

INDIAN INSTITUTE OF TECHNOLOGY DELHI


DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE PAGE

REPORT OF THE PROJECT SHORTLISTED UNDER SURA - 2018

PROJECT TITLE : EVENT MAP GENERATOR

Submitted by: Samarth Aggarwal

Name of Student	Deptt. /Centre	Entry No.	Contact No. & email address (Mobile no.)	Signature of the Students
1. Samarth Aggarwal	CSE	2016CS10395	9810201131 (cs1160395@iit.ac.in)	
2. Chinmay Rai	CSE	2016CS50615	8826064810 (cs5160615@iitd.ac.in)	

Name and signatures of the Facilitator: Prof. Mausam

His Department : Computer Science and Engineering

Mobile No. of the Facilitator : +91-9871253384

Date of submission of report in IRD : 22nd October, 2018



Summer Undergraduate Research Award (SURA)
Report
Automatically Curated Accident Maps

Chinmay Rai
Computer Science & Engineering
(2016CS50615)
Contact : 826064810
cs5160615@iitd.ac.in

Samarth Aggarwal
Computer Science & Engineering
(2016CS10395)
Contact : 9810201131
cs1160395@iitd.ac.in

Prof.Mausam
Facilitator
Department of CSE
mausam@cse.iitd.ernet.in

Prof. Rijurekha Sen
Advisor
Department of CSE
riju@cse.iitd.ernet.in

Acknowledgements

We would like to thank Prof. Rijurekha Sen for her constant guidance and support throughout the course of project. We are also grateful to Prof. Mausam for his invaluable inputs and discussions in terms of the natural language processing domain. Additionally, we would like to express our gratitude to Prof. Srikanta Bedathur, Swarnadeep Saha, Aman Madaan, all of whom were extremely supportive and responsive to our queries.

Introduction and Motivation

The project work aims at development of an end to end engine to extract data from internet, process it and present it a graphic format for analysis. The target application of this project is the road accidents that occur over the nation in a given time frame. The current mode of analysis of these events has very coarse granularity in the sense that the analysis is done on a collective basis and stats are reported as a whole for the dataset. This type of analysis is easily available and is less useful as it cannot be used to corner things down to the cause of accident and solve it from the root . We want to change this and focus on event wise information collection where each event is treated as separate datapoint. This has many advantages particularly in the for the organization working for curbing road accidents.

When writing the report for a project work it is very essential to specify the expertise level of the pupils working on the project. The expertise of the reader might vary severely from the us, hence it is important that we make our position clear in order to be judged in fair manner. This work was a combined venture of two Computer Science sophomores, who have a very basic exposure to machine learning (fundamentals, linear models, gradient descent etc.). The project work pertained mostly with Natural Language Processing and Machine Learning. It is very important to specify that the people working on this project have no exposure of NLP whatsoever and have been introduced to the basics of the subject during the course of the project. This is also evident from the fact that we have used rule based NLP for solving most of the problems that we encountered, instead of using Models that learn as they work (which is more State-of-the-art and De-facto). The project also involved the use of many tools and libraries which have built by reputed research groups and it is clearly stated that we do not have complete understanding of the functionality of those tools . We have taken help of many research papers throughout the course of work. Inferences drawn from many of those papers have been used in various parts of the project. We have referenced the list of papers at the end of report.

Data Collection

The most easily available and open source data is the News which is also available on the web in form of archives of certain media houses. In our case we chose the Times of India Archive to begin with primarily because of two reasons :

- a) There is no other Archive of such a scale and coverage as compared to TOI
- b) The Archive was properly organised in a hierarchical manner making it easy to crawl using a web spider.

