EVENT CLASSIFICATION,DETECTION & SUMMARIZATION

Tannu Priya
Department of Computer Science
University at Buffalo
tannupri@buffalo.edu

Sri Harsha, Kesapragada
Department of Computer Science
University at Buffalo
vkesapra@buffalo.edu

Dr. Rohini K. Srihari
Department of Computer Science
University at Buffalo
rohini@buffalo.edu

ABSTRACT

The combination of large open data sources with machine learning approaches presents a potentially powerful way to predict events such as protest or social unrest. However, accounting for certainty in such models depends on the data particularly when using diverse, unstructured datasets such as social media. A non-governmental organization, The Armed Conflict Location & Event Data Project (ACLED), collates and manually analyzes data on political violence and protest around the world. Manual method of collating data creates a lag of 7 days. In this paper we propose a system that automates ACLED and works on real time data. This system uses TextRank algorithm for Text Summarization, Naive Bayes machine leaning model for classification and event detection task. To extract entities out of text, we used a tool called SPACY and to get accurate event date we used Stanford Temporal Tagger.

Keywords

Text Summarization, WordNetLemmatizer, Naïve Bayes, Latent Dirichlet Allocation, SUTime, Text Rank, Spacy, entity tagging, Glove, Word2Vec, ROUGE.

1. INTRODUCTION

With the enormous increase of digitization of news articles, the task of mining these articles has become a crucial tool for aiding and understanding the information. This includes clustering, classification, categorization and summarization. The major challenge is to find relevant information from large amounts of data. News Article categorization and summarization is an effective technique that when used in combination with Information Retrieval and Information filtering systems saves a lot of user time and provides useful insights to predict social unrest events. Rather than reading the entire news article and then gaining the insight, it will be more beneficial to go through the summary of the article and still gain the theme or core information present in the article. In this way more and more information can be gathered in less time.

Now-a-days there are plenty of online news websites overwhelmed with news articles. The most important tasks of news engines are Collecting News, News Retrieval, Categorizing Search Result, Summarization and Automatic Event Detection. The quality of each of the tasks depends on the quality of several other tasks. This paper focuses on gathering insights through several content mining techniques about global social unrest events. We aim to provide insights on how much these global events can be attributed to elections. We present a simple architecture where the system takes as input from the user a particular date and gather all the news articles from the web. The articles thus gathered will be categorized into Riots/Protest or Violence against Civilians and the rest of the articles are discarded. The articles that are categorized into the above two categories will then be summarized and tagged and will be presented to the user.

2. SYSTEM DESCRIPTION

Figure.1 presents an overview of the system architecture. Initially user enters the date for which he/she wants the information. This date is then passed to document collector that retrieves the resultant news pages from the web. These articles are fed to the classification pipeline which categorizes the articles into Riots/Protests and Violence against Civilians and discard the irrelevant news articles. These articles will then be passed to summarizer which then applies the famous Text Rank algorithm to rank all the sentences in an article and retrieve the top three ranked sentences as the summary. Now since these sentences are ranked according to the order or relevance, we use this summary generated by the summarizer to extract the event date, event location and the parties involved in the event. These extracted results are then displayed to the user. We also analyzed these results from our system to gather insights on how we can attribute these events to elections happening in a country. For this purpose, we targeted the time period around phase-1 elections for India. This can be easily extended to other countries and provide insights on a global scale.

3. DOCUMENT COLLECTION

Webhose.io news API data feeds crawl and index regularly updated web content from a huge repository of news sources – including blogs, forums, news articles, and radio stations – in addition to a massive repository of historical data. A large-

