A BIG DATA FRAMEWORK FOR UNSTRUCTURED TEXT PROCESSING WITH APPLICATIONS TOWARDS POLITICAL SCIENCE AND HEALTHCARE

by

Sayeed Salam

APPROVED BY SUPERVISORY COMMITTEE:

Latifur Khan, Chair

Dohyeong Kim

Farokh Bastani

Weili Wu

*This dissertation is dedicated to*
*my family.*

# A BIG DATA FRAMEWORK FOR UNSTRUCTURED TEXT PROCESSING WITH APPLICATIONS TOWARDS POLITICAL SCIENCE AND HEALTHCARE

by

SAYEED SALAM, BS

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

December 2021

# ACKNOWLEDGMENTS

I would like to thank a number of people in my academic, social and family life who have supported me throughout my tenure as a research assistant here at UTD. First, I would like to thank my supervisor Dr. Latifur Khan for his continuous guidance and support towards fulfilling my research objectives. I am really thankful to my labmates at Big Data Analytics and Management lab. Throughout my academic period, I learned a lot from them and enjoyed their company. I would like to thank my committee members, Dr. Dohyeong Kim, Dr. Farokh Bastani and Dr. Weili Wu for evaluating my dissertation. I would like to thank my parents, my brother and other family members for playing strong supporting roles.

Finally, I would like to thank NSF and NIH for sponsoring my research.

November 2021

A BIG DATA FRAMEWORK FOR UNSTRUCTURED TEXT PROCESSING WITH APPLICATIONS TOWARDS POLITICAL SCIENCE AND HEALTHCARE

Sayeed Salam, PhD
The University of Texas at Dallas, 2021

Supervising Professor: Latifur Khan, Chair

Machine learning and deep neural networks have soared in popularity in recent years, allowing us to enhance many aspects of everyday life. While these methods are intuitive, they are very reliant on the dataset being used to build the model. A high-quality dataset boosts the model's accuracy and validates the model's output in the context of a real-world scenario. Furthermore, continuous improvement on the dataset contributes in the tuning of the model in a time-consistent way and the mitigation of temporal inconsistencies. However, preparing datasets, particularly for text domains, is difficult due to the inherent unstructured nature of the data and the use of multiple languages. Furthermore, the amount of text produced in the form of news articles or social media posts is massive, necessitating large-scale processing. The velocity at which new texts are produced demands an elastic and scalable system that can accommodate any surge of inputs while remaining resource efficient while not in use. Texts are created in a variety of ways and must be preprocessed and analyzed in order to provide well-structured, consistent data. This can be accomplished through the use of a well-defined domain-specific ontology (rule-based approach) or machine learning approaches. While rule-based systems can provide information that are more precise and are preferred in a variety of circumstances, they lack flexibility as the ontologies are often fixed and does not respond well with the continuous changes in respective domains. We propose associated

solutions to the challenges described above in this dissertation. First, we go over a scalable architecture for collecting news stories from around the world and utilizing a rule-based approach with the Conflict and Mediation Event Observation(CAMEO) ontology to generate political events. We present a summary of the generated dataset, as well as some basic analysis, to demonstrate how it relates to the real-world scenario. We present techniques to dynamically adding information to the ontology using a mining approach for discovering new political actors that works as a recommender system and retrieves more than 80% of the missing information including political figures and their roles. We discuss an extended data processing system for processing articles published in several languages, with a focus on translation methodologies and tools developed. In comparison to the English language, we demonstrate the efficacy of the coder in Spanish. When compared to equivalent events in English articles, the revised event coder with translated knowledge-base was able to recognize 83% of information in Spanish.

For healthcare, we propose an alternative strategy in which we use several machine learning algorithms and social media, such as tweets, to extract the location and severity of Road Traffic Incidents (RTI). We highlight a pipeline that goes from collecting tweets to summarizing related tweets for an RTI. We also demonstrate how semi-automatic ontology learning can be useful in determining severity and offer a simplified example in which 100% of the target rules were identified using an iterative technique.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

# CHAPTER 1

# INTRODUCTION

Online news articles have been increasingly replacing print media in recent years. Social media also allows users to access enormous quantities of data that has been curated by the users themselves. These media are now producing massive amounts of text data, which, if properly processed, will yield a big dataset of computer-interpretable data. We will be able to collect and evaluate global social and economic phenomena using this method, and we will be able to model present and future events with existing computing capabilities.

The main issues encountered throughout the data collection process are directly tied to big data-specific challenges. There are numerous news stories and social media posts that must be processed through (high volume). New texts are published on a regular basis, resulting in bursts of new information (i.e high impact social and political events, natural calamities, etc.) This type of high-speed data necessitates a processing system that is scalable and elastic in nature, minimizing resource utilization when not in use while still being able to manage data surges when needed. Furthermore, people frequently disseminate information in numerous languages, necessitating the deployment of appropriate knowledge-base or machine learning models. To accomplish this, we should translate existing knowledge bases in order to capture new information in languages other than English.

We typically use a distributed solution when dealing with large amounts of data. For prototyping the solution, regular multiprocessing modules supplied by programming languages can be utilized, however this adds overhead of process tracking, node management, and failure handling. We employ Apache Spark (for distributed computing) and Kafka (distributed streaming) in the frameworks built in successive chapters to eliminate these overheads in system architecture. To extract structured metadata from news articles and social media posts, we use Stanford CoreNLP and Universal dependency parsers.

In subsequent sections, we will briefly introduce problems we address in different chapters and describe corresponding solutions/frameworks.

## 1.1 Distributed Framework for Political Event Coding in Real-Time

Studying political activities and interaction between different entities gradually became a prominent field of research for both the social science and computer science researchers. Research is being carried out at local (limited to a particular region) and global scale, often divided in temporal manner. It is also useful to have a most recent dataset to have an up-to-date analysis. For these purposes, we need timestamped, structured information about political interactions. Keeping this in mind, we develop a distributed framework that works in real-time with Apache Kafka and SPARK for processing a global collection of news data in different languages (i.e., Spanish, Arabic) and generate those structured data. We also provide a API for easy access to the data. In this chapter, we will describe how the system works, how to access the data and possible analytical problems that can be addressed by building a model on the dataset.

## 1.2 RePAIR: Recommend Political Actors In Real-time From News Websites

Extracting a structured representation of political events from news reports is at the intersection of the computational and social sciences. A traditional approach is to use dictionary-based pattern lookups to identify actors and actions involved in potential events. A key complication of this approach is updating the dictionaries with new actors (e.g., when a new president takes office). Currently, the dictionaries are curated by humans, updated infrequently, and at a high cost. This means that tools dependent on the actor dictionaries (e.g., PETRARCH) overlook events when actors are missing in the dictionary. Since these tools use only the syntactic structure of the sentence (e.g., parse tree, etc.) for their event cod-

ing, missing actors will generate events which fail to capture actual political interaction. To overcome these issues, we propose a framework *RePAIR* to recommend new political actors in real-time from the political news articles with RSS feeds related to national/international politics across the globe. The framework identifies the semantic structure of a sentence using an Automatic Content Extraction (ACE) method and uses a frequency based actor ranking algorithm to recommend the most frequent new political actors over multiple time windows. We also suggest the associated role of recommended new actors from the role of co-occurred political actors in the existing CAMEO actor dictionary. Further we integrate an external knowledge base (e.g., Wikipedia) into our framework to capture the evolving roles of existing actors over time and recommend new roles for them. Furthermore, we consider PETRARCH and BBN ACCENT event coders for actor recommendation, and a graph-based actor role recommendation using weighted label propagation as baselines and compare them with our framework. Experimental results show our approaches outperform them significantly.

### 1.3 Automatic Event Coding Framework for Spanish Political News Articles

Today, Spanish speaking countries face a widespread political crisis. These political conflicts are published in a large volume of Spanish news articles from Spanish agencies. Our goal is to create a fully functioning system that parses realtime Spanish texts and generates scalable event code. Rather than translating Spanish text into English text and using English event coders, we aim to create a tool that uses raw Spanish text and Spanish event coders for better flexibility, coverage, and cost.

To accommodate the processing of a large number of Spanish articles, we adapt a distributed framework based on Apache Spark. We highlight how to extend the existing ontology to provide support for the automated coding process for Spanish texts. We also present

experimental data to provide insight into the data collection process with filtering unrelated articles, scaling the framework, and gathering basic statistics on the dataset.

## 1.4 Exploring the roles of social media data to identify the locations and severity of road traffic accidents

People tend to use social media to share information about the event that occurred nearby, including traffic accidents. Traffic accident reporting over the phone can initiate medical aid but often fail to correctly specify severity, location, and assessment of the overall situation. Social media information (i.e., tweets, posts, etc.) can be mined to extract supportive information to be used to improve reporting accuracy and reduce response time of first responders. In this chapter, we develop a framework that can continuously analyze and extract relevant accident reports and tested it using the data from four cities in the U.S. and Nigeria. In this framework, we collect tweets from Twitter API, identify whether they are accident-related or not, create clusters of tweets talking about the same accident, and perform a severity analysis based on the summary of the tweets. We then geolocate the accidents for which the location is mentioned (i.e. direct geo-coding) or provide an approximate location for accidents by estimating user location-based twitter feeds (i.e. indirect geo-coding). We also use a semantic role labeling approach for severity detection and present the accuracy with respect to annotated data. The results of empirical testing reveal that city-level locations were identified for 71-97% of the accidents and geo-coordinates were obtained for 33-83% of the accidents, varying across the study sites and geolocation methods. Our framework demonstrates that on average 9-11% cases social media precedes on publishing accident related information than that of actual police reports. We also discuss our approach of using Distributed, Big Data frameworks to process a large number of Tweets generated in streaming manner.

## 1.5 Organization of Dissertation

The rest of the dissertation is organized in the following way. Chapter 2 introduces a distributed processing pipeline for Political Event Coding. Chapter 3 introduces a recommendation based approach to identify potential new political actors and their roles through mining news articles. Chapter 4 highlights steps taken to do multilingual event coding. Chapter 5 discusses a system that captures location and severity related to road traffic incidents from social media messages (ie. tweets). Finally, chapter 6 draws concluding remarks and highlights future direction of works.

# CHAPTER 2

# DISTRIBUTED FRAMEWORK FOR POLITICAL EVENT CODING IN REAL-TIME

## 2.1 Introduction

Current world affairs related to politics is becoming complicated day by day. Study in this field requires lot of data to build a good analytical model. Due to the interest from political science professionals and researchers, generating the relevant data becomes an appropriate problem to address.

One key area of information extraction related to political interactions is Structured Event Data Generation. It is a process to convert unstructured text data to computer friendly structured events. The process uses fixed ontology[51] to identify critical information from a sentence, like who is acting, who is being acted upon and what type of action is happening . This is called *who-did-what-to-whom* format. One such ontology is Conflict and Mediation Event Observations (CAMEO)[40] and defined by dictionaries. There are automated event coders like PETRARCH[66] which is used to generate events using pattern lookup and entity lookup from the those dictionaries.

PETRARCH requires Parts-of-Speech tagged sentences. We apply Stanford CoreNLP[60] to generate these formatted text data. As CoreNLP is compute intensive, we need to adapt a distributed solution for dividing the load of processing in different nodes.

News articles are being published in continuous manner around the world. If we want to get a real-time comprehension of what is going around, we need to build a real time data processing framework where we can collect data in real-time, process them with the tools and present them to the user for visualization[9] and more complex analytical tasks. Unavailability of such a system hinders researchers to carry on their related tasks and demerit the efficiency of ontologies built to capture these informations.

Easy access to the dataset is important for getting the required data for processing. People from all levels of technical background should be able to easily access data with no to little training on the system.

Focusing on these key points, we develop a real-time data processing framework that collects the news articles from the web, process them with Stanford CoreNLP and PETRARCH event coder. We also build an API for easy access to the generated events in real-time.

The major contributions compiled in this chapter are as follows -

- We present a real-time data processing framework built on Apache SPARK and Kafka to generate political event data.

- We describe a rich API for accessing the generated event data.

- We highlight some example scenario where analytical models can be built using the data we have.

The rest of the chapter is organized as follows- Section 2.2 we present related works. Section 2.3 describes related helpful information on tools, ontologies, etc. Section 2.4 describes the architecture of the system. Section 2.5 briefly shows how to access the system and query for data. In section 2.6, we discuss about the multilingual event coder and how it works. Section 2.7 highlights the system specification used for the proposed framework. Section 2.8 lists the available datasets being distributed through the API. In section 2.9, we show some statistics related to data produced by the framework. Section 2.10 lists some possible scenario, where we can reuse the framework and the data to address different social and political problems.

## 2.2 Related Works

In this section we present the related works with respect to the contents of the chapter. Distributed processing of large data has been addressed by [83], [44] where author address a static dataset and the process of generating events. Here, we are working on a real-time dataset and here we need to collect and distribute the data in real-time where aforementioned works only concentrate on processing the data. We are inspired by the scalability analysis found here [83] and incorporate that to design the system. Schordt[78] has pointed out the importance of having a real-time event coding framework and data distribution. Automated Political event coding has gone through several decades and several ontologies, datasets and tools are developed. We use CAMEO ontology here. Other related ontologies are WEIS and authors[40] has provided a comparative study between those ontologies. Eck[35] also present an analysis among different conflict data-sets.

Among the available datasets ICEWS[20], Cline Center Dataset[14] are prominent. But they are not easily accessible and updated in-frequently. In terms of event coder, we found PETRARCH2 is the most flexible, easy-to-extend compared to others (i.e., BBN Accent) as reported by [17, 86]. We are also using a extended version of the event coder which uses Universal Dependency [61] to better support towards foreign languages. Such type of extension was not possible to BBN Accent like proprietary software.

## 2.3 Background

To help the reader better understand the tools and methods used in this chapter, we will provide key details of different components.

**Stanford CoreNLP** is a natural language processing tool and used for generating tagged version of regular text sentence. It can generate Parts-of-Speech (POS) tags, lemma, Named-Entity-Recognition (NER) tags, tokens, dependency diagrams, etc.