We went through the HTML structure of the source code of Article's webpage to identify the "div tags", which contained the headline and text of article and then fabricated scripts to download the list of news article indexed to each date on the archive using PHP scripts. PHP being an old language was very slow in crawling the data. After some more research we came up with a crawler library based on python called "Scrapy". Scrapy was about 10 times faster than PHP scripts hence we decided to use Scrapy to download as a JSON list of (title,url) of article. We did not download the body because most of these articles were not going to be relevant to us. Hence we decided that we will get the text of article only after we have identified which articles are relevant to us. After having crawled loads of data (which were essentially news article headlines), we need to segregate those which report about road accidents.

Data Classification

We planned to train a classifier model to segregate the text on the basis of the headlines. This Motivation to go with "Headlines" as the the input for the system was primarily because:

- a) Using the text of articles will not be computationally efficient because average length of an article is 100 words which will in turn lead to a lot of features for training.
- b) More often than not, the text of the article contains a lot of noisy data which may lead to misclassification of data.
- c) It is practically possible for human to read the headline and make the classification without giving it much thought. Since we are simply training the system to mimic this process, it seemed natural to go with headlines as the input for the system.

Machine learning uses mathematical models for prediction. We had to convert the headlines into mathematical constructs so that they can be fed into the model. One of the very trivial algorithms that anyone might suggest for text classification is the Keyword Detection, i.e. inputs are classified on the basis of presence/absence of a certain set of keywords. We decided to go through the dataset to judge whether we could use keyword detection. We realised that we could not corner things down to a particular set of words which were “always” being used for headlines of accident reporting articles. On the contrary we observed that the feature that qualifies any example as a positive one is more related to a set of words being used together rather than a word being used in the headline. To capture this trait we needed to use a model that captures relation between the words used in the headline.

We decided to use the Bag of Words Model with a slight modification. The classical bag of words model records the presence/absence of each word in an input as a N-dimensional vector where N is the number of words in the corpus. We decided to enrich this bag of words with all the bigrams and trigrams which can be formed using the words of the corpus. The corpus is defined as the collection of all the words across all headlines in the training (and test) examples.

We also made use of some very standard heuristics to support our bag of words model. To begin with, we tried using the Term Frequency - Inverse Document frequency numerical method for counting the frequency of words in the descriptor (Count Vector) of headline. It did not prove to be useful. We also curated a set of stop words, (which are very commonly used words) and removed them from being accounted for while counting, i.e. they will not be visible to the counter. We also experimented with a combination of Unigram, Bigram and Trigram to corner down to the combination which yielded the best results.

One consideration that arises here is that, in order to make this model work the corpus has to be sufficiently large, i.e. in objective terms it has to be large enough to capture all the words which may be used to define the event in question. Speaking of accidents, we statistically (by manually looking at data) decided that we should take as many examples as it requires to capture the ecosystem of words

being used in the articles related to accidents. We tend to focus only on the positive examples because our classification problem is a biased in nature, i.e. The number of article which report accident is only 1-2% of all the article in a certain timeframe. Since in the bag of words model the onus is on the presence of words we decided to go with the above suggested line of thought.

After having obtained the vector describing each headline we then sought to find the model to be used for training. One of the computationally light algorithms that is commonly used for text classification is Support Vector Machines. SVM tries to maximize the margin of the nearest datapoint from the separating hyperplane. Non-linear SVM makes use of kernels which are nothing but a algebraic transformation of the existing set of features, to obtain some new features, which could then be used for classification. This algorithm is particularly easier to train because it only considers the Support vectors (Data Points which lie within a boundary of proposed margin) in the optimization process. SVM being quick to train also helps us to tune the hyperparameter tuning by doing a grid search over the hyperparameter space to find the optimum parameters .

The Library which was used for all the above described process is the SCIKIT-LEARN library. It provides host of features ranging from vectorizing the input, variety of models, fine-tuning the model and evaluating the performance of the model. It is a python based library with easy to use and understand constructs.

For the purpose of training the system we manually curated a Training set of 2000 examples and a test set of 200 examples. After training the model we evaluated the accuracy and found it to be close to 89%. We then looked at the examples from the test set which were misclassified and added these to training examples to train a new model and evaluated it against a new Test set. After repeating the above process we obtained an accuracy of 95% and decided that we will be using this model for classification.

