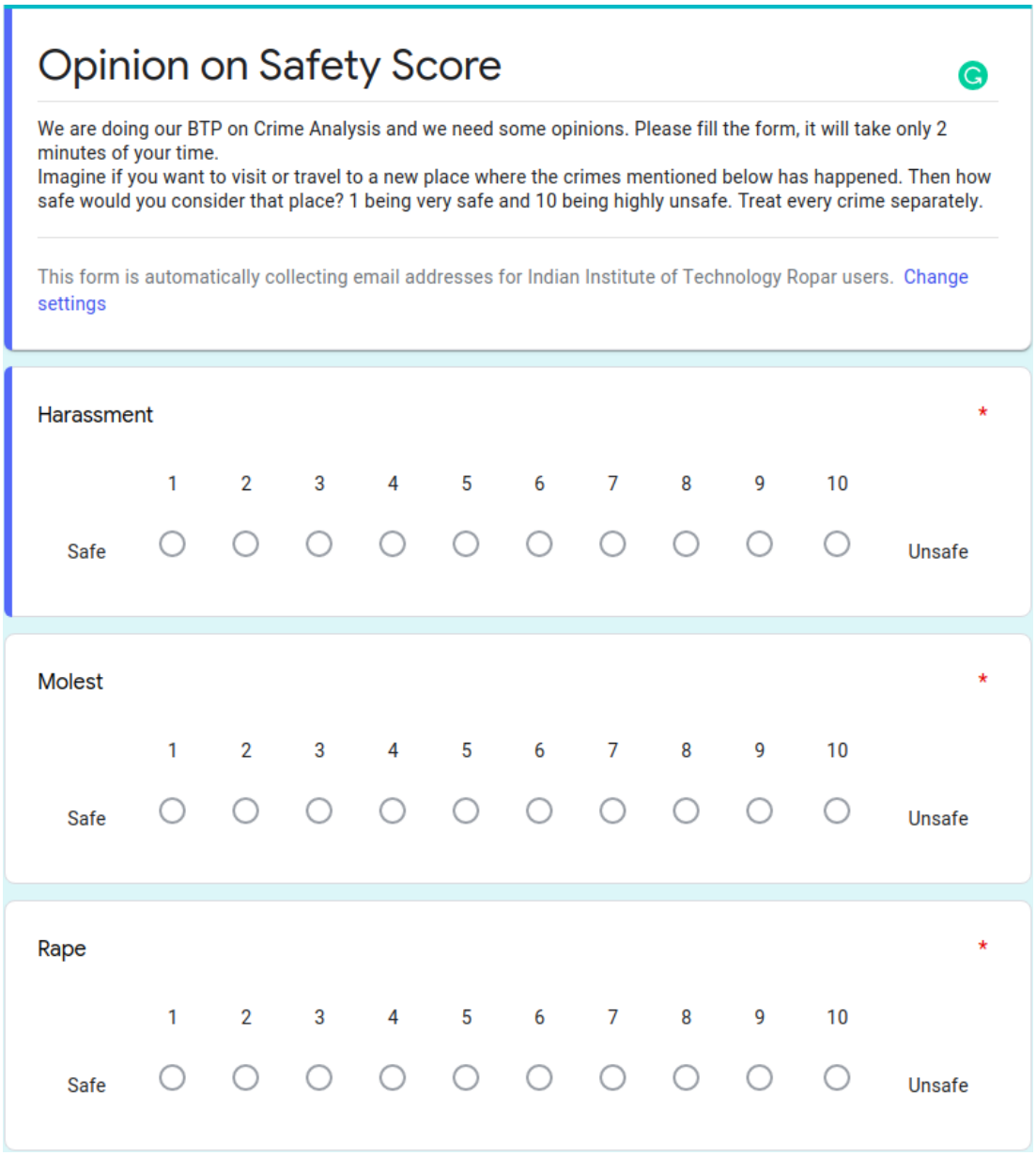

Figure A.2: Part of a survey form to find the severity of the crime

Appendix B

B.1 Database Information

A database is required to store the News Articles and Location information. Two tables were created as shown in figure B.1, to store the respective information.