**Apache SPARK** is a distributed in-memory framework for large scale data processing.[102]

**Apache Kakfa** is highly efficient distributed message oriented queue management system, useful for synchronizing input-output between different components of a distributed framework.[92]

**Political Event** is a structured piece of information consisting of an action, source (acting entity), target (entity being acted upon) and other related information. For example consider the following sentence from a news article -

```
PM Theresa May has struck a last-minute
deal with the EU in a bid to move Brexit
talks on to the next phase.
```

As a structured event, it looks like the following

```
Source - GBRGOV,
Target - IGOEUREEC
Action - 057 (Sign formal agreement)
```

PM Theresa May and EU is coded in standard format used for event generation. The structure used here is mostly known as *who-did-what-to-whom* format of event coding. The coding mechanism is depicted in the Figure 2.1.

Given a sentence, the encoder search for a matching pattern in the CAMEO verb patterns dictionary. A pattern consists of a verb and surrounding keywords. Together they signify a particular course of action and represented with event code. For example, SET in the following pattern

```
SET OUT VIEWS
```

indicates MAKE PUBLIC STATEMENT type of event (event code 010). Upon finding a match in pattern, actor dictionaries are searched for matching entities representing source

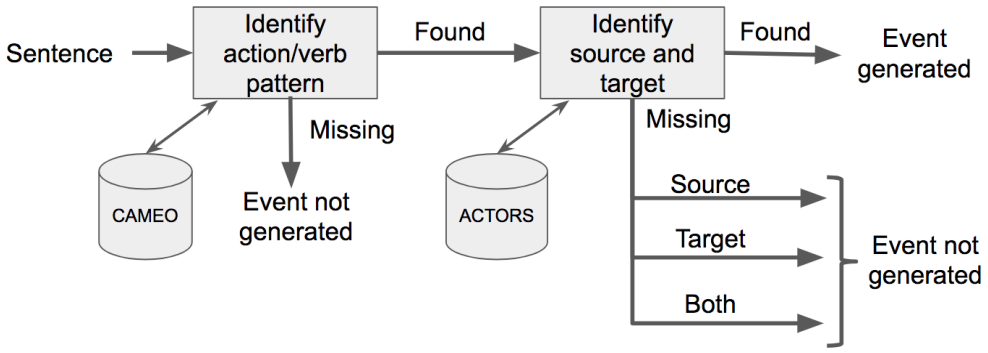



Figure 2.1: Basic Mechanism of Automated Coding

and target. After finding all the pieces of information, an event is coded by PETRARCH. If there is missing information, PETRARCH will ignore the event. This form of events are called Source-Action-Target or SAT format.

**Conflict and Mediation Event Observations (CAMEO)** is an ontology developed to capture political events and focuses on the following 4 categories

- Verbal Cooperation
- Material Cooperation
- Verbal Conflict
- Material Conflict.

It works using a knowledge-base which consists of pattern and actor dictionaries. Pattern dictionary is helpful for identifying political interactions in a given sentence. Actor dictionary is used for searching political actors found around the matched pattern, within a sentence.

**Universal Dependency (UD)** [61] is an initiative that is developing cross-linguistically consistent tree-bank annotation for many languages, with the goal of facilitating multilingual parser development, cross-lingual learning, and parsing research from a language typology perspective. The general goal is to provide a universal collection of categories and guidelines to facilitate consistent annotation of similar constructions across languages, while allowing language-specific extensions when necessary.

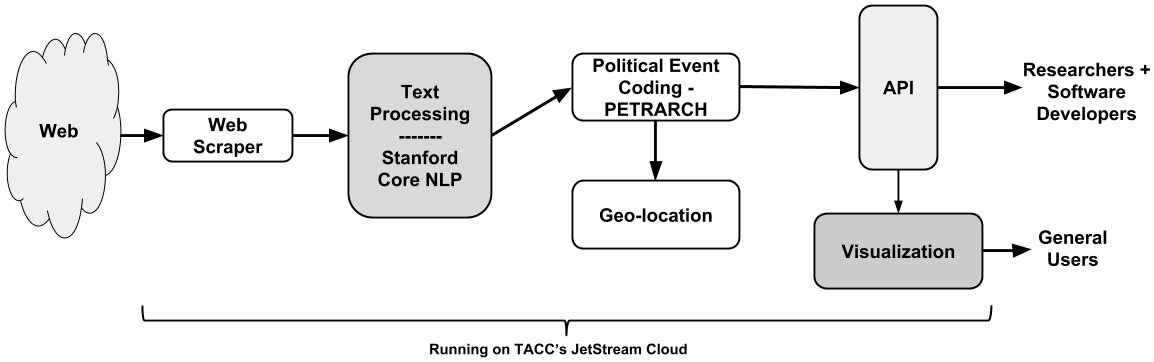



Figure 2.2: Framework Diagram

## 2.4 Event Coding Framework

Figure 2.2 depicts the components of the framework and how the pipeline works. From the web we collection news articles using Web Scraper. Then we pass those documents through the Text Processing module where Stanford's CoreNLP is used to annotate the documents at sentence level. The processed output is provided to Political Event Coding module where PETARCH and Geo-location software work together to generate geo-located events (events along with the place where it happened). The generated events are the distributed using the API to the Researchers and Visualization tools. In the following paragraphs, we discus about the real-time distributed framework for automated event coding in detail. The framework basically has three components.

- Data Collection

- Data Processing and Meta-data Generation

- Event Generation

### 2.4.1 Data Collection

We collect the data using URLs from the RSS Feed of different news agencies. There are $\approx 400$ of them are used. This news agencies are dispersed around the world. After collecting data in regular HTML form, we clean and extract main stories using the tools available here. We collect the data $24 \times 7$ with an interval of 20 minutes. We use a web-scraper[10] program for collecting and preprocessing the data in this step. We use python's newspaper library[4] to clean and extract meta-data from raw HTML documents.

### 2.4.2 Data Processing and Meta-Data Generation

As many other text-based software, event coder requires some structured from of a sentence rather than the bare text. In this step, we process the text to generate those structured form (i.e., POS Tagging, NER, etc) and use Stanford CoreNLP software for the purpose. But CoreNLP itself is slow for being computationally intensive. That's motivated us to apply distributed solution using Apache SPARK. We create multiple nodes running Apache Spark "Worker" program and and distribute the articles in different nodes and process it with CoreNLP software to generate the following annotation of sentences -

- Parts-of-Speech (POS) Tagging
- Tokens
- Named Entity Recognition (NER)
- Parse Tree
- Dependency Diagram
- Sentiment Analysis

### 2.4.3 Event Generation

Using the processed data from previous step, we generate time-stamped political event data using automated event coder. We use PETRARCH software from Open Event Data Alliance for the purpose. It generates CAMEO compatible events. For geo-locating events, we use Cliff-Clavin[33] from MediaMeter. It gives us locations associated with the events found in the sentences.

All the processed text articles and generated events are stored in NoSQL database, MongoDB. All three components use Apache Kafka to synchronize the input-output order. We provide an example of event below to help reader understand about the information generated at the end of the pipeline.


```
{
"_id" :
    ObjectId("59f7ffe8583dca26c72947e0"),
"code" : "010",
"src_actor" : "ITA",
"month" : "10",
"tgt_agent" : "",
"country_code" : "DEU",
"year" : "2017",
"id" : "59f7fea8de7923402d8a5d6d_4",
"source" : "ITAGOVBUS",
"date8" : "20171031",
"src_agent" : "GOV",
"tgt_actor" : "DEU",
"latitude" : 50.11552,
```

```
"src_other_agent" : "BUS",
"quad_class" : 0,
"source_text" : "euroobs",
"root_code" : "01",
"tgt_other_agent" : "",
"day" : "31",
"target" : "DEU",
"goldstein" : 0,
"geoname" : "Frankfurt am Main Hessen",
"longitude" : 8.68417,
"url" : "https://euobserver.com/
    economic/139660?utm_medium=rss",
"date8_val" :
    ISODate("2017-10-31T00:00:00Z")
}
```


The above event corresponds to the following sentence

```
"The recalibration of our asset purchases
reflects growing confidence in the
gradual convergence of inflation rates
towards our inflation aim," ECB
President Mario Draghi said at his monthly
press conference in Frankfurt.
```

For a detail description about each fields in the data, please refer to CAMEO codebook[77]. We developed a rich query based API running as a web-service and handling requests from clients for serving the data. The description of the API is presented in the following section.

## 2.5 API Access to Event Data

We provide API to serve generated event data. This API maintains a api-key based access restrictions and serves the data in JSON format. Interested users can query using a defined query language. In this section we will briefly describe how user can select specific portion of the data using selection and query mechanism. We provide following facilities for users to select required set of data

- Projection
- Selection
- Aggregation

All of the requests should be directed to the server hosted here -
http://eventdata.utdallas.edu/api/data

### Projection

To project on the required fields in the data, we add select clause in the query request. For example, if we want only source, target and date values from the event presented, we can add the following to the query as an argument

```
select=source,target,date8
```

### Selection

To select certain entries from the dataset, we can add query argument to the request. The structure of the query follows JSON like formation, similar to MongoDB. Our query translator on the server side translates to a MongoDB query. AN example where we find events indicates making public statements (root-code for event is "01") about United States on March 28, 2018 can be written as follows-

```
query={"target":{"$in":["USA","USAGOV"]},
"root_code":"01","date8":"20180328"}
```

### Aggregation

Above operations gives back the result at individual record level. We provide the facility to aggregate multiple records and get a summarized result at API level. It will be helpful for the users to define there own data pipeline which involves multiple stages. An example of aggregation based query is given below -

```
aggregate=[{"$match":{"source":{"$not":
{"$in"["MEX","MEXGOV"]}}}},
{"$sort":{"_id":-1}},{"$limit":10}]
```

Here we are looking the latest 10 events which does not include Mexico or Government of Mexico as source. More detailed version of how-to procedures can be found here [7].

## 2.6 Multilingual Event Coding

We are using enhanced version of PETRARCH event coder that works on universal dependency rather than only the Parts-Of-Speech tags used by original version of PETRARCH. It is helpful for multilingual event coding because of uniformity of universal dependency across language vs. differing Parts-of-Speech tags across languages. Using universal dependency based event coder, we can easily incorporate coding in other foreign language with zero effort on updating the event coder itself. We will still need the dictionaries to be translated first. But gives lots of flexibility for the non-cs background researchers as they can solely focus on the analysis and ontology extension parts. An example of universal dependency for the following sentence

```
Ukraine ratified a sweeping agreement
```

```
with the European Union on Tuesday.
```

is given below.

```
1 Ukraine Ukraine PROPN NNP Number=Sing 2 nsubj _ _
2 ratified ratify VERB VBD Mood=Ind|Tense=Past|VerbForm=Fin 0 root _ _
3 a a DET DT Definite=Ind|PronType=Art 5 det _ _
4 sweeping sweeping ADJ JJ Degree=Pos 5 amod _ _
5 agreement agreement NOUN NN Number=Sing 2 dobj _ _
6 with with ADP IN _ 9 case _ _
7 the the DET DT Definite=Def|PronType=Art 9 det _ _
8 European european PROPN NNP Number=Sing 9 compound _ _
9 Union Union PROPN NNP Number=Sing 5 nmod _ _
10 on on ADP IN _ 11 case _ _
11 Tuesday Tuesday PROPN NNP Number=Sing 2 nmod _ SpaceAfter=No
12 . . PUNCT . _ 2 punct _ SpacesAfter=\n
```

For dependency visualization, please refer to the Tree depicted in Figure 2.3

We take the Google translated version of the above sentence in Spanish and universal dependency parser gave us the following output.

```
1 Ucrania Ucrania PROPN PROPN _ 2 nsubj _ _
2 ratificó ratificar VERB VERB Mood=Ind|
Number=Sing|Person=3|Tense=Past|VerbForm=Fin 0 root _ _
3 un uno DET DET Definite=Ind|Gender=Masc|
Number=Sing|PronType=Art 4 det _ _
4 acuerdo acuerdo NOUN NOUN Gender=Masc|Number=Sing 2 obj _ _
5 radical radical ADJ ADJ Number=Sing 4 amod _ _
6 con con ADP ADP AdpType=Prep 8 case _ _
```

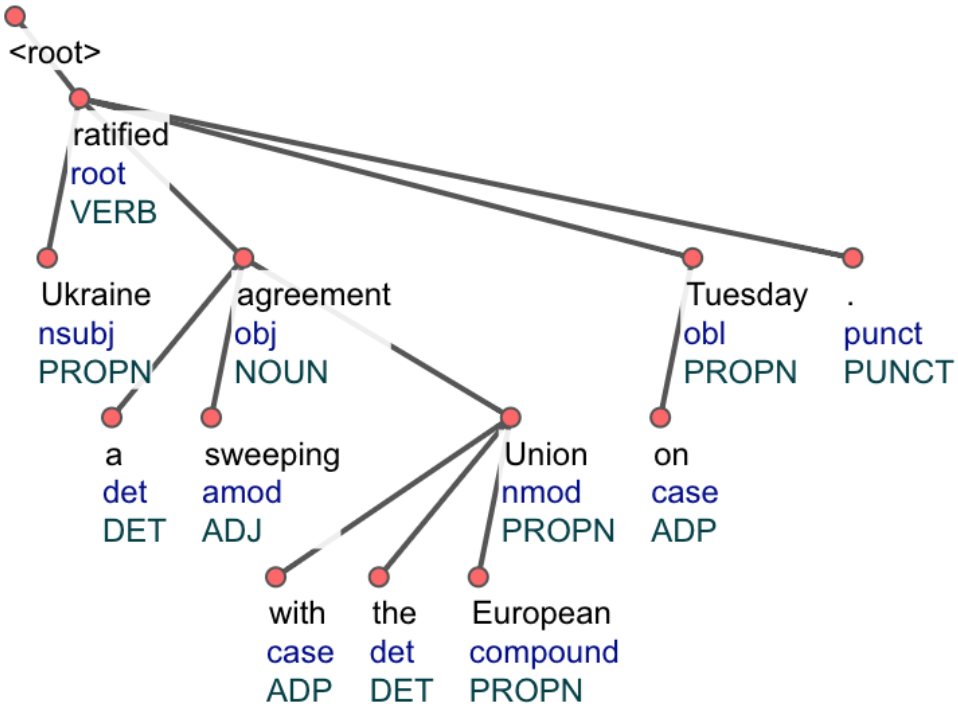



Figure 2.3: Universal dependency tree for English

```
7 la el DET DET Definite=Def|Gender=Fem|
Number=Sing|PronType=Art 8 det _ _
8 Unión Unión PROPN PROPN _ 4 nmod _ _
9 Europea Europea PROPN PROPN _ 8 flat _ _
10 el el DET DET Definite=Def|Gender=Masc|
Number=Sing|PronType=Art 11 det _ _
11 martes martes NOUN NOUN AdvType=Tim 2 obl _ SpaceAfter=No
12 . . PUNCT PUNCT PunctType=Peri 2 punct _ SpacesAfter=\n
```

and the corresponding dependency graph is shown in Figure 2.4. Comparing Figures 2.3 and 2.4, we find that both English and Spanish universal dependency tree shows similar structure. So it will provide same event if both the dictionaries have corresponding patterns. Using the above universal dependency parse, enhanced PETRARCH event coder do the encoding by