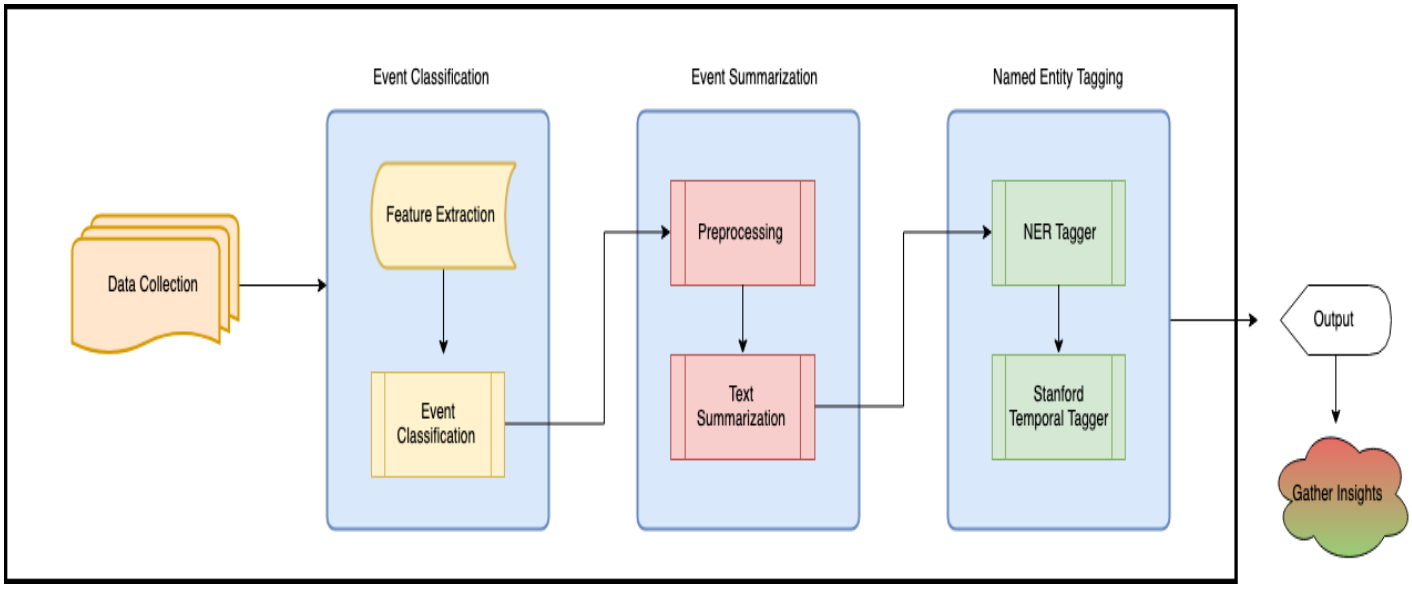

Figure 1: System Architecture Overview

scale news feed from this API and some other articles from News API is used as input to the news event extraction pipeline. The user enters date through the user interface. This date is passed to news collector that retrieves the all the resultant news pages from the aforementioned new API's.

4. EVENT CLASSIFICATION

Most of the news services today are electronic in nature in which a large volume of news articles is created every single day by the organizations. In such cases, it is difficult to organize the news articles manually. Therefore, automated methods can be very useful for news categorization in a variety of web portals.

One of the most fundamental tasks that needs to be accomplished in text classification is that of feature extraction. While feature extraction, also called as feature selection is relevant in other classification tasks, it is especially important in text classification due to the high dimensionality of text features. In general, text can be represented as a bag of words, in which a document is represented as a set of words, together with their associated frequency in the document. Such a representation is essentially independent of the sequence of words in the collection.

For our system, we used TF-IDF Vectorizer, which is form of bag of words to extract the features out of the text. TF-IDF score represents the relative importance of a term in the document and the entire corpus. TF-IDF score is composed by two terms: the first computes the normalized Term Frequency (TF), the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

Before we extract the features for classification, we clean the existing dataset of stop words. This crucial step has many advantages in any text mining technique one might use. It reduces memory overhead (since we eliminate words in consideration) and reduces noise and false positives (since we are focusing on the more important terms). It can potentially improve power of predictions. We also performed stemming, where different forms of the same word are consolidated into a single word.