To use and reuse the model we had to save the model in .sav format,also known as “Pickling of Model”. This model can then we invoked multiple times from a separate file to perform classification task.

Data Processing

The most challenging and time consuming part of the whole project was extraction of specific information from the article text which was desired by us for representation on a map. We identified that some very primary attribute related to any accidents could be the location, date, time, no. of people injured and killed and the cause of accidents. We decided that we are going to try extracting these fields for each of the article. We decided that we wanted to start with extracting location of accidents.

Failed Attempt of Extracting Location

When we started to research about the extraction of location related information on internet, we learned that there were two very popular techniques of doing so, first is the use of a Named Entity Recognition (extraction of Proper Nouns), second was the use of some tailor made tools for extracting locational information from text.

Named Entity Recognition is a very Standard Analysis Method which is used for a host of NLP tasks. A lot of good research has already been done in this field and most of the established NLP research group like Stanford NLP, Uwash NLP, have developed state-of-the-art NER tools. These tools can also classify the named entity as name of person, organization and location. These are Machine learning models trained over humongous amount of data over a large period of time and were optimized to perform quick and accurate recognition. We sought to utilize this feature to our case and extract location.

When we started to experiment with the available NER tools, we found out that they were not able to identify all the location tokens in the text. When we tried finding the solution to this problem we learned that most of the NERs that we were trying belonged to research groups of universities from the

USA and Europe. They were trained using corpuses that were curated from the local sources like print media and literature. As expected, there was an inherent bias in those trained models with respect to the grammatical structures used and names used in western world. For examples if we list out the names of places, people and organizations from India we would find that their features like word Shape (one of the feature used in NER) will be very different from that observed for names in the western world. Stanford NLP even provided the facility of retraining its model with custom dataset. We thought of procuring a dataset curated in India which could be put to use. We searched for such a dataset on the websites of many research groups in the country. In most of the cases, the labs were either not willing to share their dataset or working on NER in Indic languages, so we could not obtain a dataset.

We then turned to the tailor made tools like LNEEx, GeoParsePy, Geolocator, GeograPy, to see if they could be of any use to us. All this tools are open source and are available on Github. While trying to run these tools we realised that some of them were quite huge in size (in order of 10's of GB). On probing we found out that most of them used an index of location created from OpenStreetMap. The problem with OSM is that it's database in India is very sparse and not as good as Google Maps. Eventually we found that these tools performed even worse than NER's.

We turned to the NER again and tried finding a better one which could work with the text data that we had. We then tried the NER on Google Cloud NLP. It's performance was better than any of then tool so far and it was able to identify the words which were instances of location. But this tool brought us another realization that more often than not, there are more than one instances of locations in an article (they may be residence of victim, the source/ destination of his journey etc.); hence if we simply filter out the location instances in the article then we would be left with a list of location token with no clue of their context.

At this point of time we decided of re-evaluate the approach we were adopting, because nothing seemed to be working for us. After doing some more research and looking more carefully at the the structure of articles we observed that we need to clear the clutter from the text of the article before

jumping to extraction. We sought to filter out only those sentences which had the information which was relevant to us. A traditional approach would have been to use a text summarizer tool. After trying some of them we realised that each of those tools had different interpretation of “summary”. Unfortunately none of them summarized the information we needed. We decided that we are going to make a model for classification of relevant sentences.