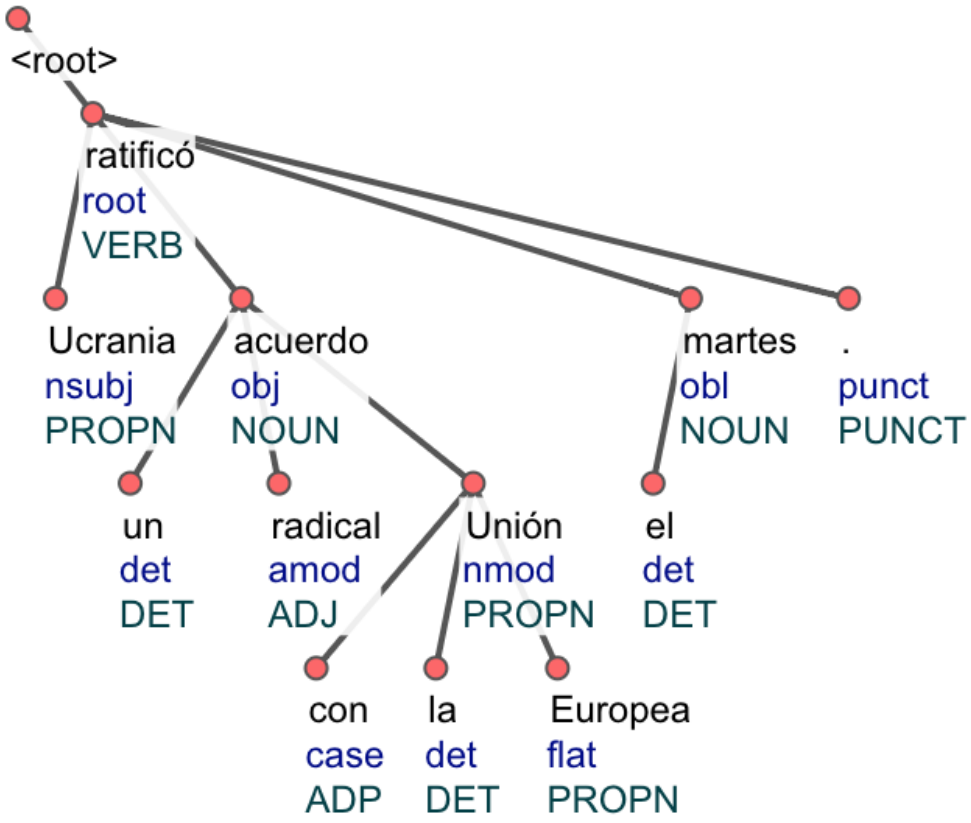



Figure 2.4: Universal dependency tree for Spanish

forming the dependency graph, identifying pattern from PETRARCH2's CAMEO dictionary and actors from the actor dictionaries. Generated events are same as presented before. Details of the process can be found here[59].

### 2.7 System Configuration

The framework components are running in different nodes of JetStream cloud under XSEDE, NSF initiative to facilitate academic research [87]. There we have in total of 7 nodes. Three of them forms a Spark Cluster with 30 cores and 90 GB of ram. We have a 6-core machine to collect data from website using the scraper program. We run a single-instance of Cliff-Clavin server on a 10 core machine along with the PETRARCH event coder. We have a separate

6 core machine for running other utility servers like Apache Kafka, Zookeeper, etc. We also have a dedicated machine for MongoDB Storage Engine. Each node in there is Dell's M630 machines.

## 2.8 Dataset Description

Using the web-based API, we are currently serving the real-time dataset along with other prominent event datasets. Those are listed in the Table-2.1 The ICEWS dataset[20] comprises events related to politics and built a system on the dataset to provide conflict early warnings. The Cline center dataset[14] covers Phoenix event data for the period of 1945 to 2015. They used PETRARCH-2 as the event coder and done event coding on the documents from New York Times (NYT), BBC Monitoring's Summary of World Broadcasts (SWB) and CIA's Foreign Broadcast Information Service (FBIS). Phoenix Real-Time is the dataset being accumulated by our proposed framework.

Table 2.1: List of available datasets

| **Event Dataset** | **Timespan** | **Number of Events** |
|---|---|---|
| ICEWS | 1995 - Sep 2016 | 15,220,347 |
| Cline Phoenix NYT | 1945-2005 | 1,092,211 |
| Cline Phoenix FBIS | 1995-2004 | 8,179,55 |
| Cline Phoenix SWB | 1979-2015 | 2,906,715 |
| **Phoenix Real-Time** | Oct 2017 - ongoing | 844,298+ |

## 2.9 Real-time Dataset Summary

In this section we provide some basic analysis of the dataset generated by the framework to give user some insight about using the data. Results presented here are calculated on the entire dataset collected from October 2017 till August 13, 2018. Figure-2.5 reflects the distribution of events based on root code(i.e 01, 02,...,20). Each indicates a particular category of events, more granular than quad class based event distribution. Here, we found

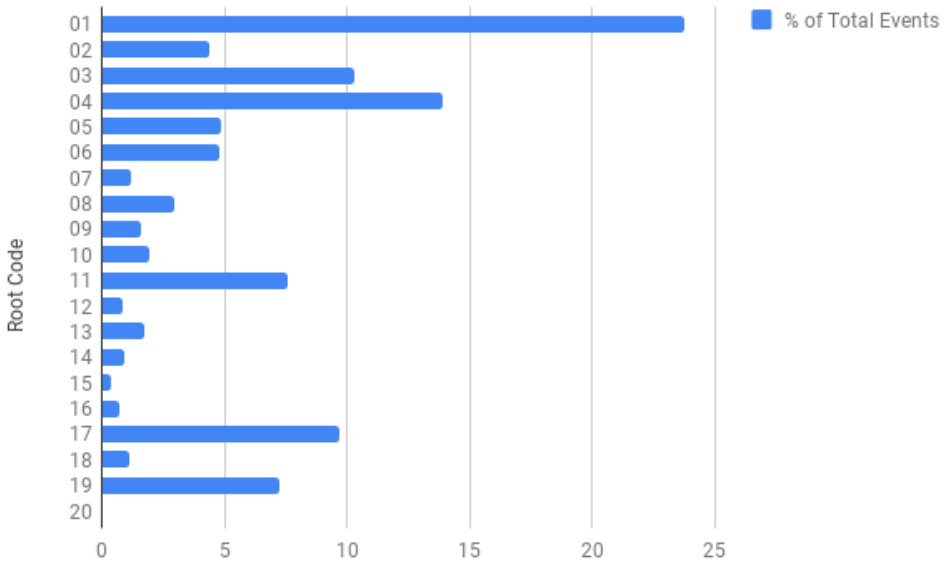



Figure 2.5: Distribution of events based on root code

MAKE PUBLIC STATEMENT (root code is 01) has the highest percentage followed by CONSULT (root code=04). This is because large number of the news articles have someone making statement towards the very beginning of the text. PETRARCH2 focuses on the first paragraph (up-to 6 sentences) for event coding. That's why there is higher percentage of events in that category. We consider the root-code 01 for further granularity and gather percentage of events belonging to each of the 10 event code (i.e 010, 011,...,019.) Again we found highly imbalanced distribution where event code 010 has largest percentage of $\approx 87\%$. It indicates most generic event where event is captured based on any public statements expressed verbally or in action not otherwise specified. In the following example, we try to derive political relation between two countries Saudi Arabia (SAU) and QATAR (QAT). We plot the quad class of the events where source is SAU and target is QAT and Figure-2.7 reflects that. There are 224 events and almost 95% of them indicates non-friendly relationship between those two entities. This also correlates with the real world scenario where we observe different conflict between those two countries including terrorism, airline embargo.

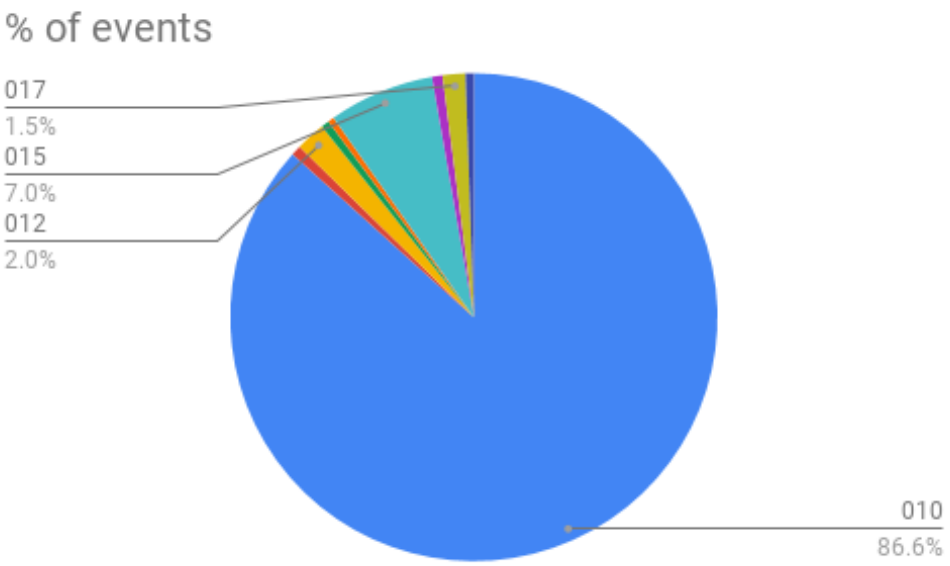



Figure 2.6: Distribution of events with root code 01

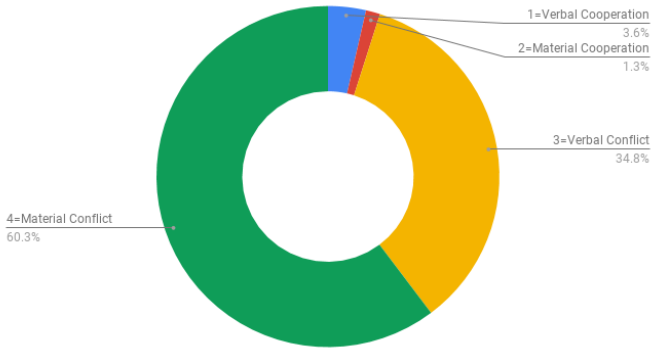



Figure 2.7: Distribution of quad classes where source=SAU and target=QAT

Another scenario between EU and United Kingdom may also be considered. Their relationship is revolving around Brexit. Here we present two graphs showing EU as source and United Kingdom as Target (Figure-2.8) and the vice-versa (Figure-2.9) By observing these figures, EU is balanced in terms of conflict and cooperation but United Kingdom mostly inclined towards compliance.

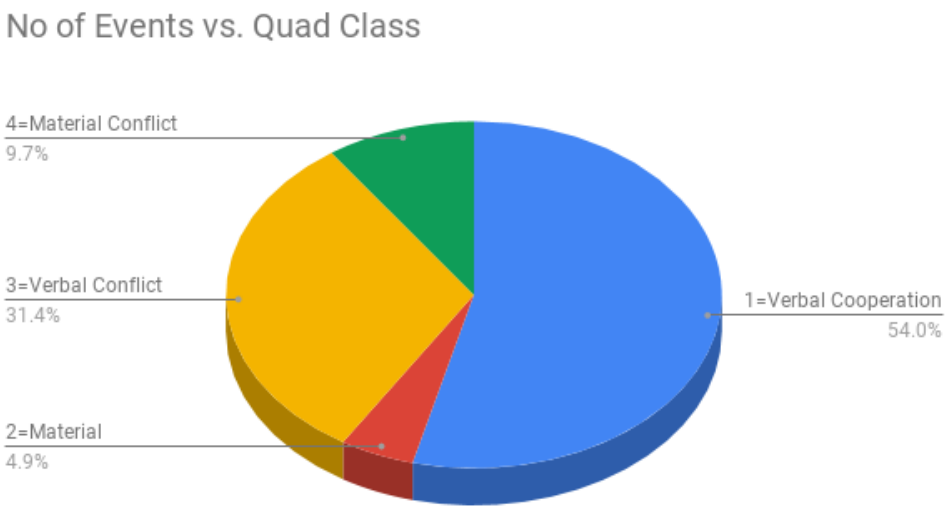



Figure 2.8: Distribution of quad classes where source=IGOEUREEC and target=GBR, 226 events

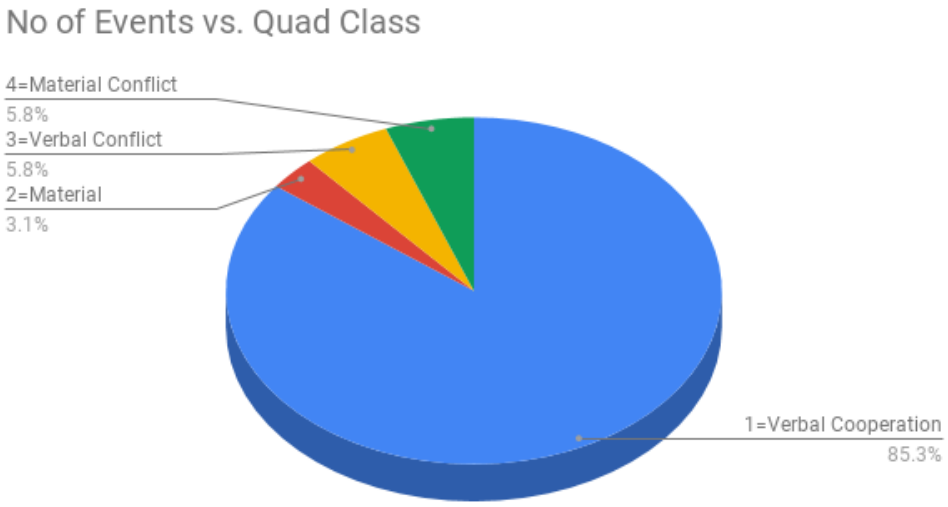



Figure 2.9: Distribution of quad classes where source=GBR and target=IGOEUREEC, 381 events

## 2.10 Related Applications

In this section we will highlight some possible scenario where mining the generated events can provide useful political or social understanding of different phenomena. As presented in the Section-2.9, simplified analysis often reveals insights about the data, supporting information that we already know, and provide answers to the unknown facts. We think, the real-time dataset along with other datasets supported by the API will continue to provide strong support to the research community for better understanding the political interactions. The framework used here can be extended to understand other non-political social phenomena (i.e,. Human Migration).