As training data, we used all the Riots/Protests and Violence Data from the Armed Conflict Location & Event Data Project (ACLED) dataset which comprises of around 10000 articles for the last three years. We used the famous Naïve Bayes Classification algorithms which has been successfully applied to document classification in many research efforts. One of the major advantages that Naive Bayes has over other classification algorithms is its ability to handle an extremely large number of features. In our case, each word is treated as a feature and there are thousands of different words. Also, it performs well even with the presence of irrelevant features and is relatively unaffected by them. The other major advantage it has is its relative simplicity. Naive Bayes' works well right out of the box and tuning it's parameters is rarely ever necessary, except usually in cases where the

distribution of the data is known. It rarely ever overfits the data. Another important advantage is that its model training and prediction times are very fast for the amount of data it can handle.

All news articles that are processed are classified by a supervised classifier that discards or assigns an event type to them. We assume that each new article contains 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 ∈ {'Riots/Protests', 'Violence against Civilians' and 'None'}.

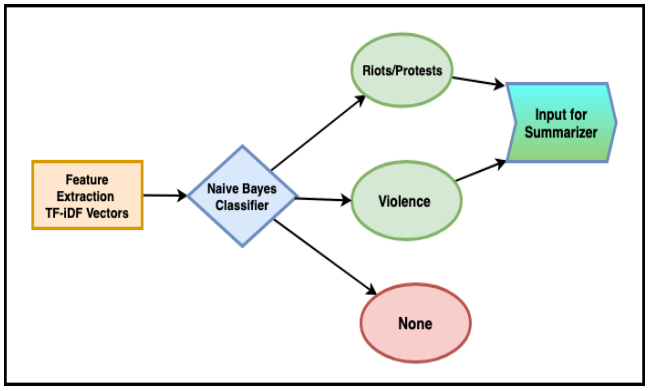

Figure 2: Event Classifier Overview

5. AUTOMATIC TEXT SUMMARIZATION

Automatic Text Summarization is one of the complex and interesting problems of Natural Language Processing. For generating meaningful summaries out of the news article we used Text Rank Algorithm. PageRank (Brin and Page, 1998) is perhaps one of the most popular ranking algorithms, and was designed as a method for Web link analysis. Unlike other ranking algorithms, PageRank integrates the impact of both incoming and outgoing links into one single model, and therefore it produces only one set of scores:

Text summarization can broadly be divided into two categories:

- Extractive Summarization: These methods rely on extracting several parts, such as phrases and sentences, from a piece of text and stack them together to create a summary. Therefore, identifying the right sentences for summarization is of utmost importance in an extractive method.

- Abstractive Summarization: These methods use advanced NLP techniques to generate an entirely new summary. Some parts of this summary may not even appear in the original text.

TextRank is an extractive and unsupervised text summarization technique. We are first preprocessing all the news articles for removing stop words and we also lemmatized the entire text using WordNetLemmatizer. This processed text is then split in individual sentences and we further used word embeddings to find vector representation of every sentence. Similarities between sentence vectors is then calculated and stored in a matrix. This similarity matrix is then converted into a graph, with sentences as vertices and similarity scores as edges, for sentence rank calculation. Finally, we used top two sentences form the final summary.

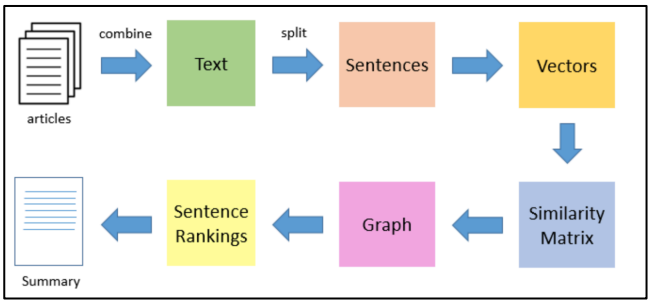

Figure 3: Architecture of Text Summarizer

5.1 Execution Steps:

5.1.1 Text:

The data from the classifier is fed in the Text summarization system and we execute preprocessing on this raw text. Preprocessing is crucial for the correct text ranking because the stops words should not affect the similarity matrix and graph computed on the later stages of the algorithm.