B.1.1 News Article Information Table

In this table, we stored information about all the crawled News Articles for analysis and future use. This table contains two types of information. First, the crawled from the website i.e. Article's Title, Text, Date Published, URL, and location section under which the News was published. Second, the information which is retrieved via News Article tagging i.e. News Type(Crime or Non-Crime) and entities present in the article. Further, the table also had a column for storing the duplicate News Article ID.

B.1.2 Location Information Table

This table was to store locations and their crime score. It also included other information like Geo-Coordinates of each location, which is used in calculating the crime score of unknown location. We used LocationIQ API, mentioned in C.1.1, to get Geo-Coordinate information.

B.

```
Database changed
MariaDB [CRIME_ANALYSIS]> desc NewsArticles;
+----------------------+--------------+------+-----+---------+----------------+
| Field                | Type         | Null | Key | Default | Extra          |
+----------------------+--------------+------+-----+---------+----------------+
| NewsArticleID        | int(11)      | NO   | PRI | NULL    | auto_increment |
| NewsArticleTitle     | varchar(200) | YES  |     | NULL    |                |
| NewsArticleText      | text         | YES  |     | NULL    |                |
| NewsArticleDate      | datetime     | YES  |     | NULL    |                |
| NewsArticleUrl       | varchar(767) | NO   | UNI | NULL    |                |
| Location             | varchar(100) | YES  |     | NULL    |                |
| AnalyzedBy           | varchar(20)  | YES  |     | NULL    |                |
| NewsType             | varchar(10)  | YES  |     | NULL    |                |
| TimeStamp            | datetime     | YES  |     | NULL    |                |
| DuplicateReferenceId | int(11)      | YES  |     | NULL    |                |
| ExtractedLocation    | varchar(256) | YES  |     | NULL    |                |
| ExtractedName        | varchar(256) | YES  |     | NULL    |                |
| entities_dict        | text         | YES  |     | NULL    |                |
+----------------------+--------------+------+-----+---------+----------------+
13 rows in set (0.00 sec)

MariaDB [CRIME_ANALYSIS]> desc LocationInfo;
+------------------+-------------+------+-----+---------+----------------+
| Field            | Type        | Null | Key | Default | Extra          |
+------------------+-------------+------+-----+---------+----------------+
| location_id      | int(11)     | NO   | PRI | NULL    | auto_increment |
| queried_name     | varchar(50) | YES  |     | NULL    |                |
| display_name     | varchar(50) | YES  |     | NULL    |                |
| city             | varchar(50) | YES  |     | NULL    |                |
| state            | varchar(50) | YES  |     | NULL    |                |
| postcode         | varchar(6)  | YES  |     | NULL    |                |
| class            | varchar(20) | YES  |     | NULL    |                |
| type             | varchar(20) | YES  |     | NULL    |                |
| osm_type         | varchar(20) | YES  |     | NULL    |                |
| lat              | double      | YES  |     | NULL    |                |
| lon              | double      | YES  |     | NULL    |                |
| min_lat          | double      | YES  |     | NULL    |                |
| max_lat          | double      | YES  |     | NULL    |                |
| min_lon          | double      | YES  |     | NULL    |                |
| max_lon          | double      | YES  |     | NULL    |                |
| crime_score      | double      | YES  |     | NULL    |                |
| LastModifiedDate | datetime    | YES  |     | NULL    |                |
+------------------+-------------+------+-----+---------+----------------+
17 rows in set (0.00 sec)
```

Figure B.1: Description of important Tables in Database

Appendix C

C.1 API Details

This appendix talks about APIs used in this project. We used two APIs, one to get the location details and the second one to correct spellings.

C.1.1 LocationIQ

This API is used to extract the location's information like its Geo-Coordinates, full address, type of location, and bounding box. This extracted information is stored in the Location Information table of DB. The python code of this API is shown in figure C.1.

