TL;DR

An event extraction and summarization system

Rajkushal Ananthkumar†

Department of CSE, SUNY Buffalo, Buffalo, NY 14214, USA, rajkusha@buffalo.edu

Akshaya Krishnan Pattammal

Department of CSE, SUNY Buffalo, Buffalo, NY 14214, USA, ak243@buffalo.edu

ABSTRACT

Automatic extraction of news events relating to violence against state elements or civilians is a valuable capability for decision support systems. Existing systems to perform such extraction involves human intervention to collect and process data into the desired format (ACLED, GDELT). The project entitles "TL;DR" describes a large-scale automated system for extracting violent incidents relating to protests/riots and violence against civilians. A comprehensive architecture is outlined that can identify, categorize, summarize and perform entity slot filling against the target event types. Furthermore, an attempt is made to relate the recorded events to politics and elections taking place in the countries India, Indonesia, and Thailand. This allows a better understanding of the driving factors for these events.

KEYWORDS

GDELT, ACLED, slot filling, automatic extraction

1 SOLUTION ARCHITECTURE

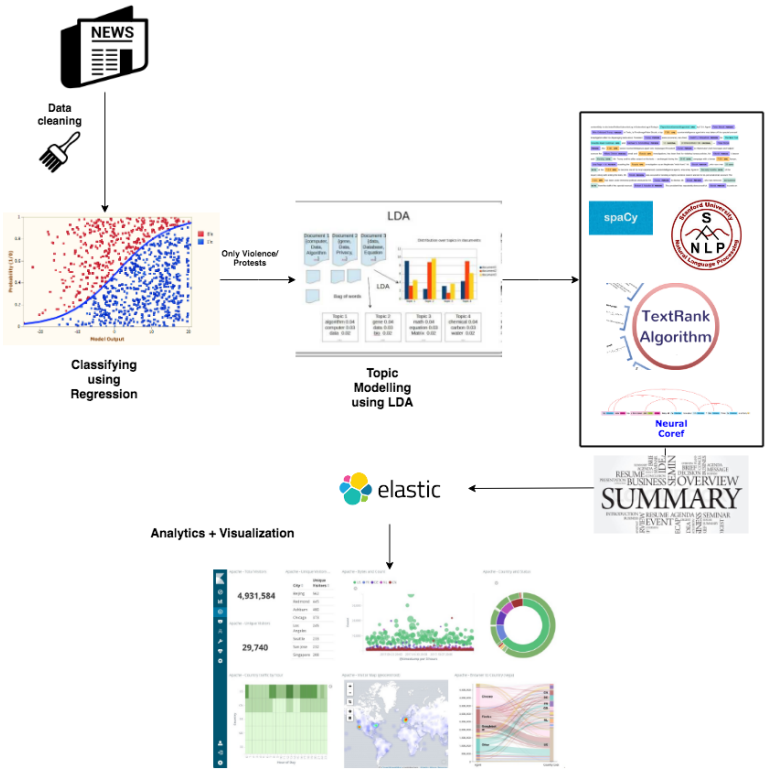

Figure 1: Solution Architecture

2 DATA SOURCES

Article extraction was accomplished using Recursive crawlers on the target news websites. A total of 35 news sources across various country-specific websites were used to crawled and each article was used as input to the news event extraction pipeline. The pipeline performs three key functions:

(1) Detecting if a news article report is a protest/riot-related event or, A violence against civilians' event.

(2) Determining if the news article report is pertaining to politics.

(3) Extracting the event entities such as location, date and Actors/organizations involved.

For Event Detection,

1. First, a pre-filtering step is to employ in which a list of keywords is used to exclude articles that do not contain event-related terms in the article text. The heuristic pertaining to how many event-related terms need to occur is determined separately for each event type and country. This allows the system to handle a large volume of news articles.
2. News articles are then analyzed by a supervised classifier that discards or assigns an event type to the accepted articles

3 DATA CLASSIFICATION

The project implemented an ML-based supervised classification scheme, where pre-classified data was used to train the model.