5.1.2 Sentences:

In the next step of processing we have broken the entire text in individual sentence using sent_tokenize( ) function of the nltk library.

5.1.3 Vectors:

We are processing sentences to vectors using GloVe word embeddings. GloVe word embeddings are vector representation of words. These word embeddings will be used to create vectors for our sentences. We could have also used the Bag-of-Words or TF-IDF approaches to create features for our sentences, but these methods ignore the order of the words (and the number of features is usually pretty large). We have used the pre-trained Wikipedia 2014 + Gigaword 5 GloVe vectors.

5.1.4 Similarity Matrix:

In next step, we found the similarities between the sentences using the cosine similarity. We populate a matrix with each entry in the matrix representing the similarity between the corresponding sentence pair.

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|}$$

5.1.5 Graph Computation:

This similarity matrix is then converted into a graph. The nodes of this graph represent the sentences and the edges represent the similarity scores between the sentences. On this graph, we applied the PageRank algorithm to arrive at the sentence rankings.

5.1.6 Sentence Ranking:

The graph computed is highly connected, with a weight associated with each edge, indicating the strength of the connections established between various sentence pairs in the text. The text is therefore represented as a weighted graph, and consequently we are using the weighted graph-based ranking.

After running the ranking algorithm on the graph, sentences are sorted in reversed order of their score, and we selected the three top ranked sentences for inclusion in the summary.

6. Named Entity Tagging

Named Entity Recognition, also known as entity extraction classifies named entities that are present in a text into pre-defined categories like "individuals", "companies", "places", "organization", "cities", "dates", "product terminologies" etc. It adds a wealth of semantic knowledge to your content and helps us to promptly understand the subject of any given text. For our purposes, we extracted event date, location and parties involved from each article to be presented to the user.

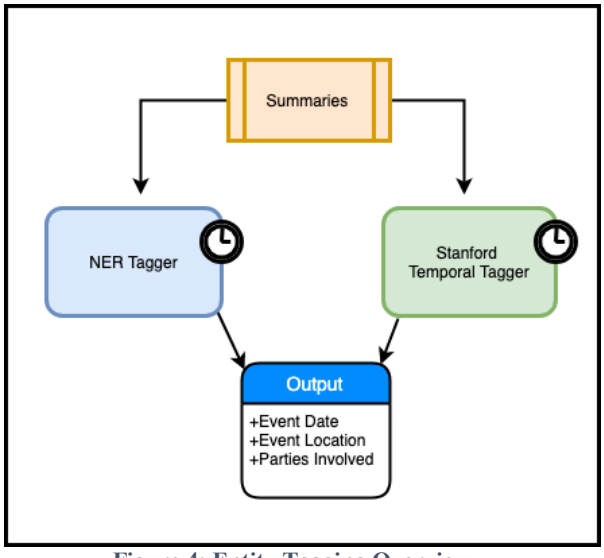

Figure 4: Entity Tagging Overview

[1] Angel X. Chang and Christopher D. Manning. 2012. SUTIME: A library for recognizing and normalizing time expressions. In LREC 2012.

6.1 Event Location and Parties Involved:

For extracting entities like Event Location and Parties involved, we leveraged **Spacy** which is an open-source software library for advanced Natural Language Processing, written in the programming languages Python and Cython. It comes with well-engineered feature extractors for Named Entity Recognition, and many options for defining feature extractors. One of the most powerful features of spacy is the extremely fast and accurate syntactic dependency parser which can be accessed via lightweight API. The parser can also be used for sentence boundary detection and phrase chunking. The relations can be accessed by the properties ". children", ". root", ". ancestor" etc. Spacy also provides inbuilt integration of dense, real valued vectors representing distributional similarity information. It uses GloVe vectors to generate vectors. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Spacy is very efficient in detecting named entities and hence, we opted it for the task of detecting Event location and parties Involved.