To study human migration, we need similar ontology driven information extraction mechanism. We can rely on human annotator to provide us CAMEO-like dictionaries useful for capturing migration related events in the news articles. We can semi-automate the process by using similar system as automatic actor detection framework. The key difference is we have to identify patterns involving verbs rather than named entities, which is much harder problem to address and open for research. Once we have the necessary pieces of information to run the system, we can reuse this framework to accomplish the goal.

Once we gather these information, either political or social, we can mine and extrapolate related events. For example, observing two entities and try to predict they are becoming prone to war can be within the scope of the data generated by this framework. Simple analytical study on previous wars and associated entities may reveal threshold values of associated parameters.

But, thankfully now a days, war a less common phenomena than political instability where a nation or organization falls victim to its own citizen or people. Identifying stability at state or organizational level will be an appropriate scenario for the dataset and framework. We will human annotator to help us identify a time-line for the instability to occur (from start

to end). We will do sequence analysis[89] on the subset of the data related to that before it started (to identify the cause) and during start and end of time (to identify the sequence of phenomena that led to the result). Following sequences will be studied w.r.t the learned ones and classified to reflect potential future course of events.

Another key area of news articles mining is news credibility identification (aka Fake News Detection). Machine driven authentication to a statement or news is still being explored as a research field. There are several parts in this validation mechanism like Fact Checking[46], Relevancy analysis at article level[25] etc. The data we collected will be used as search-able knowledge base for fact checking. As event coding works at a sentence level, a statement can be easily converted to structured form for matching against other facts already collected.

# CHAPTER 3

# REPAIR: RECOMMEND POLITICAL ACTORS IN REAL-TIME FROM NEWS WEBSITES

## 3.1 Introduction

Political event data [18] are machine-encoded from news reports. Events are categorized based on a set of dictionaries for the political actions and actors. The typical format to create these data is to determine "who-did/said-what to whom" [5, 39]. While the set of actions or verbs is finite and can be matched to a fixed ontology, the set of source or target nouns is large and always expanding. Identifying these political actors along with their roles is important to transform news reports into useful data for the study of international relations and civil conflict. The final event data has the form of a source-action-target sequence of political interactions.

Currently, actors and roles are added to the dictionary manually. Automated coders (i.e., PETRARCH [5]) use those dictionaries to identify events. However, if a source or target actor is *not* in the relevant actor dictionary, PETRARCH will not code an event related to the new actor. This motivates us to design a framework for recommending new political actors in real-time so that the dictionaries stay up to date with new actors. Furthermore, existing event coders (e.g., PETRARCH) use only the syntactic structure (i.e., parse tree) of a sentence which fails to capture the semantic content of a sentence. More precisely, they struggle to encode event from a complex sentence. As a result, either they generate events with wrong actors (source/target) or fail to generate any event at all (more details in section 3.3.2). This drives us to use the sentence semantic for our actor recommendation framework.

Automatic Content Extraction (ACE) programs [30] infer entities, relations, and events from natural language data. An event consists of relations which are a number of participants (ACE entities) and each participant has a semantic role that it plays in the event (agent,

object, source, target). Machine learning-based semantic role labeling techniques [41] consist of the detection of the semantic arguments associated with the predicate or verb of a sentence and their classification into their specific roles. The PropBank[16] Corpus provides structured predicate-argument annotation for the entire Penn Treebank [91]. Each verb in the Treebank is annotated by a single instance in PropBank and acts like an event containing information about the source, target, and location of the verb.

Implementing an automated system for recommending new political actors poses some key challenges. First, an actor may come with multiple aliases, e.g., 'Barack Hussein Obama', 'Barack Obama', 'President Obama', etc. Currently, a human expert puts a dictionary entry for each of these aliases in Conflict Analysis and Mediation Event Ontology (CAMEO) actor dictionary [39]. Second, the role of an actor changes over time. For example, 'Shimon Peres' served multiple political roles in Israel during his lifetime. Finally, processing a large volume of news articles across the world in real-time demands a fast, scalable, and distributed computing solution.

Building on earlier work [85] to address these challenges, we develop a real-time, distributed framework, *RePAIR*, to recommend new political actors and their associated roles. The chapter highlights the following contributions:

First, we propose a novel time window-based, unsupervised technique for new actor and role recommendations. We designed a frequency-based actor ranking algorithm with alias actor grouping from news articles that also integrates an external knowledge-base (Wikipedia) to capture the timeline of an existing actor's role change and to suggest possible new roles. Second, we have developed an ACE based semantic event coding to augment the traditional syntactic event coding for our actor recommendation framework. Third, we develop a distributed framework using Apache Spark Streaming [101] to address the scalability of coding political events from news articles in real-time. Finally, we compare our proposed ap-

proaches with state of the art (e.g., PETRARCH, BBN ACCENT event coding) and shown the effectiveness of our work.

The rest of the chapter is organized as follows: Section 3.3 explains the overall framework. Section 3.4 describes our new actor with role recommendation technique in details. Section 3.5 shows the experimental results for new actor detection. Section 3.6 covers the related works.

## 3.2 Background

**CAMEO** (Conflict and Mediation Event Observations) is an event coding framework to record political events (both material and verbal interactions) in a structured who-did-what-to-whom format. It was originally intended to the study of interstate conflict mediation [79]. Over time, it has developed as a "next generation" coding scheme for automated coding and the detailed coding of sub-state actors. There are dictionaries for actors and actions (verbs) which form the knowledge-base for the framework. Tools using this framework first consider the structure of the sentence to identify possible actors and actions, then they look for their existence in the corresponding dictionaries. Once found, they can use an appropriate actor and action coding to represent the event. Each action in CAMEO is represented with a designated numeric code. Each actor is represented with corresponding actor codes (i.e., GOV, MIL, USA, etc.). These are associated with the actors based on a timeline since the role of an individual can change when they achieve a new position in government or other agencies.

**PropBank** [71] is a corpus which annotates text semantically. It adds a semantic layer to Penn Treebank [91] which captures the accurate predicate-argument structure by annotating predicates and the semantic roles of their arguments.

### 3.3 Framework

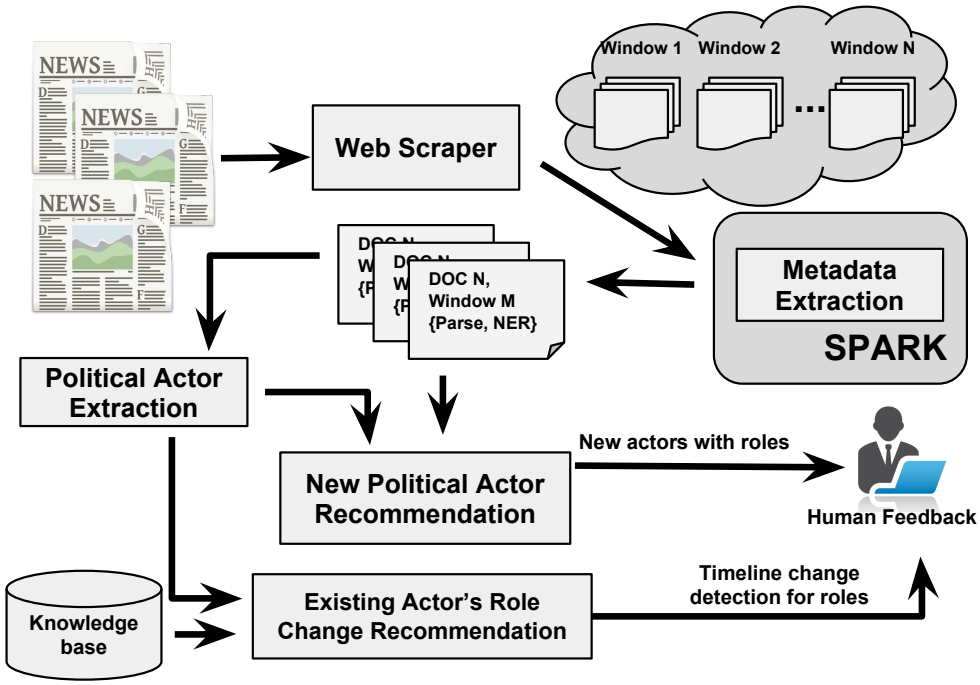



Figure 3.1: Framework for real-time new political actor recommendation

Figure 3.1 illustrates our new political actor recommendation framework. *RePAIR* collects political news stories periodically through web scrapers. Later, it uses CoreNLP [60] and PropBank to extract NLP metadata from the stories such as parse trees, tokens, lemmas, NER, semantic events (by PropBank), etc. and stores them in MongoDB. We use Apache Spark streaming [101] with Kafka [53] to collect all the scraped data, periodically. We use CoreNLP and PropBank annotation inside a Spark worker node to scale up the process. Next it classifies political events only using CAMEO action ontology and dictionary from the metadata. It fetches possible new actors from the source and target of the events and NER. We implement a time window-based, unsupervised actor frequency ranking technique to recommend potential new actors and their related roles. Finally, a human expert validates the recommended actors and roles to update the dictionary.

### 3.3.1 Web Scraper

We use a web scraper [69] to collect news articles and extract the content of the news. It collects news stories from about 400 RSS (Rich Site Summary) Feeds [8] every 2 hours. The news articles are shipped through Apache Kafka to an Apache Spark-based data processing module[84].

### 3.3.2 Metadata Extraction

Inside our data processing unit, Stanford CoreNLP parses the text of news documents, extracting metadata such as Parts-Of-Speech (POS) tagging, the Parse Trees, and importantly Named Entity Recognition (NER). The results are stored in MongoDB.

**PropBank-based event coding** PropBank generates events [15] from the 'predicate-argument' structures for each of the sentences of a document. It returns the arguments corresponding to the semantic roles of an agent/source (ARG0) and a target (ARG1, ARG2, etc.). It has extra annotations of modifiers that describe time (AM-TMP) and location (AM-LOC) references in a sentence.

Table 3.1 shows the generated events for the following example.

*"Obama vowed again on Sunday to help France hunt down the perpetrators of the attacks."* - The Washington Post 11/15/2015

Table 3.1: PropBank annotation example

| Verb | ARG0 | ARG1 | AM-TMP |
|---|---|---|---|
| vowed | Obama | to help France hunt down the perpetrators of the attacks | again, on Sunday |
| help | Obama | France hunt down the perpetrators of the attacks | |
| hunt | France | down the perpetrators of the attacks | |

The advantage of PropBank over PETRARCH for event encoding is in the processing of compound sentences. For example, consider the following sentence:
*An AFP reporter said Bashar-Al-Assad agreed to take help from President Trump and his Government.*
Using PETRARCH, "AFP reporter" and "Bashar-Al-Assad" are the source and target, and the action is "said". However, the political event is between Bashar-Al-Assad and President Trump. PropBank simplifies the sentence and captures the interaction between those two entities.

### 3.3.3 Political Actor Extraction and Recommendation

For a given sentence, PropBank provides arguments for the source and target of each event. Each event contains a verb/action which is looked up in CAMEO to categorize it as a political event. We parse these arguments and filter named entities using NER, capturing those that are missing in CAMEO actor dictionary as potential new actors. We also detect possible actor aliases and group them together. Based on this list of new actors and aliases, we introduce a time window-based actor ranking technique that selects the top $N$ actors for each window. Roles are suggested using the roles of co-occurring actors. This is described in detail in section 3.4.

### 3.3.4 Existing Actor's Role Change Recommendation and Verification

We integrate external knowledge bases (e.g., Wikipedia, Google Knowledge Graph [2]) to capture role changes for existing actors. If a role change is detected, then our framework captures the time-line changes and recommends the new role (detail ed in section 3.4.5). Finally, we provide a graphical user interface and a dashboard to the end user/human expert to provide recommendations about the new political actors with their role to update the actor dictionary for CAMEO coding via PETRARCH.