C.1.2 Bing Spell Check

In some News Articles, we observed that the spelling of a location is different from others. This can either be due to mis-spelling of location in the article or maybe because the native people of that place write it in that way only. However, the LocationIQ API that we used, returns NULL value for locations that are misspelled. So this mis-spelling issue is a hurdle in using LocationIQ API. To avoid this hurdle we need to correct those spellings(i.e. to find the most widely used spelling of that word). To perform this task we used Bing Spell Check API. Figure C.2 shows the python code of API.

```python
class LocationIQ(object):

    api             = 'api_key'
    url             = "https://api.locationiq.com/v1/autocomplete.php"
    count           = 0   # for maintaining request count

    def return_location_details(self, loc, knownRequest=True):
        '''
            Function to request API for location information
            loc          : queried location
            knownRequest : to check if this location is already requested earlier or not.
        '''
        data = {
            'key' : self.api,
            'q'   : loc
        }

        try:
            if self.count <= 1000:
                response = requests.get(self.url, params = data)
                self.count += 1
            else:
                ## sleep for X seconds until api refreshes
                current = datetime.now()
                next_day = datetime.now() + timedelta(days=1)
                next_day.replace(hour=1, minute=0, second=0)
                sleep_time = abs((next_day - current).total_seconds())
                time.sleep(sleep_time)
                response = requests.get(self.url, params = data)
                self.count = 1
        except Exception as e:

            f = open(api_logfile, 'a')
            f.write("======== Log written on " + str(datetime.now())+ "========\n")
            f.write("Error while LocationIQ request/date calculation DB table: word\n")
            f.write(str(e))
            f.write("======================== End of error ===================\n\n" )
            f.close()

            return {}

        return response
```

Figure C.1: Python code for LocationIQ API

```python
class SpellCheck(object):

    api             = "api_key"
    endpoint        = "https://api.cognitive.microsoft.com/bing/v5.0/spellcheck"  ## endpoint to make request to
    data            = {}
    ## other required params
    params          = {
        "mkt"  : "en-us",
        "mode" : "proof"
    }

    headers         = {
        "Content-Type" : "application/x-www-form-urlencoded",
        "Ocp-Apim-Subscription-Key" : api
    }
    already_present_in_DB = []

    def set_data(self, text):
        '''
            Function to set data for spell checking
        '''
        self.data['text'] = text.lower()

    def spell_check(self):
        '''
            Function to request api and get response
        '''
        try:
            self.data['text'] = self.filter_new_words()
            if self.data['text'] == '' or not self.data['text']:
                    return None
            response = requests.post(self.endpoint, headers=self.headers, params=self.params, data=self.data)
            return response.json()

        except Exception as e:
            f = open(api_logfile, 'a')
            f.write("======== Log written on " + str(datetime.now())+ "========\n")
            f.write("Error while filtering list or requesting BING SPELL Check API\n")
            f.write(str(e))
            f.write("======================== End of error ====================\n\n" )
            f.close()
            return None
```

Figure C.2: Python Code for Bing Spell Check API

Appendix D

D.1 Interface

To make this framework for the end-user, we required an easy and user-friendly interface that can provide the user with the crime score of queried location. To achieve this, we developed a web interface, as shown in figure D.1, where the user can input the location, and the respective crime score will be shown to the user.

D.