6.2 Event Date:

For detecting Event date, we are **using Stanford Temporal Tagger: SUTIME**[1]. SUTIME is a rule-based temporal tagger built on regular expression patterns. Temporal expressions are bounded in their complexity, so many of them can be captured using finite automata. SUTIME uses a regular expression language for expressing how text should be mapped to temporal objects. SUTIME is built on top of TOKENSREGEX, a generic framework included in Stanford CoreNLP for defining patterns over text and mapping to semantic objects. Using this framework, rules for how to map text to the temporal objects provided by the SUTime Java library are specified. Temporal tagger takes reference date as the input and would use the regular expressions to compute the exact date. We are using the input from the user as the reference date for this purpose. SUTime extracts normalizing temporal expressions and then maps text to temporal objects. Below is a quick result of SUTime.

Expression	*SUTime*
this week	1989-W43

7. Analysis

To gather some insights on how our system was performing and how we can attribute the unrest events that our system

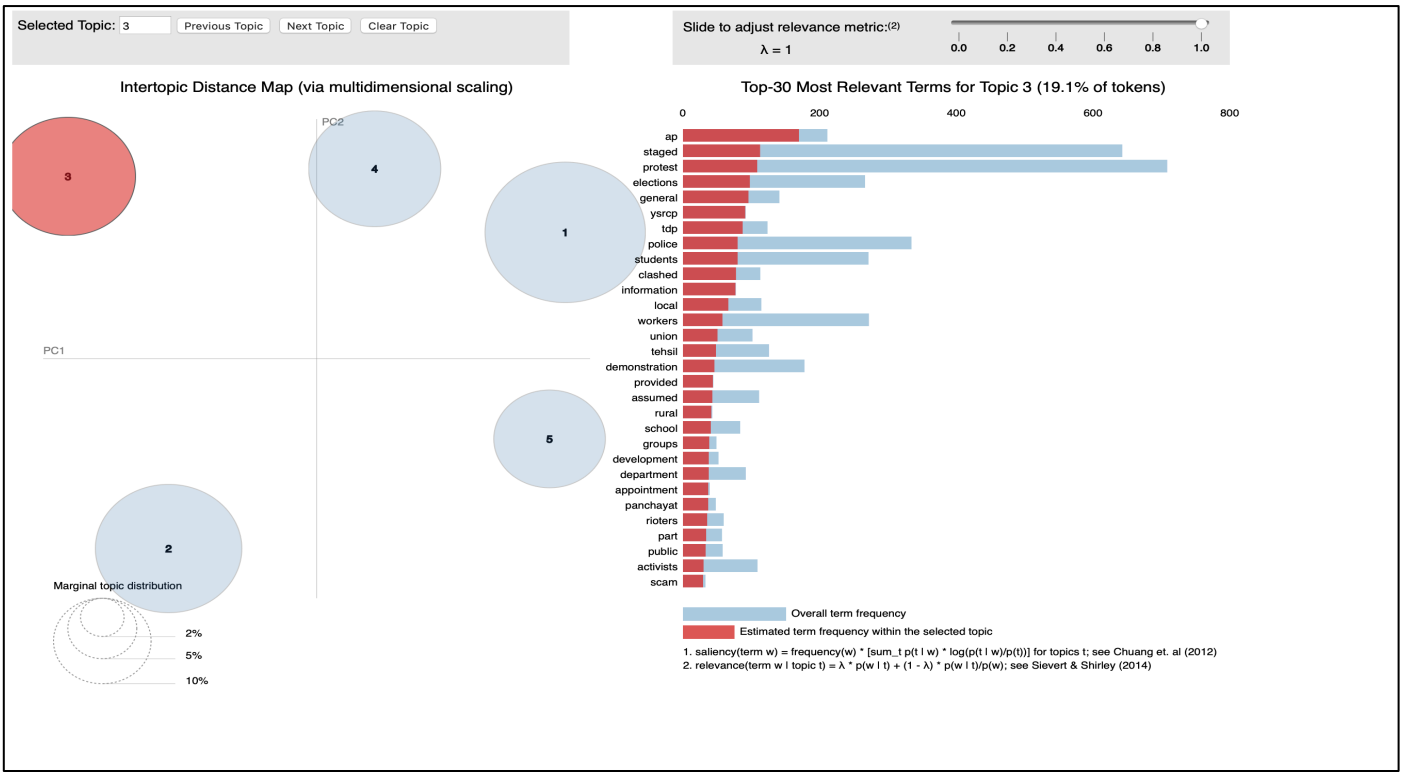

Figure 5: Topic Modelling for News Articles from India

classified to elections, we used LDA topic modeling to find the hidden topics. **Topic modeling** is a type of statistical modeling for discovering the abstract "topics" that occur in a collection of documents. **Latent Dirichlet Allocation (LDA)** is an example of topic model and is used to classify text in a document to a particular topic. It builds a topic per document model and words per topic model, modeled as Dirichlet distributions.