Filtering relevant sentences

To classify the sentences of an article, we first had to tokenize the text into sentences. Though it may seem a trivial task using the NLTK library, but since we were working with a large corpus, there would always be few examples that would cause failure. The “SentTokenizer” function in the NLTK library tokenized the text where a string was terminating with a dot and the next string starts with after a space character. It is customary in English to start the new sentence after giving a space after a full stop. But our corpus was written by humans hence was contained instances where this was not followed. We had to preprocess the text before feeding into the tokenizer. After dealing with this issue we then tried training a SVM classifier similar to that used for headlines. We curated a dataset of relevant sentences and another of irrelevant sentences and trained the model. The results were not very promising because :

- a) The model had no constraint to the number of relevant sentences per article. It saw each sentence as a separate entity. This meant there were instances where no sentences were selected for a particular article.
- b) The SVM was successful in headline classification because the headlines of the accidents contained a very similar kind of words arranged in the similar feature to report minimal information. While in this case then sentences we were dealing with were full length sentences which had lots of information ranging from location, time, date and casualty. This information being larger was also written using different grammar and sentence structure in different articles. This made it hard for the model to capture it.
- c) The amount of noise in the text was tremendous, and might have caused misclassification in many cases.

- d) The kind of sentences we were trying to filter did not even pertain to a single class. Some of them described time/date , some described location, while others described the number of casualty. Sometimes sentences also contained combination of these.

We again went to analyse the data to get some insights from it. We then observed that all the sentences which deemed to be relevant to us, were characterized by one common feature. All these sentences used verbs which belong to particular set of verbs. We thought that we might be able to capture these sentences if we come up with heuristic to adjudge if a sentence contains the right “Trigger” verbs.

We decided that first we will concentrate on time and location and tried to curate a set of “Trigger” verbs whose presence could help our case. The “Importance” of all the “Trigger” verbs in the set was also not equal. Higher the “Importance” of verb in sentence, higher are the chances that the sentences will contain some relevant information. Hence we split up the set into three degrees according to the “Importance” of verbs. We assigned weights to each of these sets. The code takes all the verbs in the sentences and assigns a score to the sentence. The three top scoring sentences were selected from each article. There were some additional heuristics used for example a sentence was immediately counted in without scoring if it had some verb/phrase of great “Importance”. Some examples that we were very frequently characteristic of sentences that were relevant to us are “took place” and “met with an accident”. After implementing the sentence selector for extracting locations, we realised that the same set of trigger verbs do not indicate the presence of time clause as well. Hence, we curated different sets of trigger verbs for location, time and casualty detection. Configuring and testing the model multiple times led us to a model which seemed to be decent to work with.

We are fully aware of the fact that this model will have a lot of bias with respect to this dataset, because we used rule-based modelling at the very fundamental level. But we also want to point to the fact that this dataset is large enough for us to achieve what we sought to achieve from it and our purpose is not to make a general purpose sentence classifier.

One of the very important tool that we were using at this point of time was OpenIE 5.0. It is an relation extraction tool which is made by joint venture of DAIR, IITD and Uwash NLP group. Briefly speaking the tool uses a trained model to split up sentences into (Subject, Verb, object clauses) tuples. It also has ability to recognize multiple subjects and could categorize information in the object part into spatial and temporal clauses.

We realised that we could obtain the location and time information clause by processing the “object” part of sentences which contain the trigger verbs. The set of Trigger verbs that we are using focuses on events like death, killing, crashing, hitting, running over etc. So the “object” part of the selected sentences provide us the information about those events, which is precisely what was desired by us. Here, it is imperative to state that we are preserving the context thereby correcting on one of the crucial mistake we committed earlier.

After classifying and filtering out the relevant sentences we looked to the “object” part of those sentences. Following is a list of such phrases:

1. “in crashed into the barricades along BRTS (Bus Rapid Transit System) route in Akurdi”
2. “for dinner to a restaurant near Bhakti-Shakti Chowk, On Wednesday night”
3. “near Chena creek bridge along Ghodbunder Road at 5pm”
4. “in separate accidents in the city on Thursday night”
5. “from the Nirankari Colony in north - west Delhi 's Mukherjee Nagar”
6. “by a heavy vehicle near the Mangolpuri market In another incident”
7. “12 people and eight parked vehicles before hitting a pole in west Delhi's Tilak Nagar on Thursday evening”
8. “after an unidentified vehicle hit him near Harsaru village on Pataudi road on Wednesday night”
9. “about a dozen passengers near Sursangda village of Kalavad taluka of Jamnagar, on Saturday morning”