## 3.4 Recommending A New Actor and Role

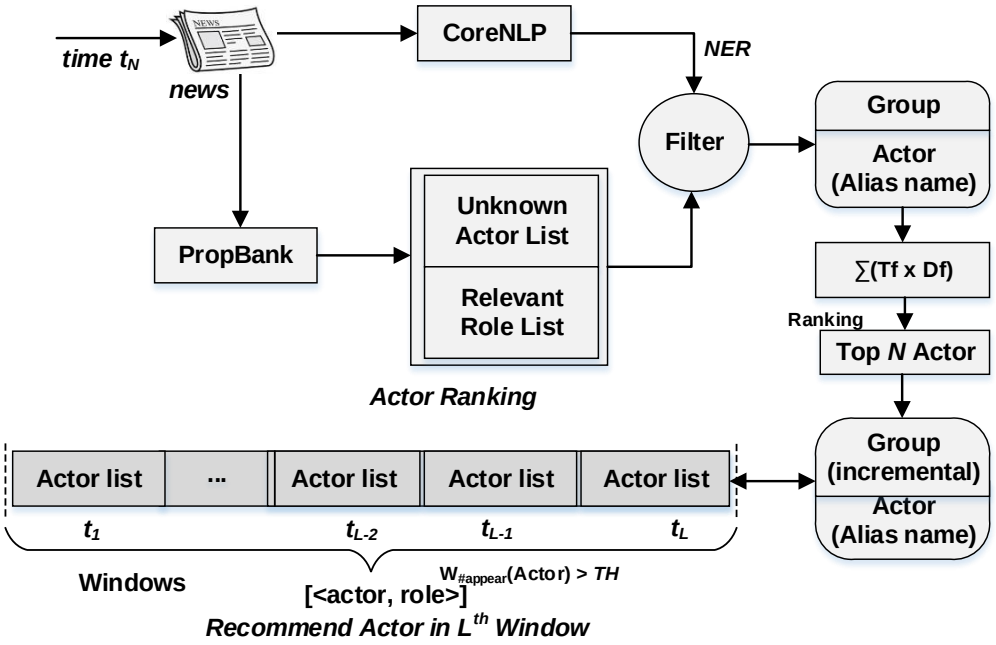



Figure 3.2: Actor recommendation procedure in *RePAIR*

Figure 3.2 depicts the details of our real-time actor recommendation for the dictionary updates.

### 3.4.1 New Actor Discovery

We use a frequency-based ranking algorithm to recommend the top political actors and their roles. Stanford CoreNLP parses the raw text of each news report and outputs NER for each sentence. Next, PropBank generates event codes with arguments for each sentence using its practnlp tool [6]. From these PropBank arguments, we filter a named entity list from the NER output. This list contains both existing actors and probable new actors. The role of an existing actor can be looked up from CAMEO dictionary. For example, *President Obama* has role $USAGOV$. We filter out potential new actors by removing existing actors from the proposed NER actor list. Sometimes, an actor may come in different alias in the same

article. For example, *Barack Hussein Obama* may come as *Obama* or *Barack Obama* in many sentences. As a result, we need to group these actors with all possible alias names and make a single actor name (e.g., *Barack Hussein Obama*). We use two similarity measures Levenshtein Edit Distance [54] and MinHash [22] independently to group actors' aliases. These similarity measures need a similarity threshold $sim_{th}$ which is estimated from existing CAMEO actor dictionary (detailed in section 3.5.2). After grouping, we calculate the term-frequency of each actor in a news article. For an actor alias, we compute the maximum count of the common phrases for an actor's name as its term-frequency for grouped actor aliases. We assume that a common phrase from all aliases will be found in a document. For example, the term-frequency of *Barack Hussein Obama* will be the maximum frequency of *Barack Hussein Obama, Obama, and Barack Obama* as Obama is the common word among the names. Similarly, we group aliases for actors across documents in a given time window. Finally, for each actor $a$, we calculate a document frequency $df$ and use the following equations to generate a score:

$$rank(a) = \sum_{d \in D} tf(a, d) \times df(a, D) \tag{3.1}$$

We use term frequency $tf(a, d) = count_{a \in d}(a)$ to show how frequent an actor $a$ is in document $d$. Equation 3.1 shows the rank calculation of an actor $a$, where $df(a, D) = \frac{|d \in D : a \in d|}{|D|}$ is document frequency and it shows how frequent an actor $a$ comes across all news articles/document in the set $D$.

### 3.4.2 Actor Role Discovery

We collect all events from the news reports that contain new and existing political actors. The CAMEO actor dictionary contains existing actors and their roles, which historically have been coded by humans. We assume that new actors' political roles will be related to

Table 3.2: Symbol for algorithm

| | | | |
|---|---|---|---|
| $D$ | document set | $d$ | document |
| $m$ | meta-data | $sim_{th}$ | similarity threshold |
| $E$ | event code set | $e$ | event code |
| $A$ | actor set | $a$ | actor |
| $R$ | role set | $r$ | role |
| $tf$ | term freuency | $df$ | document frequency |
| $N$ | top N actor | $L$ | number of window |
| $k$ | merged actor as key | $TH$ | Threshold in window |

their most frequent co-occurring existing political actor roles. So we collect the roles from the existing CAMEO dictionary and map them to new actors' roles. We update actors' maps with a role across news articles. If an actor appears in two news articles then we include all possible roles across the articles increasing common roles.

### 3.4.3 Recommending New Actors in Real-time

We use algorithm 2 to describe the user recommendation procedure for dictionary updates. For time window $W$ of length $L$ we receive the actor list with ranking $M_{rank}$. Then, take the top $N$ actors from the list (lines 5-7) and update the new actor map $M_A$ and its role map $M_R$. These are incrementally updated during each time window. For each actor in $M_{rank}$, find the list of co-occurring actors in $M_A$. If there are no actors in $M_A$, insert it in $M_A$ with occurrence 1 and the role in $M_R$. If the closest match is present, then merge the actor name as required. Later, increment the occurrence in $M_A$. In the same way, update the role of that actor in $M_R$ (lines 8-20). When the number of actors increases in $M_A$ and $M_R$, their size also increases, so we introduce a $(min, max)$ threshold $TH_{min}$ *and* $TH_{min}$. After the $L^{th}$ time window, if the total occurrence of an actor is greater that $TH_{max}$, we consider this as a new actor and recommend the top $N$ roles from $M_R$. On the other hand, if the total occurrence is below $TH_{min}$, discard the actor from $M_A$ and $M_R$ (lines 21-31). After the $L^{th}$

**Algorithm 1** New actor discovery

**procedure** ACTORDISCOVERY($DocumentSet\ D$)
  $Actor\ Map\ M \leftarrow \{\}$
  **for each** $d \in D$ **do**
    $m \leftarrow CoreNLP(d)$
    $E_d, A_{current} \leftarrow PropBank(d)$
    $A_{all} \leftarrow m.NER()$
    $A_d \leftarrow A_{all} - A_{current}$
    $R_d \leftarrow E.Roles()$
    **for each** $a \in A_d$ **do**
      $tf \leftarrow count_{a \in d}(a)$ ▷ calculate tf
      **for each** $a_i \in A_d - a$ **do**
        **if** $Match(a, a_i) > sim_{th}$ **then**
          $a \leftarrow Merge(A_d, a, a_i)$
          $tf \leftarrow max(tf, count_{a_i \in d}(a_i))$
        **end if**
      **end for**
      $k \leftarrow \arg\max_k \{Match(M(k), a) > sim_{th}\}$
      **if** $k \neq empty$ **then**
        $k \leftarrow Merge(M, k, a_i)$
        $M.insert(k, [(tf, d, R_d)])$
      **else**
        $M.insert(a, [(tf, d, R_d)])$
      **end if**
    **end for**
  **end for**
  $M_{rank} \leftarrow \{\}$
  **for each** $k \in M.keys()$ **do**
    $df \leftarrow |k \in d : d \in D|/|D|$ ▷ calculate df
    $rank \leftarrow \sum_{tf \in M(k)}(tf) \times df$
    $M_{role} \leftarrow \{\}$
    **for each** $R_d \in M(k)$ **do**
      **for each** $r \in R_d$ **do**
        $M_{role}(r) \leftarrow M_{role}(r) + 1$
      **end for**
    **end for**
    $M_{rank}.insert(k, (rank, M_{role})$
  **end for**
**end procedure**

**Algorithm 2** Real-time new actor recommendation

**procedure** ACTORINREALTIME($TH_{min}, TH_{max}, N, L$)
  $New\ Actor\ Map\ M_A \leftarrow \{\}$
  $New\ Actor\ Role\ Map\ M_R \leftarrow \{\}$
  **for** $t \leftarrow t_1\ to\ t_L$ **do**
    $D \leftarrow getDocuments(t)$
    $M_{rank} \leftarrow ACTORDISCOVERY(D)$
    $M_{topN} \leftarrow M_{rank}.top(N)$
    **for each** $< a, (rank, M_{role}) > \in M_{topN}$ **do**
      $k \leftarrow \arg\max_k \{Match(M_A(k), a) > sim_{th}\}$
      **if** $k \neq empty$ **then**
        $k \leftarrow Merge(M, k, a)$
        $M_A(k) \leftarrow M_A(k) + 1$
      **else**
        $M_A(a) \leftarrow 1$
      **end if**
      **for each** $r \in M_{role}$ **do**
        $M_R(k) \leftarrow M_R(k) + 1$
      **end for**
    **end for**
  **end for**
  **for each** $a \in M_A$ **do**
    **if** $M_A(a) \geq TH_{max}$ **then**
      $M_A(a) \leftarrow$ "$new\ actor$"
      $M_R(a) \leftarrow$ "$new\ actor's\ role$"
    **end if**
    **if** $M_A(a) \leq TH_{min}$ **then**
      $M_A(a) \leftarrow$ "$discard\ actor$"
      $M_A.remove(a)$
      $M_R.remove(a)$
    **end if**
  **end for**
**end procedure**

time window, we recommend new actors with possible related roles. Human experts will verify this and update CAMEO dictionary accordingly.

### 3.4.4 Graph-based Role Detection Technique

We also consider a graph-based technique to suggest roles for the recommended actors. This is an alternative to our frequency-based role detection technique. The role of a new actor will be influenced by existing actors with whom he/she has mostly interacted. Therefore, we use weighted label propagation technique [58] to infer possible roles from existing related political actors.

We formulate a graph $G = (V, E)$ that implies the interaction between actors. Here, $V$ is the set of actors (contains both existing and recommended actors). For two actors, $u$ and $v$, $(u, v) \in E$ represents those actors are mentioned in the same news article. We also assign a weight function $w(u, v)$ as follows

$$w(u, v) = \text{co-occurrence count of } u \text{ and } v$$

After that, we assign labels to the nodes which are the roles found in the CAMEO actor dictionary for existing actors. For recommended actors, we simply put an empty label. We use $label(x)$ to denote the label for actor x.

$$label(x) = \begin{cases} \text{roles from dictionary} & \text{if } x \text{ is an existing actor} \\ empty & \text{otherwise} \end{cases}$$

Our iterative label propagation algorithm assigns weights to each possible role for a recommended actor using neighboring actors. Weights are assigned for each role based an associated edge of the interaction graph, $G$. We repeat this process $N$ times or until the

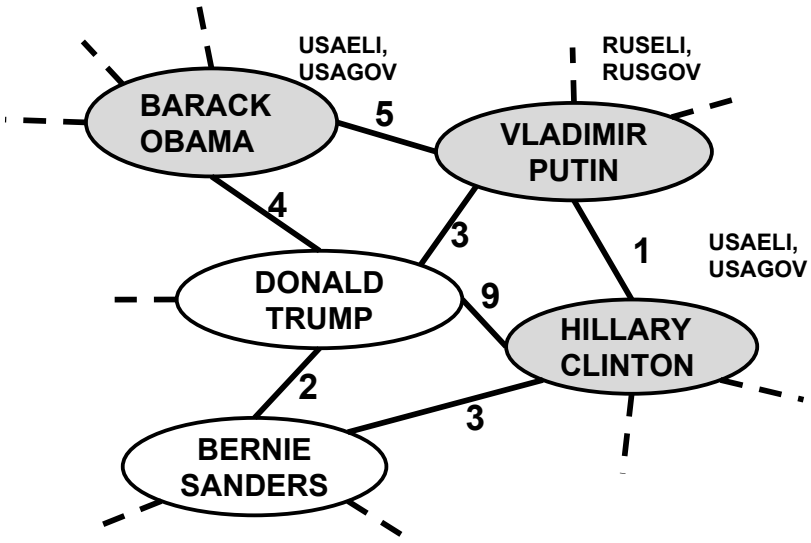



Figure 3.3: Example Scenario for Graph Based Role Recommendation

label assignments are stable. The weighting function for roles works as follows,

$$role - weight(u, role) = \sum_{v \in Neighbors(u)} T(v, role) \times w(u, v)$$

where $T(v, r)$ is used as indicator variable that simply returns 1 if $v$ has 'r' in it's list of roles, 0 otherwise. In Figure 3.4.4, we determine the role for DONALD TRUMP. He is associated with both existing actors (gray nodes) and new political actors. All the existing actors have their roles associated with their nodes. After calculating weights for all roles from neighbors, we find USAGOV, USAELI, RUSGOV as the top 3 possible roles for DONALD TRUMP. The outcome here is that we detect that he is a government leader who has run for president. A limitation is that we have not connected him to anyone who is a prime minister in a parliamentary democracy.

### 3.4.5 Integrating External Knowledge Bases

External knowledge bases (e.g., Wikipedia, Google Knowledge Graph) store up-to-date news, and are helpful for recommendation systems for actor dictionary updates and validation. Al-

though they are significant big data resources they are not alternatives to CAMEO dictionary which contains structured, human-encoded data (e.g., action/actor dictionaries about political events) since these knowledge bases contain more generic, unstructured data. However they suffer from the same problem as the CAMEO dictionaries if a new political actor appears. To integrate these external knowledge bases and identify any time-line changes in the role of an existing or new actor, we query the Google Knowledge Graph [2] for appropriate entities. This disambiguates among closely related entities across popular words. As an example, searching for *Mullah Mohammad Rabbani* in the Wikipedia API, produces no related entries, but the Google Knowledge Graph resolves him to *Mohammad Rabbani* locating him in the Wikipedia database. Experimentally over a sample of 1000 actors from 18000 actors finds only 20 of them are resolved only with Wikipedia, but 850 are correctly identified when using the Google Knowledge Graph. Identifying whether there is a change in the timespan for the actor's role by observing the dates associated with each of his/her responsibilities from Wikipedia pages is then listed for human confirmation. For example, former President of United States, Barack Obama has a dictionary entry where his role is USAGOV and started on 20 January 2009 but has no end-date. From Wikipedia, we find both start date and end date of his presidency. We match start date with the existing entry of the dictionary and add the end date to the list of actor dictionary updates. We also suggest a new role for an existing actor if he/she is assigned to a new political position based on the information from Wikipedia.