Figure 5 shows the topic modelling for various articles that are classified as Riots/Protests during the time period of phase 1 elections from India. As we can see around 20% of the articles fall under topic 3. If we look at the most prevalent words in this category, we can find words related to elections and other words such as TDP and YSRCP which are regional parties in and around the places where phase 1 elections happened around that time. So, we can attribute almost 20% of these unrest events around that area to the general elections. Figure 6 shows a heatmap of India with the number of riot events that happened as the elections are approaching. As the elections are approaching a greater number of events are happening in those states that are going for elections during the phase 1. This shows a strong correlation between proliferation of unrest events and elections in most of the countries

8. Evaluation

To evaluate our system, we used ACLED dataset as the benchmark. We have manually collected over 100 articles, ACLED has summarized and categorized and used our system to classify and summarize these articles. We employed different metrics to compare different modules. For Text Summarization, we used ROUGE metric to evaluate machine generated summaries with that of ACLED summaries.

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. ROUGE**2** automatically determine the quality of a summary by comparing it to ACLED summaries created by humans (Reference). The measures count the number of overlapping units such as n-gram, word sequences, and word pairs between our system generated summary and the ideal summaries created by ACLED. Below is the detailed description of this matric along with the results:

8.1.1 ROUGE-N: N-gram Co-Occurrence: We are taking computing N-gram (here, n=1) overlaps for every summary.

[2] C. yew Lin. Rouge: a package for automatic evaluation of summaries. pages 25–26, 2004.