It can be easily observed that much of the locational information is sandwiched between prepositions. Hence we thought that we will break these parts and compare the obtained list of phrases with a POS-tagged index of the original sentence. If the phrase contains a proper noun, then it is a location with a very high probability. *It is important to note that Part Of Speech (POS) index should be created using original sentence, because POS tagging is context based.*

You can also see that many of the tokens sandwiched between the prepositions are time tokens. These time tokens can be easily extracted using a library like SUTime Tagger, but the problem would be that the library only highlights time tokens, but does not organize them systematically into a data structure. We decided to write a Regular expression matcher code, which will output proper organized temporal information. Below is the definition of the data Structure used to describe the time and date of accident.

```
*****
enum Day {unspecified, monday, tuesday, wednesday, thursday, friday, saturday, sunday;}
enum TimePeriod { none, early_morning, late_night, noon, evening; }
class date_time
{
    public int hrs,min;          public boolean am;          public TimePeriod period;
    public int date;            public int month;          public Day day;
    public String date_of_happening="";          public date_time(){}
}
*****
```

While mentioning the date of accident it is essential to highlight the fact that the date of reporting is naturally, not the same as the date of accident. Accidents often reported in terms of the day of the week. For example “A 30-year-old pedestrian was killed after an unidentified vehicle hit him near Harsaru village on Pataudi road on Wednesday night”. This line reports “wednesday” as the day of

accident. To compute the date of accident we had to find the day of week when it was reported and then reference the day of accident mentioned to find out the actual date of the accident.

Casualty Detection

For the purpose of finding the casualty of accident we are using the subject of the sentences which are classified as relevant. The subject clauses were obtained from the first element of the relational tuple in the output of OpenIE. Some examples of the subject clauses are :

he, she, two others, two persons, the driver, Nitin, her mother, a man, the victim, four of his colleagues, a 25-year-old man, at least 16 amarnath pilgrims, ten people, including two children, four members of a family and their driver, etc.

Since we are mainly interested in the number of casualties rather than who those casualties actually were, so the task at hand is to output a single number given a subject clause. At first we thought that we should simply aim at the digits occurring in the subject clause. However, when we analysed the frequency distribution of different subject clauses, we realised that the number of casualties are reported as a number in digits in a very small chunk of reports. For the vast majority, this number is either written in words or has to be inferred from the context. Even when the subject clause contains a digit, it straight away cannot be taken to be the number of casualties as more often than not, digits are used to denote other informations such as age of the casualty (eg. a 25-year-old man).

Keeping the above problems in mind, we developed a regex based extractor to identify numbers from the subject clause. We also built a hash-table of different words and the number that they map to. This included numbers written in words (eg. five, seven), special words denoting a certain group (eg. duo, couple, trio, both), singular pronouns (eg. he, she), etc. Along with the number, we also took note of its word index. This was used to evaluate the context in which the number was used to decide whether or not the number actually denoted the count of casualties.

For eg. if the number is followed by 'year' then almost certainly, it is used to denote the age of the victim.

A major issue that popped up while attempting this approach was that some of the subject clauses enumerated inanimate objects as well. For eg. - A speeding car. Our system so far would even count this as a casualty which is incorrect. The solution to this problem was enabling our system to distinguish between animate and inanimate objects. For this, we used 'Wordnet', a lexical database developed by Princeton University.

If the subject clause contained the conjunction 'and', then we heuristically split that clause at the connector 'and', recursively found the number of casualties in each of the sub-clause and added the results to obtain the total number of casualties. Other such rules were employed to precisely determine the number of casualties.