The label used for Multi-Class Classification was as per the scheme:

Non-relevant data : 0
Protest/Riot Data: 1
Violence against civilians Data: 2

The Data and labels were obtained from GDELT and ACLED.

Each new article is assumed to contain at most one event and framed the problem as a multiclass supervised learning task, where the objective is to train a classifier capable of accurately predicting the discrete class label {0, 1, 2} for a given news article.
Different classification approaches were compared for this task, namely Naive Bayes, Logistic regression, K Nearest Neighbors and Random Forest. It was found that initially, while the K Nearest Neighbors approach produced the best accuracy, a problem of overfitting the training data occurred. Using a Random Forest on a grid search parameters outperformed all other approaches.
Word embeddings were used as input features in all classifiers. In particular, a 200-dimensional Doc2Vec model was trained. Since these Doc2Vec vectors are able to capture event-specific syntactic and semantic information a better Vector representation for the classifier was achieved

Some other improvements that were made to assist feature extraction were :

- Tokenization.
- Stop word removal.
- N-Gram based tokens of size 2, to improve recognizing of terms eg “New York” as a single Term.
- L2 norm for vector regularization to avoiding overfitting.
- Set minimum number of documents a term is required to appear in (eg 5), in order to be considered for classification.

4 ENTITIES EXTRACTION

4.1 Topic Modeling

“*Topic Modeling is a technique to extract the hidden topics from large volumes of text.”[1]*

The articles returned by the classifier is of two types: Protests/ Riots and Violence against a group of civilians. To discover election-related insights, it was necessary to perform a semantic analysis of the documents. It would be a painstaking task to manually read hundreds of articles and attribute them to election violence. So, using the LDA approach for topic modeling it was possible to extract the articles and the keywords related to election violence.
The Gensim package provides with a function that could be used to train an LDA model. This requires our documents to be pre-processed (ie stopwords removal, lemmatized, etc). Additionally, one has to create a dictionary and a corpus with terms from the articles. These are fed as parameters to the function. This library could be used to obtain the topics and associated keywords. The number of topics is a hyperparameter and one has to modify as per needs. A coherence score is a measure of how well the model is performing.

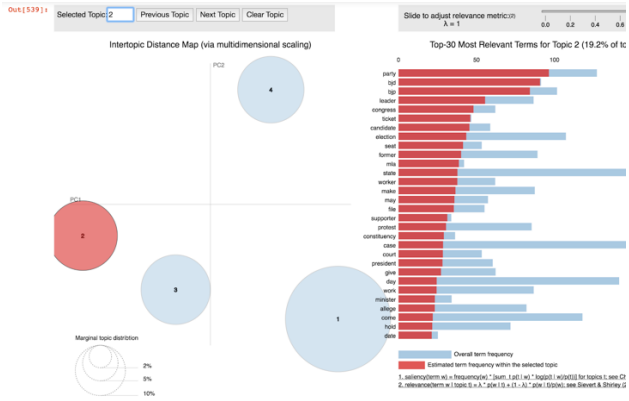

Figure 2: Topic Modeling

To further enhance the understanding, one could use the library pyLDAvis. Each bubble represents a topic. The bigger the bubble, the more prevalent the topic is among all the articles.

The Gensim library has another mode named mallet model. The mallet model provides a better quality of topics. Using, this model, a graph between the number of topics and the coherence score is generated to find the optimum value.