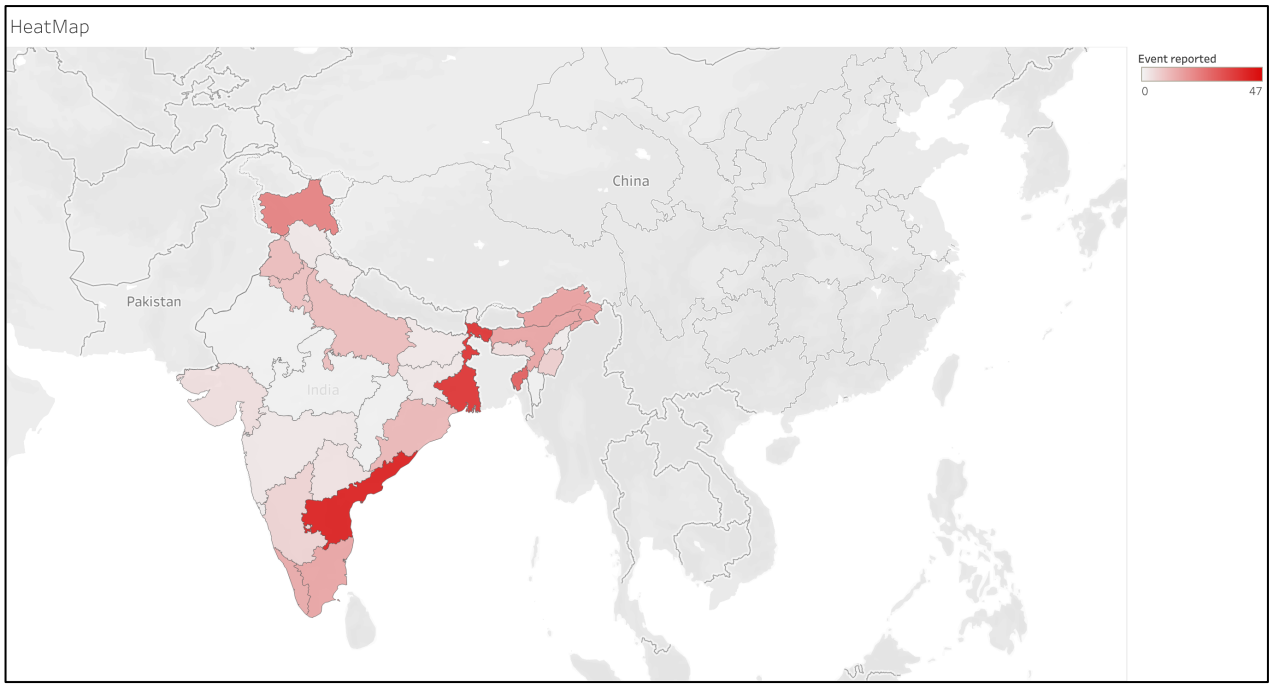

Figure 6: Heatmap of India showing Riot/Protest Events

Then we are computing average of all the individual ROUGE SCORES to gauge how well the text summarizer is working.

$$\text{ROUGE-N} = \frac{\sum_{S \in \{ReferemceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

8.1.2 *ROUGE-L: Longest Common Subsequence*: In Longest common subsequence we are taking into account sentence level structure similarity and it identifies longest co-occurring in sequence n-grams automatically.

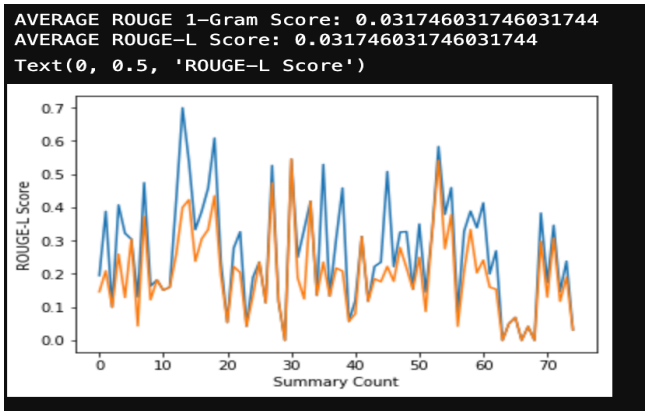

Figure 7: ROUGE SCORE

For event classification we used the same 100 articles that ACLED categorized as either Riots/Protests or Violence. The graph below shows the confusion matrix on how well the classifier has performed.

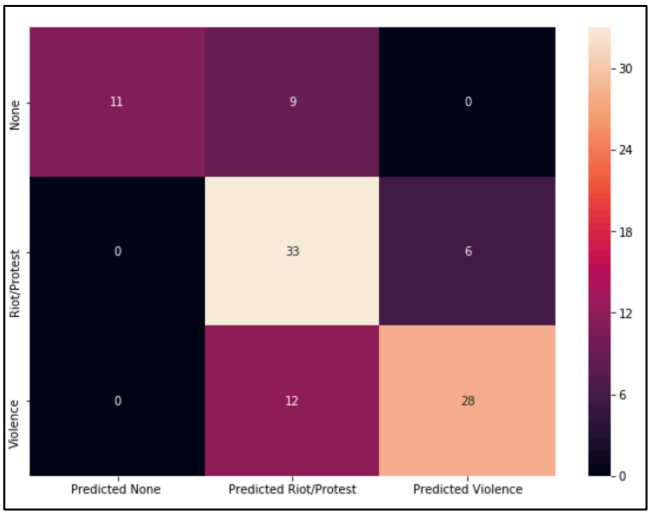

Figure 8: Confusion Matrix For the Classifier

Since, ACLED's entity tagging in terms is not so precise, we did a manual evaluation of how well our taggers are working and are able to get almost 95% accuracy in terms of predicting event date and 60% of the times, we are able to accurately identify the location.