The task was to come up with a single number for each article. But the system described above outputs a number for every subject clause. Since we select roughly 3 sentences from every article, hence we will have the same amount of choices of number of casualties. Heuristically, we decided to choose the maximum out of these values as the number of casualties in that article. The intuition behind this decision was that in most articles, one of the sentences given an account of the casualties in totality. The rest of the sentences simply shed more light on the different classes of victims.

For eg. 11 people were injured in a road accident. Four of them are critically injured and were immediately taken to the hospital.

Processing this sentence will yield 11 and 4 as outputs. It is evident from this example why the choice of max (11 in this case) is suitable. This marked the completion of the casualty detection system of our engine.

Cause Detection

The next step is to detect the cause of accident. By this stage, we had already seen a large number of news articles. We noticed that more or less all the accidents can be attributed to about 10-12 primary causes such as overspeeding, drink and drive, etc. The cause of the accident was extracted in a manner similar to location extraction. We first identified the relevant sentences using trigger verbs. We

narrowed down on 10 classes and collected about 30 training examples for each class. Finally, we trained a machine learning classifier on this dataset using logistic regression.

Testing and Evaluation

While building any system, it is imperative to have an evaluation metric in place that tells as the extent to which our system is able to achieve its objectives. In our case, we manually curated a dataset consisting of about 200 articles from July 2017 and enlisted their location, time, number of casualties and cause. Next, we gave those articles to our engine and compared the outputs obtained. The output comparison was done manually so that the evaluation could be precise. The following were the results:

Accuracy in location extraction = 77.84 %

Accuracy in time extraction = 63.29 %

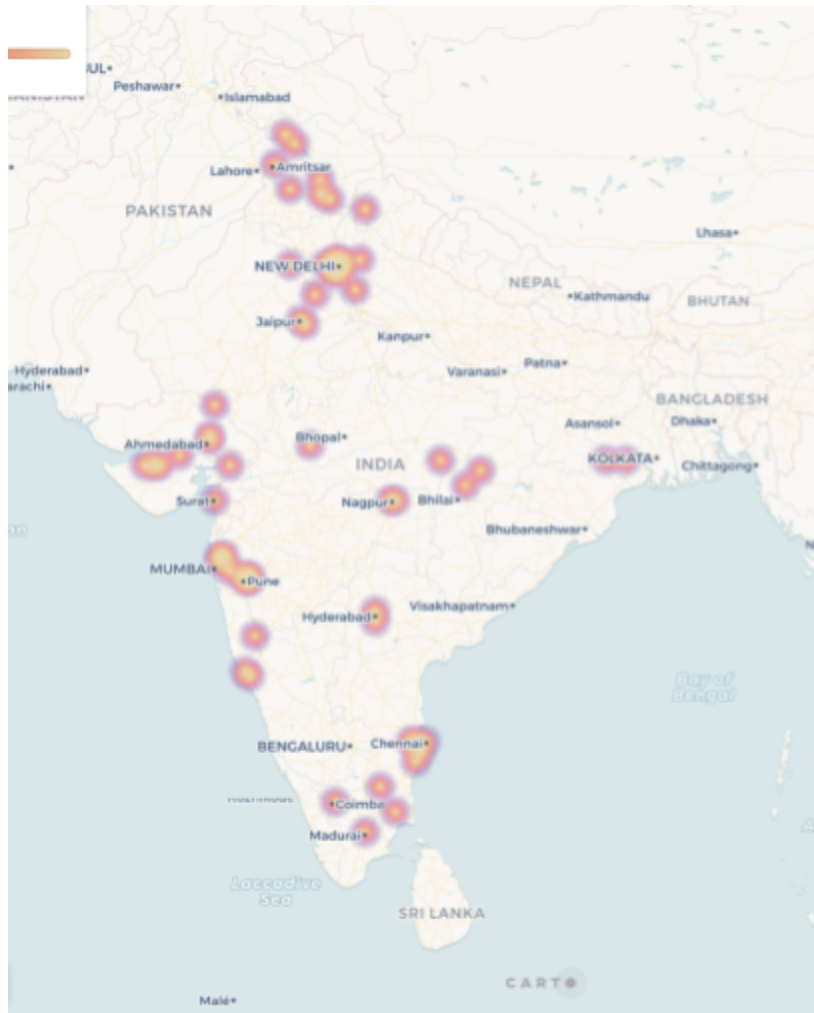
Upon analysis, we found that the reason behind low accuracy in time detection was that the method of selecting sentences based on appropriate trigger verbs is not very effective in extracting notions of time. Furthermore, this evaluation brought forward some more verbs that our system should have included in the first place. Steps to further improve the performance are still going on.