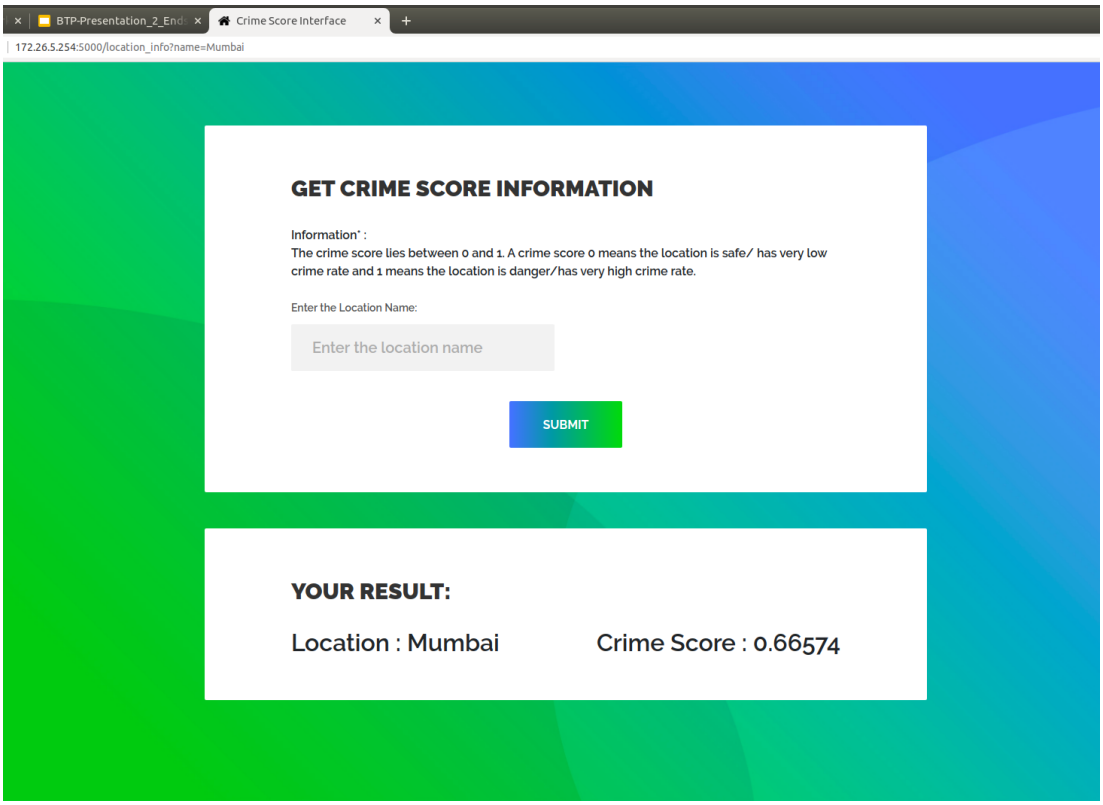

Figure D.1: User-friendly Interface to query location's crime score

Appendix E

E.1 Heatmap Tools

Visual representation is the best way to analyse the data and since we have the data from all over the country we were able to make the heat map. We had all the geocoordinates of the locations along with their crime score. We stored these data in the **GeoJson** format file. We used the tool **Mapbox** and GeoJson file as input data to generate the heat map. When zoomed out, you can see a continuous map in figure 5.2 and if you zoom in, individual crime locations can be seen in figure E.1.

E.