## 3.5 Experiments

### 3.5.1 Setup and Dataset

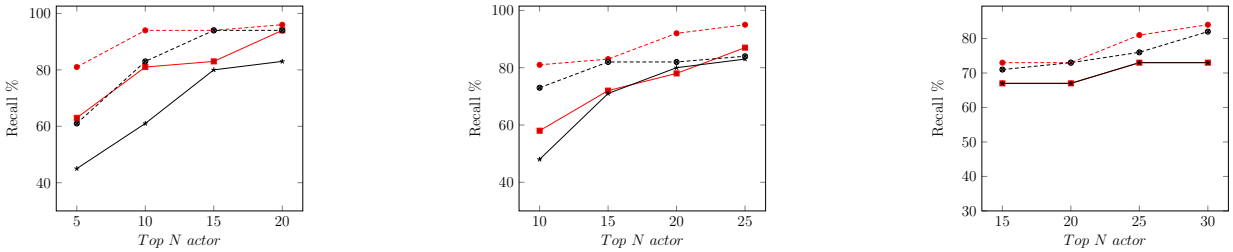
To evaluate our framework, we take 10 time windows at 24 hours interval, containing 131,932 news articles for our experiments. This dataset contains news stories from July 11, 2017 to July 20, 2017.

### 3.5.2 Threshold Estimation

We use partial string matching to measure the similarity for grouping actors and their aliases via MinHash and Levenshein edit distance. These text matching metrics give a value between 0 and 1 indicating the textual similarity, with 1 being an exact match. One needs to set a threshold value for whether two strings represent the same actor using these metrics. To estimate the threshold, we consider the similarity values for actors in the existing CAMEO actor dictionary. For example, consider the following snippet from the actor dictionary for former U.N. Secretaries General Kofi Annan and Boutros Boutros Ghali:

```
....
KOFI_ANNAN_
+SECRETARY-GENERAL_KOFI_ANNAN
.....
BOUTROS_BOUTROS_GHALI_
+BOUTROS_BOUTROS-GHALI_
....
```

Both are followed by multiple aliases in the dictionary (lines with a '+'). We estimate a threshold value that minimizes the number of false positives (i.e., reporting high similarity

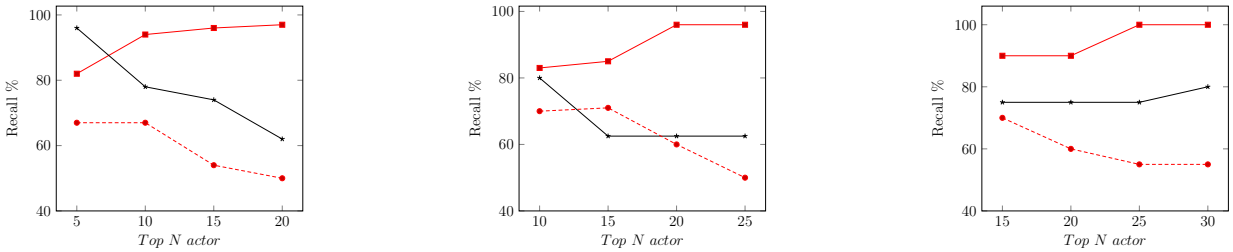



(a) Deleted actor = 5 (b) Deleted actor = 10 (c) Deleted actor = 15

Figure 3.4: Performance for Actor recommendation. Recall: Edit distance (PropBank) -●-, MinHash (PropBank) -⊗-, Edit distance (PETRARCH) —■—, MinHash (PETRARCH) —⋆— when they are not same actor). The estimated thresholds for the Levenshein and Min-Hash methods are 0.75 and 0.4 respectively.

### 3.5.3 Baseline Methods

To compare actor detection methods we use the event coders from BBN Accent [21], PETRARCH, and the ACE methods from PropBank. For role recommendation, we compare our frequency-based and graph-based approaches.




(a) Deleted actor = 5 (b) Deleted actor = 10 (c) Deleted actor = 15

Figure 3.5: Performance for role recommendation. Recall: Edit distance —■—, MinHash —⋆—, Exact match -●-

### 3.5.4 Experiment 1: Performance Evaluation

This first experiment evaluates the framework's correctness across event coding methods. We first select 30 existing political actors from the CAMEO actor dictionary and randomly

remove some of them. We start with removing 5, 10 and 15 actors. The deleted actors are considered for recommendation (as if they are newly discovered). We calculate how many of them are recommended by our algorithm to obtain a recall. We repeat the experiments 10 times to obtain averages for 3 different numbers of deleted actors and plot the result in Figure 3.4. We cannot compute precision because all are possible recommended actors. The retrieved actors are 15-20% of the recommended actors.

In Figure 3.4, we see that if we increase the top $N$ actors recommended at each time window, it increases the possible retrieval of a deleted actor. We also see that our framework which has the ACE PropBank event coder (dotted line) has higher recall than PETRARCH (solid line) for recommending new actors. This is expected because as pointed out earlier PETRARCH can make incorrect decisions in compound sentences. As a result, our framework will generate more relevant actors than PETRARCH. Moreover, the edit distance actor alias grouping (red) has higher recall than MinHash actor alias grouping (black) because Edit distance considers each character where MinHash consider each word when comparing aliases for an actor.

In the next experiment, we suggest the possible roles for identified actors in the previous experiment, using a similar sampling process as first experiment. As a reference, we have the actor roles (R1) listed in the CAMEO dictionary. Now using our frequency-based role recommendation algorithm we predict appropriate roles (R2) for them. We calculate the intersection of the two sets, R1 and R2. If the output is non-empty set then we consider it a success. In this experiment, we keep the size of R2 at 5. The roles are selected based on frequency. The top 5 frequent roles are presented as possible actor roles. Again we vary the number of deleted actors and the number of top-$N$ recommended. We reported the ratio of a number of successes for the number of actors retrieved. Figure 3.5 shows the results. We see that edit distance has a slight advantage for actor recommendation. We use word set in the MinHash calculation and it uses Jaccard similarity. So, for close matched actor

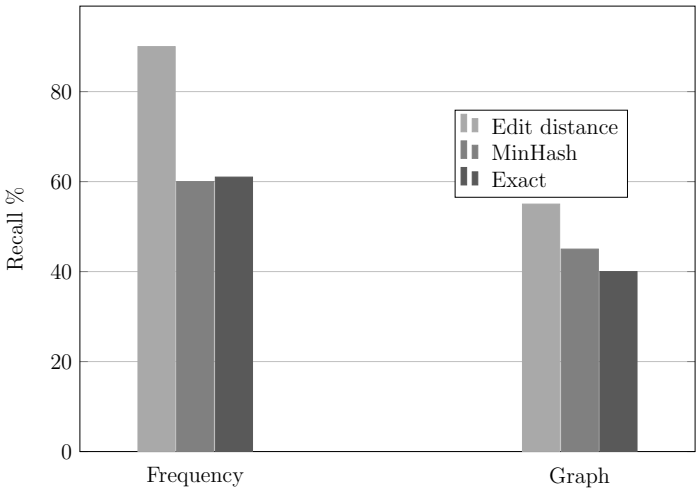



Figure 3.6: Comparison of actor role recommendation with baseline: (N = 15, deleted actors = 15)

aliases like 'Donald Trump' and 'Donald J. Trump', the edit distance metric gives higher similarities than MinHash. For a similar reason, we see recall variation in Figure 3.5 for role recommendation. Here edit distance performs better when the top $N$ increases, whereas MinHash and exact matching have similar levels of performance (worse than edit distance).

**Comparison of role recommendation techniques:** Extending this experiment, we delete some well-known actors from the existing CAMEO dictionary and then try to retrieve their roles by frequency-based and graph-based role detection techniques. We fix the numbers of recommended actors per window and number of deleted actors to 15. We do not vary these parameters for the graph-based approach because the resulting graph is then very sparse. This makes the role inference hard with the label propagation technique. We consider all the actors suggested by the actor recommendation algorithm and their interactions in the graph. Figure 3.6 shows the results. Here, the frequency-based approach outperforms the graph-based approach for all the similarity measurements because it infers new actors' roles from existing actors' roles. The graph-based approach considers roles from neighbors who can be either existing or new actors. In that case, an error in one role assignment can propagate to others. The label propagation algorithm has fewer labels to propagate and the edges

between new actors will be higher because they have a higher possibility of co-occurring in a document.

### 3.5.5 Experiment 2: Recommendation of New Actors with Roles

Next we list the possible newly recommended political actors based on the estimated threshold. The threshold is how many times an actor appeared in the time windows. We set the number of time windows $L = 10$ and (max, min) frequency thresholds of $(TH_{max}, TH_{min})$ = (5, 2) in algorithm 2. So if any actor appears in five (more than 50% of the window length), he/she will be suggested as a potential new political actor. In the case of roles, we list the most probable roles from their co-occurred existing political actors. Table 3.3 shows the list of recommended actors across all windows. We use both MinHash and edit distance based string similarity and list the actors and roles output side-by-side. We find that both the approach detects similar roles for identical actors. But the recommended user list varies across the methods.

When recommending new actors, comparison among the PropBank, PETRARCH, and BBN ACCENT event coders shows precision of the three methods after human validation. Here we report what percentage of recommended actors actually represents a political entity. Following the pattern of previous experiments, PropBank performs best in this scenario.

### 3.5.6 Experiment 3: Scalability Test

We ran our experiment on a Spark cluster which has 1 master node and 10 slave nodes. Each node has 8 vCPU, 16 GB memory and 1 TB space.

We are using CoreNLP which is computationally expensive [84]. Figure 3.8 shows the scalability of average document processing time of 131,932 news articles using Spark with varying number of worker nodes. So, we take average processing time per article. It scales up almost linearly.

Table 3.3: List of recommended actors with their roles

| **Edit distance** | | **MinHash** | |
|---|---|---|---|
| **Actor** | **Top 3 roles** | **Actor** | **Top 3 roles** |
| DONALD TRUMP | USA, USAGOV, GOV | DONALD TRUMP | USA, USAGOV, GOV |
| EMMANUEL MACRON | USAGOV, FRA, FRAGOV | EMMANUEL MACRON | USAGOV, FRA, FRAGOV |
| MITCH MCCONNELL | GOV, USA, USAGOVLEG | REX TILLERSON | USAGOV, USA, QAT |
| REX TILLERSON | USAGOV, USA, QAT | MITCH MCCONNELL | GOV, USA, USAGOVLEG |
| LIU XIAOBO | MED, CHNOPP, CHN | RODRIGO DUTERTE | PHLGOV, LEG, GOV |
| RODRIGO DUTERTE | PHLGOV, LEG, GOV | YEMI OSINBAJO | NGAGOV, NGA, GOV |

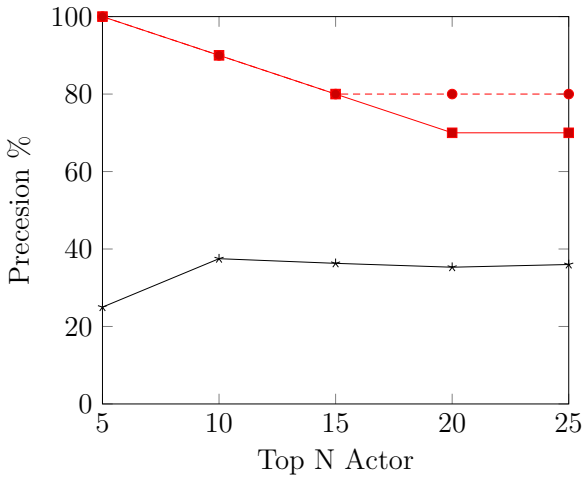



Figure 3.7: Baseline coding comparison in actor detection: PETRARCH —■—, BBN ACCENT —⋆—, and PropBank - -●- -

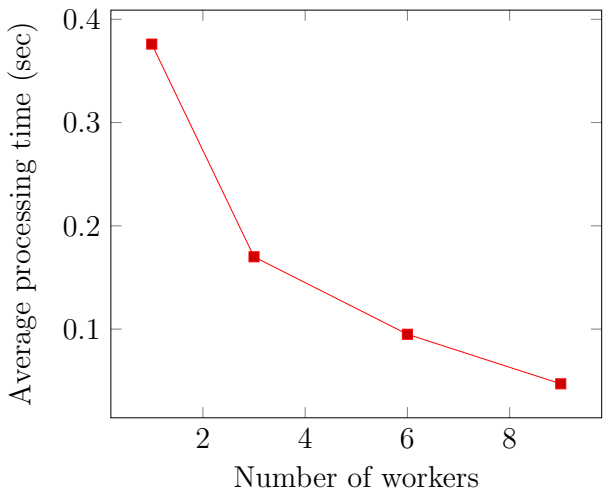



Figure 3.8: Average processing time for of 131,932 documents

### 3.6 Related work

Political event data analysis has been developed over many decades in international relations and security studies. Saraf, et al. [76] develp a recommendation model to determine if an article reports a civil unrest event. Schrodt and Van Brackle [80] show a complete picture of generating events from raw news texts. While each of these works focuses on political event data generation and analysis, none incorporate dynamic actor dictionary building. Here, we design a real-time scalable framework that detects new actors dynamically.

Role recommendation based on surrounding keywords in the document can also be considered. For example, 'President' as keyword occurring in front of Barack Obama may help infer his role. But we may not get enough keywords like this to infer roles. Often they do not convey the complete role for an actor. With the keyword 'President', we can imply Barack Obama as a government employee, but we cannot infer his country. Moreover, CAMEO uses short-form encoded roles (e.g., USAGOV) which require a static mapping in the above procedure. Automatic Content Extraction (ACE) based event extraction systems use patterns or machine learning for labeling. Ritter et al. [73] proposed TWICAL - an open domain event extraction and categorization system for Twitter. Georgescu et al. [38] show event extraction using Wikipedia. Weninger et al. [97] show future Web content extraction algorithms. All the above methods use classifiers to identify event types which performs well in generic domains but not in a restricted a political domain. Therefore, we use a mixed model of CAMEO and PropBank to extract political events.

Finally, Extracting events from raw news articles in real-time demand a scalable distributed streaming framework. We choose Spark Streaming because it is matured and widely used in industry.