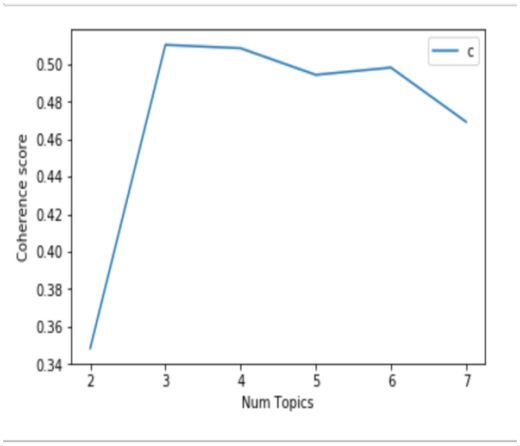

Figure 3: Graph between num topics and coherence score

4.2 Tagging Entities

To extract entities, two NER taggers are used. spaCY is an industry standard tagger. Though it is faster, it is not as accurate as StanfordNER tagger. For each entity, a list is created for each tagger. For example for the DATE field, two lists are created. spaCY_DATE denotes the date tagged by spaCY. Stanford_DATE denotes the DATE tagged by the Stanford tagger. Usage of two NER taggers for a different set of sentences was initially done as a forethought for slot filling. However, better approaches were developed for it later. Needless to say, the idea of using a hybrid approach did help with obtaining a different purview of entities tagged when the set of sentences varied.[4]

ALGORITHM 1: spACY tagger

1. Extract the first sentence mentioning the DATE.
2. Set window_size = 1
    a. Search for other entities in the sentence containing the DATE.
    b. Search for other entities in the sentence before and after the sentence containing the date (window_size = 1). [2]
    c. Add the tagged values to the spaCY lists.
3. Note the corresponding sentences for each entity.

ALGORITHM 2: Stanford tagger

1. Search for all entities in the entire text.
2. Add the tagged values to their corresponding Stanford lists.

Now, for each entity, there are two lists. But, the challenging task would be to narrow down to fewer values.

ALGORITHM 3: Slot filling

1. Run text rank algorithm for the entire article and extract the keywords.
2. For each entity
    a. list1 <= spacy_list
    b. list2 <= stanford_list
    c. combined_list <= list1 + list2
3. For each value in textrank[6]
    a) score_per_value <= levenshtein_distance(textrank_value, combined_list_value)
    b) Create a dictionary with key as the term and the value as the score obtained in the previous step.

Example:
Textrank_ORG = [Google] => keywords of the article

spacy_ORG/ stanford_ORG = [Google, Facebook, Microsoft, Amazon, Salesforce]

intersection_ORG = [{Google:1.00}, {Facebook:0.675}]

Thus, the most relevant terms along with their score for each entity are obtained.
The dateparser library was used to standardize the date value for each article. This library has a function that returns the value of the date when a specific date is given as the "relative point".
So, for example, if an article that was published on May 5th, 2019 has a keyword mentioning "yesterday", then this function would return "May 4th, 2019" as the date.

The latitude and longitude of each location were found using the geopy library in Python and Google's geocoding library.

4.3 Summarizer

For the summarization task, the Gensim's summarizer [5] with slight modifications worked really well. Though other options such as "summa" library were tried, the gensim's summarization performed better.
Since articles were of varying lengths, the ratio of the summary was modified depending upon the length of the article.

5 Evaluation

5.1 Dataset preparation

To construct the dataset we followed the following steps:

1. Download ACLED data using ACLED data export tool
2. Reverse Google search using "Source" and first 10 words of "Summary" from ACLED record to obtain URL against article.
3. Extract Article raw text by crawling the article URL.
4. For data extraction, we use News Please [9]. news-please is an open source news crawler that extracts structured information from news websites.News Please uses the following Python packages internally -
   a. Scrapy : Scrapy is an application framework for crawling web sites and extracting structured data
   b. Newspaper : Python library for extracting generating JSON representation of news articles.
   c. Readability : Python library for Data cleaning.
5. Regular expression matching was finally performed to remove Emojis and other Website specific special characters.

Total records collected was 1991 articles:

- Data from Protest/Riot related articles were 542 articles
- Data from violence against civilians related were 142 articles.
- Data from unrelated category (Neither Protest nor Violence) were 1307 articles.

Data records count was based on observed heuristics of frequency of articles in the major National News websites.
Based on this heuristic observation, we came up with a ratio of 6:2:1 to corresponding to Unrelated events, Protest/Riot and Violence against civilians.
The above-mentioned Data count more accurately describes the real world incident count, and hence was used to model the Classifier dataset.

5.2 Classification Accuracy

Word embeddings were used as input features in all classifiers. In particular, a 200-dimensional Doc2Vec[8] model was trained. Since these Doc2Vec vectors are able to capture event-specific syntactic and semantic information a better Vector representation for the classifier was achieved.