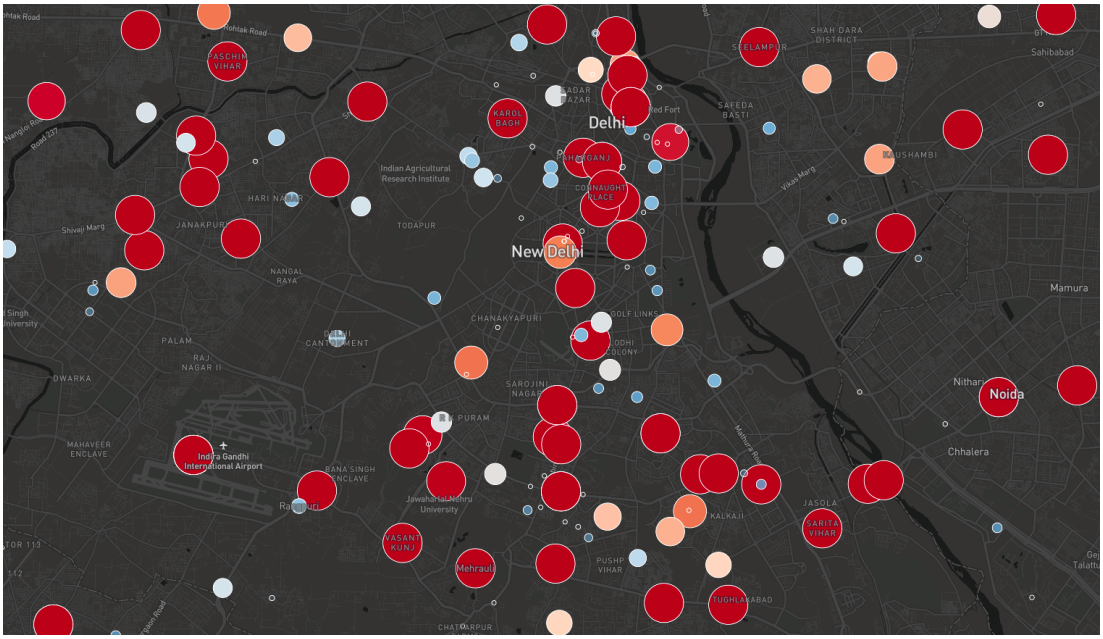

Figure E.1: Zoomed In Crime Heat Map of Delhi

References

Rexy Arulanandam, Bastin Tony Roy Savarimuthu, and Maryam A Purvis. Extracting crime information from online newspaper articles. In *Proceedings of the second australasian web conference-volume 155*, pages 31–38, 2014. 3, 4, 5

Neeraj Goel, Rajat Sharma, N Nikhil, SD Mahanoor, and Mukesh Saini. A crowdsourced adaptive safe navigation for smart cities. In *2017 IEEE International Symposium on Multimedia (ISM)*, pages 382–387. IEEE, 2017. 4

Mehedee Hassan and Mohammad Zahidur Rahman. Crime news analysis: Location and story detection. In *2017 20th International Conference of Computer and Information Technology (ICCIT)*, pages 1–6. IEEE, 2017. 4, 5

Isuru Jayaweera, Chamath Sajeewa, Sampath Liyanage, Tharindu Wijewardane, Indika Perera, and Adeesha Wijayasiri. Crime analytics: Analysis of crimes through newspaper articles. In *2015 Moratuwa Engineering Research Conference (MERCon)*, pages 277–282. IEEE, 2015. 3, 4, 5, 9, 11

Robert M. Johnson. Lifting the hood on nltk's ne chunker, 2016. URL `https://mattshomepage.com/articles/2016/May/23/nltk_nec/`. Last Accessed 2019. 15

Anant Joshi, A Sai Sabitha, and Tanupriya Choudhury. Crime analysis using k-means clustering. In *2017 3rd International Conference on Computational Intelligence and Networks (CINE)*, pages 33–39. IEEE, 2017. 3, 4, 5

Juan et al. Using tf-idf to determine word relevance in document queries. In

Proceedings of the first instructional conference on machine learning, volume 242, pages 133–142. Piscataway, NJ, 2003. 11

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. Stanford named entity recognizer (ner), 2006. URL `https://nlp.stanford.edu/software/CRF-NER.html`. Last accessed 2019. 15

Vasu Sharma, Rajat Kulshreshtha, Puneet Singh, Nishant Agrawal, and Akshay Kumar. Analyzing newspaper crime reports for identification of safe transit paths. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 17–24, 2015. 3, 4, 5

Devendra Kumar Tayal, Arti Jain, Surbhi Arora, Surbhi Agarwal, Tushar Gupta, and Nikhil Tyagi. Crime detection and criminal identification in india using data mining techniques. *AI & society*, 30(1):117–127, 2015. 3, 4, 5

Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994. 8

Sunil Yadav, Meet Timbadia, Ajit Yadav, Rohit Vishwakarma, and Nikhilesh Yadav. Crime pattern detection, analysis & prediction. In *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*, volume 1, pages 225–230. IEEE, 2017. 3