# CHAPTER 4

# AUTOMATIC EVENT CODING FRAMEWORK FOR SPANISH POLITICAL NEWS ARTICLES

## 4.1 Introduction

Spanish is the second most spoken language in the world with over 460 million native speakers. Out of a total of 195 countries, there are twenty Spanish-speaking countries, such as Spain, Venezuela, Colombia, Mexico, etc. Spanish-speaking countries have become a hotbed for political conflict like the crisis of leadership in Venezuela or Colombia's war on drugs. With the prominence of Spanish-speaking countries in global interactions and political instability ravaging these countries, it becomes pertinent to create a processing tool that can parse through Spanish news for signs of political crisis. This tool will parse unstructured Spanish text into structured event code.

Automated coders (i.e. PETRARCH) are specifically designed to complete structured interpretation of political events in English and are guided by ontologies like CAMEO, a Conflict and Mediation Event Observation ontology[77, 51, 12, 50]. This ontology is laid out in such a way that the automated event coder can generate Source-Action-Target (SAT) formatted events where "Source" is the initiator, "Action" is what has been initiated, and "Target" is the entity being acted upon. The format is also known as the *who-did-what-to-whom* pattern.

However, PETRARCH is limited in that it only works for English news articles. It uses language specific parsers (i.e. Stanford CoreNLP [60]) to generate parse trees and works with relationships between words to find useful patterns representing actions and named entities. The process is hard to extend in other languages without rewriting the core logic from the ground up. This problem originates due to the non-uniform nature of metadata (i.e., parts of speech tags) across different languages.

Although substantial work has been done to convert unstructured English text to structured event code, there has been little to no work done to convert Spanish unstructured text to structured event code.

We develop a tool, the first of its kind, that facilitates structured political event code from unstructured Spanish news articles. Event coding in Spanish can be done in a number of ways. The a straightforward way is to translate Spanish text into English and perform English event coding on the translated texts. The drawbacks of this would be cost and the poor quality of translation, hindering the tool's performance. Another way is to develop a tool that will work with the raw Spanish text and perform Spanish event coding. We adopt this later strategy. To event code in Spanish, we cannot use readily-available CAMEO ontology, because CAMEO ontology defines verb-action patterns and actors in English. So, instead, we create an extended ontology that contains the verb-action patterns and actors in Spanish. Using this extended ontology, we parse Spanish texts with a language independent parser called UDPipe and match words with event code.

To facilitate event coding for Spanish texts, we make the following contributions: first, we extend cameo ontology for verb-action patterns and actors for Spanish texts; second, we utilize language independent parsers; third, we develop a full fledged framework that captures a stream of real time Spanish texts from various Spanish news websites using Apache Spark [102] and Kafka [92], parses them, and generates automated event code; finally, we develop a fully functional prototype and empirically analyze its effectiveness.

The chapter is organized as follows- Section 4.2 represents the current research and development work going into to our system. Section 4.3 highlights some key concepts and tools that help the reader understand the rest of the chapter. Section 4.4 highlights the organization of different modules in the pipeline. Section 4.5 shows results obtained on the performance of different modules.

## 4.2 Related Works

In this section we present the related works with respect to the contents of the chapter. Distributed processing of large data has been addressed by [83], [44] where author address a static dataset and the process of generating events. Here, we are working on a real-time dataset and here we need to collect and distribute the data in real-time where aforementioned works only concentrate on processing the data. We are inspired by the scalability analysis found here [83] and incorporate that to design the system. Schordt [78] has pointed out the importance of having a real-time event coding framework and data distribution. Automated Political event coding has gone through several decades and several ontologies, datasets and tools are developed. We use CAMEO ontology here. Other related ontologies are WEIS and authors [40] has provided a comparative study between those ontologies. Eck [35] also present an analysis among different conflict data-sets. Among the available datasets ICEWS [20], Cline Center Dataset [14] are prominent. But they are not easily accessible and updated in-frequently. In terms of event coder, we found PETRARCH2 is the most flexible, easy-to-extend compared to others (i.e., BBN Accent) as reported by [17, 86]. We are also using a extended version of the event coder which uses Universal Dependency [61] to better support towards foreign languages. Such type of extension was not possible to BBN Accent like proprietary software. Another dataset that matches with some types of events in CAMEO ontology would be the Global Terrorism Dataset(GTD) [1] which lists worldwide terrorist activities including different types of protest. The key aspect of the dataset is it is human annotated but doesn't link well with the original news source. We are currently working with this dataset to use it as a benchmark tool for the automatic event coder.

## 4.3 Background

To help the reader better understand the tools and methods used in this chapter, we provide key details of fundamental concepts.

**Conflict and Mediation Event Observations (CAMEO)** [77] is an ontology developed to capture political events and focuses primarily on the following four categories, also known as "Quad Classes".

- Verbal Cooperation - when two entities are agreeing or co-operating verbally on a matter
- Material Cooperation - when an entity (source) is actively helping another entity (target)
- Verbal Conflict - when two entities are in disagreement on a matter
- Material Conflict - when an entity (source) is in conflict physically with another entity (target), i.e., protest, war, etc

It works using a knowledge-base of pattern and actor dictionaries. The dictionaries contain over 6000 patterns representing 200 types of events (encoded in three digit format like, 010, 370, etc). Pattern dictionaries are helpful for identifying political interactions in a given sentence. Actor dictionaries are used to search for political actors around the matched pattern.

**Political Events** are structured pieces of information consisting of an action, source (acting entity), target (entity being acted upon) and other related information. For example, consider the following extract from a news article -

```
As coronavirus cases crop up across the
United States, some governors and other
```

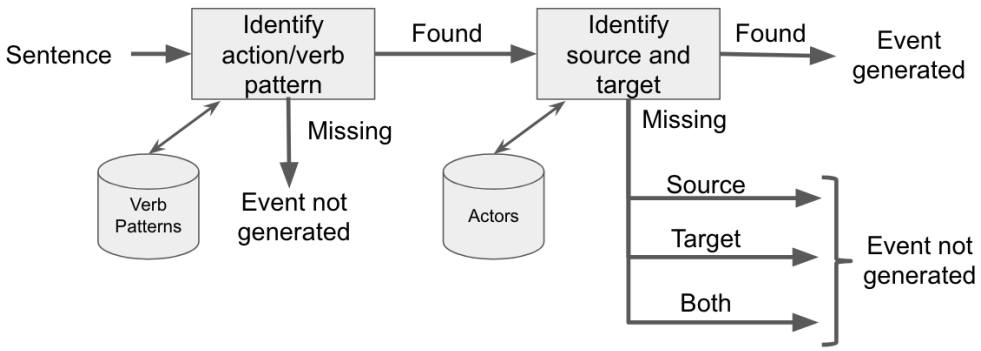



Figure 4.1: Basic Mechanism of Automated Coding using PETRARCH

```
leaders are scrambling to slow its spread,
banning large public gatherings, enforcing
quarantines and calling National Guard
troops.
```

As a structured event, it looks like the following

```
Source - ---GOV
Target - ---MIL
Action - 041 (Discuss by telephone)
```

"governors" and "National Guard troops" are coded in standard ISO-3 coding format for event generation. The structure used here is mostly known as the *who-did-what-to-whom* format of event coding. The coding mechanism is depicted in the Figure 4.1

Given a sentence, the encoder searches for a matching pattern in the CAMEO [77] verb patterns dictionary. A pattern consists of a verb and surrounding keywords. The pattern signifies a particular course of action and is represented by event code. For example, SET in the pattern "SET OUT VIEWS" indicates a MAKE PUBLIC STATEMENT type of event (event code 010). Upon finding a match in pattern, actor dictionaries search for matching entities representing the source and target. After finding the necessary information, an event

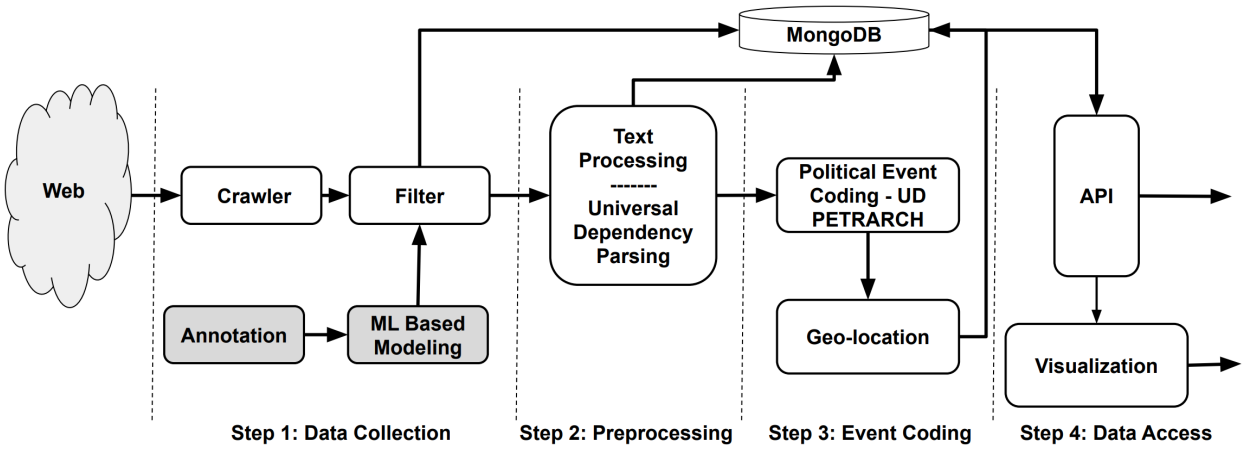



Figure 4.2: Multilingual Event Coding Framework Diagram

is coded by PETRARCH. If there is missing information, PETRARCH will ignore the event. This sequence of events are called Source-Action-Target or SAT format.

**Universal Dependency (UD)** [61] is an technique that supports cross-linguistically consistent tree-bank annotation. Its overall goal is to provide a universal collection of categories and guidelines to facilitate consistent annotation of similar constructions across languages, while allowing language-specific extensions when necessary. We will be using "ufal-udpipe" [88], a python package for generating universal dependency parse trees.

## 4.4 Multilingual Framework

We design an infrastructure to download articles from the web, process them to generate metadata, and run event coding and geolocation algorithms followed by a distribution of the data using a web based API. The following subsections will describe each stage in detail from the framework.

Figure 4.2 depicts the steps of the framework and how the pipeline works. The framework can be divided into sequential steps [100] as follows -

- Step 1: Data Collection

- Step 2: Preprocessing
- Step 3: Event Coding
- Step 4: Data Access

At Step 1, we collect Spanish news articles using Web Crawler. Then we filter out the Spanish articles and keep only the politically relevant articles using ML based filtering classifier (Filter). Filtered documents are passed through the next step (Step 2) where within the text Processing module universal dependency parse trees are generated at the sentence level (Text Processing). This step runs on Apache Spark to ensure scalability. The processed output is provided to the Political Event Coding module where UD-PETRARCH (Political Event Coding) and geolocation software work together to generate geolocated events (events along with the place where it happened) and completes Step 3. In the next step, the generated events are then distributed using the API to the Researchers and Visualization tools (API and Visualization). The steps highlighted in Figure 4.2 represents a real-time streaming data processing unit. There are other components that supports the online framework (module with highlighted background in Figure 4.4). Those are as follows -

- Model building and validation for the news article filter to support "Step 1: Data Collection" part
- Ontology translation for supporting the "Step 3: Event Coding"
- Validation of multilingual event coding w.r.t events generated in English.

### 4.4.1 Step 1: Data Collection

In this step, we run a crawler to obtain Spanish news articles using a list of news agencies from Spain, South America, Central America, and other Spanish-speaking countries. These

articles that are downloaded daily become the input for the framework. A detailed list of Spanish news websites can be found here [68].

**Article Annotation**

This step is required for Spanish news articles as we download these articles without topic restrictions and the websites often do not present a way to collect data section wise. Sometimes, a news article may also have URLs to other news articles that the crawler can follow and download. After the data collection, we will get a lot of articles that are not relevant and of those that are relevant, not all of them are political. We annotate a subset of 450+ randomly chosen articles and annotate whether the articles are relevant, and if so, whether they are focused on politics. Once an article is found to be politically relevant, we classify it into quad-class categories of the CAMEO ontology and thus create a simplified Gold Standard Records (GSR). This set of articles will be used for validating the accuracy/coverage of the event coder.

**Article Filtration - Relevant/Irrelevant followed by political/non political**

We have designed a machine learning algorithm to filter out articles that are not relevant to politics (either irrelevant articles or articles from other sections like business, sports, etc). We have used TF-IDF on Bi-grams for feature selection technique and RandomForest, NaiveBayes, and SVM as classifiers [45]. We also applied deep neural network based classifiers like, BERT [27] with its own feature extraction technique.

### 4.4.2 Step 2: Preprocessing

After filtering out the irrelevant articles, we process the remaining documents to generate metadata from raw text. The metadata includes Parts-of-Speech (POS) tags, named-entity tags along with the dependency relationships. We use ufal-udpipe python package to generate

the tags and relationship between words at a sentence level. To explain Step 2 and 3, we use Sentence 1 for reference.

**Sentence 1:** *The UN Security Council on Tuesday unanimously approved a United States' resolution on the recent deal between the U.S. and the Afghan Taliban, a rare endorsement of an agreement with a militant group.* [1]

The Spanish translation of Senetence 1 is below -

*El martes, el Consejo de Seguridad de la ONU aprobó por unanimidad una resolución de Estados Unidos sobre el reciente acuerdo entre Estados Unidos y los talibanes afganos, un respaldo poco frecuente de un acuerdo con un grupo militante.*

For each sentence in a document, we generate a dependency parse tree (Figure 4.5). The parsing job requires a lot of time to complete when ran in standalone mode. We have adapted a Spark-based distributed system for this task as it speeds up the processing almost linearly with an increase in the number of processing cores. Each Spark worker node generates a dependency parse for sentences from a batch of news articles and stores them in a MongoDB instance.