9. CONCLUSION AND FUTURE WORK

We have described and Implemented a system that would bring structure to the news articles and could act as a centralized noise free database for the events reported under the 2 categories of Riots/Protest and Violence against Civilians. Our System could classify if any article lies under these two categories and would extract important entities like Location, Date and Parties involved from the event reported. It would also give an extractive summary for a concise understanding of the event. This system efficient works on real time data.

In future work, we plan to implement Wikification, which is the process of annotating the mentions of concepts in a document with the URL of the Wikipedia page about that concept. Also, for Event classification: Since we are limited by the number of training samples from ACLED for a particular country, we are planning to incorporate more samples to train our classifiers using other conflict databases available online. Datasets like Uppsala Conflict Data Program (UDCP) and Urban Social Disorder Dataset (USD) have public datasets available and we are planning to use these datasets along with ACLED for our final evaluation. For Text Summarization, we are planning to optimize our summarization using Adapted TextRank1 for Term Extraction which is a Generic Method of Improving Automatic Term Extraction Algorithms. For Entity Identification, our model dependents on the quality summary. Improving on Text summarization will help us improve data for entity Tagging

10. ACKNOWLEDGMENTS

The authors would like to thank Dr. Rohini K. Srihari and her PHD student Lu Meng, for their continuous support and guidance.

11. REFERENCES

[1] C. yew Lin. Rouge: a package for automatic evaluation of summaries. pages 25–26, 2004.

[2] Angel X. Chang and Christopher D. Manning. 2012. SUTIME: A library for recognizing and normalizing time expressions. In LREC 2012.

[3]] Steinberger, J., Ježek, K.: Evaluation measures for text summarization. Computing and Informatics 25, 1001–1025 (2012)

[4] Fabrizio Sebastiani, Machine learning in automated text categorization, ACM Computing Surveys (CSUR), v.34 n.1, p.1-47, March 2002 [doi>10.1145/505282.505283].

[5] Barzilay, R., & Elhadad, M. (1999). Using Lexical Chains for Text Summarization. In Mani, I., & Maybury, M. T. (Eds.), Advances in Automatic Text Summarization, pp. 111–121. The MIT Press.

[6] Nallapati, Ramesh, Zhou, Bowen, Gulcehre, Caglar, Xiang, Bing, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. In Proc. of EMNLP, 2016.

[7] Steinberger, J., Ježek, K.: Evaluation measures for text summarization. Computing and Informatics 25, 1001–1025 (2012).

[8] Rish, I.: An Empirical Study of the Naive Bayes Classifier. In: IJCAI 2001 Workshop on Empirical Methods in AI (2001)

[9] Barzilay,R.,&Elhadad,M.(1999).UsingLexicalChainsforText Summarization.InMani,I., & Maybury, M.T. (Eds.),Advances in AutomaticTextSummarization, pp.111{121.TheMITPress.

[10] Ferreira R, De Souza L, Dueire R et al (2013) Assessing sentence scoring techniques for extractive text summarization. Expert Syst Appl 40:5755–5764.

[11] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 55–60.

[12] Raleigh C, Hegre H. 2005. Introducing ACLED: an armed conflict location and event dataset. Work. Pap., Int. Peace Res. Inst., Oslo

[13] Blei et al., 2003 D.M. Blei, A.Y. Ng, M.I. Jordan Latent dirichlet allocation J. Machine Learn. Res., 3 (2003), pp. 993-1022

[14] McKenzie and Janowicz, 2015 G. McKenzie, K. Janowicz Where is also about time: a location-distortion model to improve reverse geocoding using behavior-driven temporal semantic signatures Computers, Environment and Urban Systems, 54 (2015), pp. 1-13

[15] https://www.researchgate.net/publication/326904600_A_Generic_Method_of_Improving_Automatic_Term_Extraction_Algorithms