Logistic Regression	Multinomial Naive Bayes	KNeighborsClassifier (neighbors=5)	Random Forest
85.38%	82.73%	87.55%	92.5%

Table 1: Classification Accuracy

The metric used to perform evaluation was Accuracy, ie (Number of Correct Classifications / Total number of articles).

5.3 NER Tagging using ACLED as the ground truth

The dataset is updated with the ACLED's fields. The ACLED's fields were very abstractive in nature. It was indeed a tough task to match the fields. The aim of the evaluation is to assess the performance of TL;DR and make necessary modifications.
For the same reason, different approaches were considered for evaluating the fields. Firstly, manual evaluation of the fields was compared and reported. This was done so that one gets to manually comprehend the tagged entities along with the summary. Human comprehensibility would provide better confidence as the evaluation is ideally to check the information with an already existing gold truth. The system performed quite decently.

Method (a): Manual Approach

For manual evaluation we compared the ACLED True class with the predicted values of our system.

For ACLED we considered the below column values -

1) assoc_actor_1
2) actor2
3) assoc_actor_2
4) location
5) Event_date

From the Predicted Values we consider the following values -

1. Standardized_dates: Represents the date output predicted from our system
2. Intersect-person: Represented the Ranked Ordered list of the top 4 Person Entities predicted from our system
3. intersect-ORG: Represents the Ranked Ordered list of the top 4 Organization Entities predicted from our system
4. Intersect-location: Represents the Ranked Ordered list of the top 4 Location Entities predicted from our system

assoc_actor1	actor2	assoc_actor2	event_date	location
53.66 %	48%	63.63%	60.78%	70.58%

Table 4: Manual evaluation results

To check if we have made a correct prediction we compare the relevant ACLED columns with the relevant Predicted Values, and mark them as a match if we get a human agreeable match.

Eg: For the below ACLED article:

assoc_actor1	actor2	assoc_actor2	event_date	location
BJP: Bharatiya Janata Party	Civilianso urndia)	CPI(M): Communist Party of India (Marxist)	23-Mar-19	Basudebpur

Table 2: Example of entities tagged by TL;DR

With the following fields:
Intersect_person - [{'Kamalpur': 1.0}, {'Jitendra': 1.0}]
Intersect_ORG - [{'CPI': 1.0}, {'Kamalpur': 1.0}, {'BJP': 1.0}, {'Police': 0.6666666666666666}]
Intersect_Location - [Odisha, 'Lok', 'Sabha']
Standardized_dates - 03/23/2019

We make the following Evaluation:

assoc_actor1	actor2	assoc_actor2	event_date	location
1	0	1	1	1

Table 3: Manual evaluation methodology

Note: We mark location true because Basudebpur is in Odisha

Similarly, after manual evaluation on all records in out data set, we get the following Accuracies:

Another approach as seen below was to evaluate how "exact" were the entities in comparison to what the ACLED's cells contained. Although the score was disappointingly low, there were several learning outcomes.

Method (b): Edit distance method

i. Compare the ranked list of TL;DR with the fields of ACLED using "Levenshtein" method.

ii. Increment the score if a match occurred.

It was concluded that this kind of evaluation did not perform well because ACLED was human-written. TL;DR works solely on the content written by the journalists.

For example,

ACLED's value for the assoc_actor = "Farmers (India)"

ORG generated by TL;DR = "Bharatiya Kisan Union"

Though "Bharatiya Kisan Union" is actually the relevant organization, it has been reported as "Farmers(India)" by ACLED. So, the relationship needs to be figured by reading the text and then resolving it's meaning. After this, if the context was correct, the score is increased.