Plotting and Visualization

The last stage of this project is depicting the findings in a visual format. For the same, first we geotagged the locations output by the engine. We have used “Google’s Geo Tagging API” for the same. Next, the geographical coordinates obtained were given to Carto (a visualization tool). Alongside, other graphs plotted included plotting the number of casualties in accidents with their pinpointed location, plotting maps for accidents occurring in a specific time interval, etc.



Accident Map
for July 7,
2017 showing
locations only



Accident Map
(Heat Map) for
July 7, 2017
showing
accidents
weighted by
their number
of casualties

Summary

In our SURA project, we successfully created an end to end system that when given the url of an online newspaper archive (such as <https://timesofindia.indiatimes.com/2018/5/26/archivelist/year-2018,month-5,starttime-43246.cms>) will output an accident map depicting the location, time and number of casualties incurred in the accident. Other information such as the cause of accident can also be retrieved from our engine. We have employed web scrapers to scrape news articles from these newspaper archives. After this, different natural language processing techniques have been employed to extract the desired informations. Plotting and visualisation were achieved with the help geo tagging and Carto. We experimented with different NLP techniques to improve the performance at different stages. Further, unlike the state-of-the-art tools used in the west that are accustomed to understanding of western language and scenarios, this tool is developed specifically for the indian english and scenario. It will be interesting to see how this tool rolls out to curb accidents and provide key insights when employed on a large scale.

Our code base is available at <https://github.com/samarthaggarwal/SURA>

Uses and Applications

Following are some interesting applications of this tools:

1. Monitoring accidents across a landscape
2. Identification of accident prone location along with the major cause of accidents there thereby taking the most effective measure to curb them
3. Evolution of accident-prone zones by considering accident maps for different times
4. Heat maps depicting accidents weighted by the number of casualties caused
5. Once integrated with FIRs as an input mechanism, the system will become exhaustive and hence more reliable

References

1. A Deep Learning Approach for Detecting Traffic Accidents from Social Media Data, 2018 - by Zhenhua Zhang, Qing He, Jing Gao, Ming Ni.
2. <https://github.com/dair-iitd/OpenIE-standalone> - Open Information Extraction (Open IE) system from the University of Washington (UW) and Indian Institute of Technology, Delhi (IIT Delhi)
3. Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pp. 363-370. <http://nlp.stanford.edu/~manning/papers/gibbscrf3.pdf>
4. Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55-60.
5. Bird, Steven, Ewan Klein, and Edward Loper (2009), *Natural Language Processing with Python*, O'Reilly Media.
6. Sibun, Penelope & David S. Farrar, Content Characterization Using Word Shape Tokens, *Proceedings of the 15th International Conference on Computational Linguistics*, Kyoto, Japan, 1994, pp 686-690.
7. Al-Olimat, H.S., Thirunarayan, K., Shalin, V., Sheth, A.: Location name extraction from targeted text streams using Gazeer-based statistical language models, vol. 11, no. 17 (2017).
8. Aman Madaan, Ashish Mittal, Mausam, Ganesh Ramakrishnan, Sunita Sarawagi : Numerical Relation Extraction with Minimal Supervision (2016).
9. <https://github.com/NEO-IE/Numberule> : Number rule based approach for numerical relation extraction.
10. <https://wordnet.princeton.edu> : Wordnet resource by Princeton University