Method (c): Context establishment

Libraries used - Huggingface neural-coref, Gensim's word2vec.[3]

1. Train all the documents and build a word2vec model.
2. Pass the value in ACLED's actor and the value in TLDR as parameters to word2vec model.similarity() function and obtain a score.
3. Set a cut off value and choose those terms that crossed the threshold as similar terms. However, this does not guarantee the relationship between two terms. For example, [[0.9994035, 'Sasha', 'college']] - This probably says that the two words are extracted from the same article. We need to know if they are related.
4. Now, check if these terms are correlated by using the coref-resolution library.
5. This library returns terms and their highest scored neighbor terms. These two terms could be highly correlated. Now, we need to backtrace correlated terms by checking which parent term pointed to the current term.
   Basically, we obtained trees of correlated terms.
   [[math, college, Sasha], [She, Sasha], [Thursday, college, Sasha], [college, Sasha], [Sasha]]
   If the two words existed in this list, then they are related.
6. From steps 4 and 5, if two words are actually correlated, it was observed to be the entities of concern. The score was accordingly incremented.

Example:

Sasha went to college on Thursday. She does not like to attend math class. The output of the coref-resolution would be:

[[math, college, Sasha], [She, Sasha], [Thursday, college, Sasha], [college, Sasha], [Sasha]]

From this, we know that "She", "college", "Thursday", "math" are all related to Sasha.

S.no	Location (marked correctly)	Actor (marked correctly)
1.	59/122	62/122

Table 5: Evaluation results for Context establishment method

% accuracy for Location: 48.3%
% accuracy for Actor: 50.8%

5.4 Evaluation of Summarization using Rouge Metric:

The evaluation of the summary generated using Gensim's summarizer was done using the Rouge library in Python. The "notes" field of ACLED was compared to the machine-generated summary.

	Precision	Recall	F Score
Rouge 1	44.85%	19.91%	25.89%
Rouge 2	19.31%	7.91%	10.41%
Rouge L	39.6%	17.65%	18.78%

Table 6: Rouge Scores

As seen in the graphs and table, though the precision score was decent. The recall was low.

5.5 Evaluation of live data

Since there is no gold truth to be matched against, evaluation of live data is an interesting problem. An empirical method for the sake of evaluation of live data was proposed. This is a single strategy for evaluating both the summary and the tagged values. Method for live data evaluation:

- Each tagged value is searched for in the summary.
- If the list for the corresponding entity was empty, then we subtract one from the total sum of the articles present. This means that we are heavily relying on the NER taggers to have tagged that particular entity. If no entity was tagged, then we assume that the article did not mention it.
    - Eg: ORG_SCORE - corresponds to the number of times the tagged organization was present in the summary.
    - ORG_TOTAL - corresponds to the total number of times an organisation was present in the entire article.
    - ORG_percentage: (ORG_SCORE/ ORG_TOTAL) * 100

S.no	Person	Location	ORG
1	89%	58%	92%

Table 7: Live data evaluation results

6 Our Results

Summary: Patna Bihar India, Apr 4 ANI Congress workers created ruckus at the party office here on Thursday in protest against the denial of ticket to former party MP Nikhil Kumar from Aurangabad parliamentary constituency. The workers also shouted the slogan 'Nikhil Kumar Zinadabad.' Kumar was also present in the office when the ruckus took place. Kumar had successfully contested from the seat in 2004 when the Congress fought in alliance with the RJD and the LJP. Kumar, a former Delhi Police Commissioner, unsuccessfully contested from Aurangabad in 2014 against BJP s Sushil Kumar Singh.

Date: 4/4/2019
ranked_list_PERSON: [{'**Kumar**': 1.0}, {'Patna Bihar': 0.625}, {'**Nikhil**': 1.0}]
ranked_list_ORG: [{'**Congress**': 1.0},{'Kumar': 1.0}, {'ANI': 1.0}]
Location: ['India', 'Aurangabad', 'Patna', 'Bihar', 'India', 'Lok', 'Sabha']

From the above example, we can see that for the 'PERSON' entity, terms like 'Nikhil' and 'Kumar' have been given higher weight than 'Patna Bihar'. The date tagged by our system is accurate because 'April 4th' in the text and the 'Thursday' correspond to the same day. Similarly, for the 'ORG', the term 'Congress' has a higher weight

7 Data Pipeline

Figure 4: Data pipeline

The Data Pipeline is divided into the following modules -
1. Logstash :Data is often scattered many systems in many formats. Logstash is a data processing pipeline that ingests data from a multitude of sources simultaneously. By using Logstash, a continuous ingest pipeline is configured, that runs instantaneously on live data.
2. Elastic Search : Elasticsearch is a distributed, RESTful search and analytics engine which stores each news article as a document within an index. The input article text is tokenized and filtered so that it can be searched and ranked by relevance. Further faceting and aggregations are applied so that each field can be broken down and combined to derive insights. [7]
3. Kibana : Kibana is used to visualize the underlying data. Multiple dashboards are created which allows us to discover insights as well as drill down into specific events, using features such as country, organizations involved, and the subcategories generated from the previous steps.
Sample Outputs :Some examples of events that occurred in the previous months to date are shown. In particular, two examples of event representations extracted from news articles successfully linked together by the NER and Topic modeling system are displayed.

1. Phase 1 Elections in India :

The below visualization shows how relevant incidents that occurred during the time period April 9 - April 20 are displayed. We can see that all the events are centered around the states/regions that went to polls around the same time period that we are analyzing.

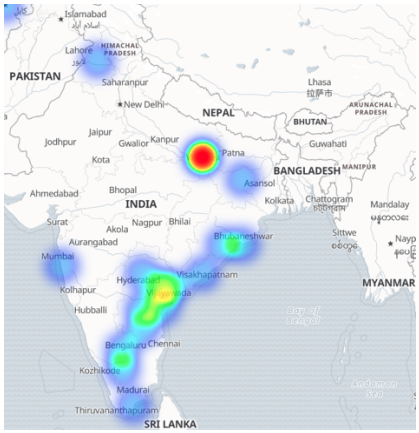

Figure 5: Phase 1 results

2. Pulwama Terrorist Attack:

The below visualization shows the incidents surrounding the terrorist attach that occoured in Pulwama, Kashmir. The system accurately captures the location of the attack, as well as the protest that occoured as a result of this event.

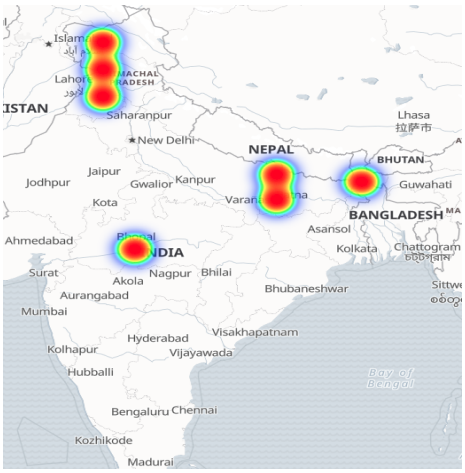

Figure 6: Terrorism in India

8 Conclusion

A new system for News event extraction is described, that can process agency news from multiple news websites. The system has been trained to identify, classify, and resolve 3 types of events, and makes them available to downstream client applications via a continuous streaming mechanism via logstash. The data is then stored as an index via Elasticsearch which helps to search and aggregate the extracted data, Capturing valuable insights. The proposed hybrid coreference with textrank approach allows for more precise Entity Extraction relating to Organizations and People concerned in the event. In future work, the model can be extended to capture events from social media, such as Twitter and also support multiple languages. Since the events are stored as Indexes, the system could also be trained detect duplicate events, using vector similarity.

References

[1] https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/
[2] https://nlp.stanford.edu/courses/cs224n/2008/reports/11.pdf
[3] https://github.com/huggingface/neuralcoref - Neural coref
[4] https://medium.com/district-data-labs/named-entity-recognition-and-classification-for-entity-extraction-6f23342aa7c5 - Ensemble classifier
[5] https://rare-technologies.com/text-summarization-in-python-extractive-vs-abstractive-techniques-revisited/ - Extractive summarization using Gensim's Text Rank
[6] https://towardsdatascience.com/textrank-for-keyword-extraction-by-python-c0bae21bcec0 - Textrank for keywords extraction
[7] https://www.elastic.co/ - Data pipeline
[8] https://www.kaggle.com/snap/amazon-fine-food-reviews/downloads/Reviews.csv/2 - Doc2Vec classification
[9] https://github.com/fhamborg/news-please - News Please crawler