### 4.4.3 Step 3: Event Coding

After generating metadata, we process it with the UD-PETRARCH event coder [59] to generate time-stamped political events, making them CAMEO compatible events. For geolocating events, we use Cliff-Clavin [33] from MediaMeter. It gives us locations associated with the events found in the sentences. Here is a brief description on how the event coding process works for a particular sentence. Given the sentence with dependency relations, the

[1] https://www.thehindu.com/news/international/un-security-council-endorses-us-taliban-deal/article31035580.ece

Table 4.1: Noun and Verb Phrases for Sentence 1

| Noun Phrase | Verb Phrase |
|---|---|
| Council, Tuesday, U.S. Afgan, Taliban, resolution | approved |

coder first identifies the noun and verb phrases of the sentence. Examples of the phrases are listed in Table 4.1 generated from Sentence 1.

The coder then identifies the root verb, which is "approve" in this case. Then using the dependency relationship between noun phrases and the root verb, the coder creates triplets in the form of (source, action, target). An example triplet that eventually qualifies for an event is as follows:

```
u'matched_txt': u'- * + OF [080]
#line:9440',
u'source_text': u'THE UN SECURITY COUNCIL',
u'target_text': u'A UNITED STATES'
RESOLUTION',
u'verb_text': u'APPROVE',
u'verbcode': u'080'
```

Here the "matched_text" is the verb pattern in the sentence that has been found in the CAMEO verb dictionary. The source and target are matched against the actor dictionaries and once all the information are found, the triplet becomes an event.

All the processed text articles and generated events are stored in the MongoDB. All these framework components use Apache Kafka to synchronize the inputs and outputs.

### Ontology Translation From English

To capture events in articles published in Spanish or in another foreign language, we have to translate the existing CAMEO ontology to the corresponding language. There are two parts

```
--- COMPROMISE [080] ---
COMPROMISE
CONCILIATE
SETTLE
. . .
. . .
MEDIATE
INTERCEDE
INTERMEDIATE
ARBITRATE
- CRISIS WILL NOT BE * UNTIL $ &HOSTAGE RELEASED [1053]
# RESOLVE
- &FIGHT AFTER COLLAPSE OF * [190:190]  # NEGOTIATE
- SAID + HAD * CREDIBILITY [111]  # COMPROMISE
- SAID * WITH + BEAR FRUIT [050:050]  # NEGOTIATE
- * &DISPUTE TO_RESPOND + [050]  # RESOLVE
- SAID MUTINY WOULD BE * [013]  # RESOLVE
- * WITH ANYONE EXCEPT + [125]  # NEGOTIATE
. . .
. . .
```

```
--- COMPROMISE   [080]  ---
ARREGLO
SOLUCION
CONCILIAR
RESOLVER
. . .
. . .
INTERCEDER
INTERMEDIO
INTERMEDIARIO
JUZGAR
ARBITRAR
- CRISIS NO SERA * HASTA QUE &REHEN $ SEA LIBERADO
[1053]        # RESOLVE
- CRISIS NO SER * HASTA QUE &REHEN $ SEA LIBERADO
[1053]        # RESOLVE
- &PELEA DESPUES DEL COLAPSO DE *        [190:190]
# NEGOTIATE
- DICHO + HABIA * CREDIBILIDAD        [111]        #
COMPROMISE
-  + DIJO HABIA * CREDIBILIDAD        [111]
. . .
. . .
```

Figure 4.3: Snippets from English and Spanish (highlighted) verb dictionaries. The entry starts with a main verb, followed by related verbs and patterns (lines starting with "-")

that needs to be translated at this step: the verb-patterns and the list of political entities (political leaders, organizations, etc.).

**Translating verb patterns:** Verb patterns are used to capture the interaction between two entities. To translate verb-patterns, we adopt a semi automated way and develop an online application to facilitate the collaboration between human annotators. We first present a basic translation of verbs using the Wordnet [11] synsets in Spanish. Human annotators are asked to assess the quality of translation w.r.t the CAMEO code category and after feedback, we do a majority analysis to include the translation in the new dictionary [70]. A snippet of translated verb dictionaries are presented in Figure 4.3

**Translating Actors:** With this approach we translate CAMEO dictionaries containing political entities and organizations. The algorithm uses BableNet translations database to translate dictionaries written in one language to another specified language. BabelNet is a multilingual encyclopedic dictionary which was created by seamless integration of the largest multilingual Web encyclopedia - i.e., Wikipedia - the most popular computational

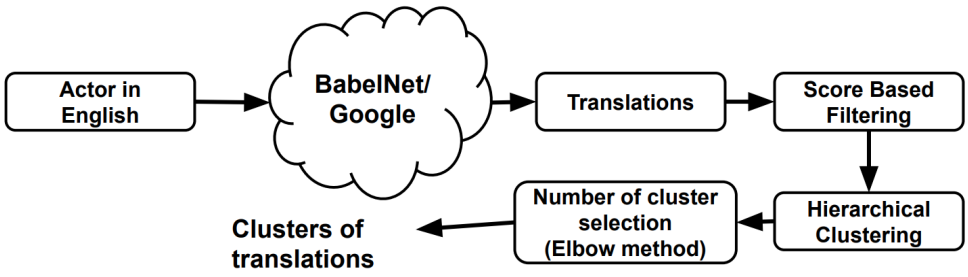



Figure 4.4: Steps in translating Actors in English to Spanish

lexicon of English - i.e., WordNet, and other lexical resources such as OmegaWiki and the Open Multilingual WordNet. We observed that the Bablenet translation database is more accurate for translating agents, actors, countries, and organization names than other translation sources such as Google and JRC databases [48].

Since BableNet is based on multiple lexical resources, like Wiktionary, OmegaWiki, Wikidata, Wikipedia infoboxes, free-license wordnets, Wikiquote, FrameNet, VerbNet, and others, it is able to overcome problems that may arise from other translators such as ambiguous and non-existing translations. Also, BableNet fills in lexical gaps in resource-poor languages with the aid of Statistical Machine Translation and it connects concepts and named entities in a very large network of semantic relations.

The translation process (depicted in Figure 4.4) takes a dictionary file in English as an input and produces a translated version in Spanish. It goes through each of the entities and their synonyms and translate them using BabelNet database[65] to Spanish. We get several translations and each of them are associated with scores. Translations with the highest scores are considered first. If no translation on BabelNet is present, Google Translate is used to do the translation task. For example, for translating the entity "AFGHANISTAN", the translation

"INVASION_DE_AFGANISTÁN_DE_2001"

comes first , while

"ESTADO_ISLAMICO_DE_TRANSICIÓN _DE_AFGANISTAN" comes second, and "INVASIÓN_DE_AFGANISTAN_DE _2001" comes in third place based on the associated score.

In the case of translating a synonym, the returned translated synonym set is clustered with other translated synonym sets belonging to the same main Entity. The algorithm leverages Levenshtein distances technique[63] to calculate the similarity between each synonym pair in the synonym set. Then the algorithm creates a two-dimensional array that stores the Levenshtein distance between each pair of synonyms in the set. After that, the algorithm builds a tree structure diagram of possible clustering based on hierarchical clustering techniques[64, 96]. After that, the elbow method[98] is applied to determine the optimal number of clusters based on the hierarchical tree. For example, if the translated synonym set is of size 8 as follows:

```
S1: 'ESTADO_ISLAMICO_DE_TRANSICION_DE
_AFGANISTÁN',
S2: 'ESTADO_ISLÁMICO_DE_TRANSICION_DE
_AFGANISTÁN',
S3: 'ESTADO_ISLAMICO_DE_TRANSICIÓN_DE
_AFGANISTAN',
S4: 'EJERCITO_DE_AFGANISTAN',
S5: 'EJERCITO_NACIONAL_AFGANO',
S6: 'PROVINCIA_DE_PARWAN',
S7: 'PROVINCIA_DE_BĀMIYĀN',
S8: 'PROVINCIA_DE_BAMIYĀN'
```

And they are numbered as S1 to S8 Then the two dimensions array of the distances are listed in Table 4.2:

Table 4.2: Distances between translated texts identified as S1 to S8

| | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 |
|---|---|---|---|---|---|---|---|---|
| S1 | 0 | 1 | 2 | 25 | 28 | 31 | 30 | 29 |
| S2 | 1 | 0 | 3 | 25 | 28 | 31 | 30 | 29 |
| S3 | 2 | 3 | 0 | 24 | 29 | 30 | 31 | 30 |
| S4 | 25 | 25 | 24 | 0 | 12 | 14 | 15 | 14 |
| S5 | 28 | 28 | 29 | 12 | 0 | 16 | 19 | 18 |
| S6 | 31 | 31 | 30 | 14 | 16 | 0 | 6 | 5 |
| S7 | 30 | 30 | 31 | 15 | 19 | 6 | 0 | 1 |
| S8 | 29 | 29 | 30 | 14 | 18 | 5 | 1 | 0 |

Then, a tree structure diagram of possible clustering is built based on hierarchical clustering techniques, After that the elbow method is applied and result in three clusters based on hierarchical clustering as follows:

```
Cluster 1:
'ESTADO_ISLAMICO_DE_TRANSICION_
DE_AFGANISTÁN',
'ESTADO_ISLÁMICO_DE_TRANSICION_
DE_AFGANISTÁN',
'ESTADO_ISLAMICO_DE_TRANSICIÓN_
DE_AFGANISTAN'

# Cluster 2
'EJERCITO_DE_AFGANISTAN',
'EJERCITO_NACIONAL_AFGANO'

# Cluster 3
PROVINCIA_DE_PARWAN
'PROVINCIA_DE_BĀMIYĀN',
```

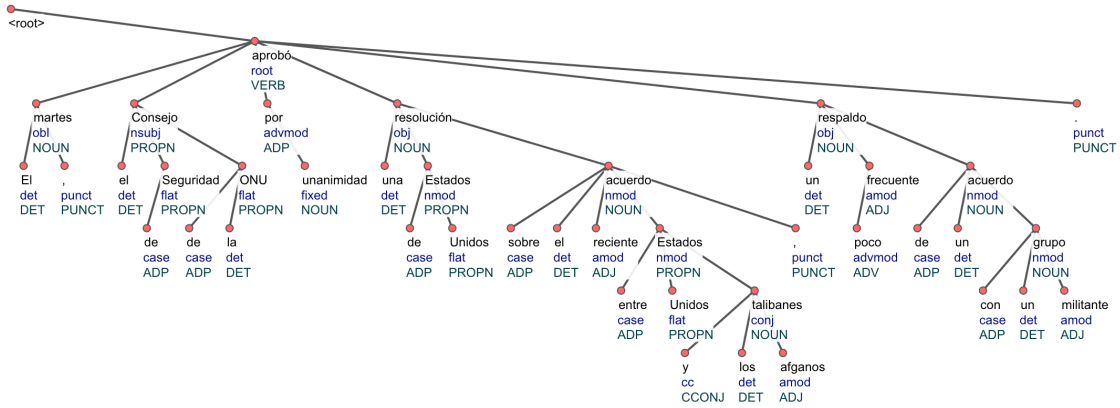



Figure 4.5: Universal dependency tree for Sentence 1 in Spanish

PROVINCIA_DE_BAMIYĀN

### Multilingual Event Coding in Spanish

We are using UD-PETRARCH event coder that works on universal dependency rather than only the Parts-Of-Speech tags used by original version of PETRARCH. It is helpful for multilingual event coding because of the uniformity of universal dependency parses across language in comparison to the inconsistency of Parts-of-Speech tags across languages. Using a universal dependency based event coder, we can easily incorporate coding in other languages with no effort needed on updating the event coder itself. We will still need the dictionaries to be translated first. Using a universal dependency parser, however, gives flexibility for the non-CS background researchers to solely focus on the analysis and ontology extension parts.

Figure 4.5 shows how the dependency relations are structured between words in the translated version of Sentence 1.

Once again the root verb here is "aprobó", meaning "approve" in English. This time UD-PETRARCH genrates the following triplet

```
'matched_txt': '- * + OF [080]

#line:9440',
```

```
'source_text': 'EL CONSEJO DE
SEGURIDAD DE LA ONU',
'target_text': 'A ESTADOS
UNIDOS RESOLUCIÓN',
'verb_text': 'aprobó',
'verbcode': '080'
```


and we find the same event where the source is USA, the target is IGOUNO, and the event type code is 080.

**Cross-lingual validation for generated events.**

To identify whether similar events reported in different languages can be captured by the event coder, we run the following validation procedure. First, we select a set of documents in Spanish that has been annotated to be politically relevant. Then we translate them to English using Google Translate API. Afterwards we have a parallel corpus of English and Spanish documents. Then we run respective language parsers to generate universal Dependency relationships and feed them to the UD-PETRRACH event coder. We observe the generated triplets and do a semantic comparison between reported source and target text with the help of BabelNet [65]. We also follow whether they are reporting the same event type code. We will highlight the findings in Section 4.5.

### 4.4.4 Step 4: Data Access

We provide API to serve generated event data. This API maintains API-key based access restrictions and serves the data in JSON format. Interested users can query using a JSON based query language (similar to MongoDB). Users can select the portion of the data they are interested in by subsetting the data with query parameters. They can also focus on particular

fields on each record, aggregate/group the query results, and query for the metadata about the datasets. Additionally, there are user defined libraries in R programming language [52] built on top of the web-based API to make accessing the data easier. Details of the access policy can be found here [7].

## 4.5 Experiments

In this section we will discuss about the experiments conducted for different modules of the framework.

### 4.5.1 Scalability: Universal Dependency Parse generation

As we point out earlier, the universal dependency parser is the most compute-intensive task in the pipeline. We adopt a distributed system based on Apache SPARK and Kafka to parse multiple documents in parallel. Each worker node in the Spark cluster gets a subset of the documents to process. The synchronization is maintained by Kafka, guaranteeing that there is no duplicate in processing task. Figure 4.6 show the relationship between the execution time and number of processing cores and follows a linear monotonic decreasing function. For this experiment we selected a subset of 1,400 Spanish news articles and generated dependency parses for each of the sentences in those articles.

### 4.5.2 Document Translation vs Ontology Translation

With our current approach, we translate the CAMEO ontology to support the event coding framework for Spanish news articles. Another route was to translate the articles into English first and then apply an English event coder to generate the events. However, the cost to use paid APIs, like Google Translate for that which would have cost us $125,000 for the current corpus of 2.5 Million Spanish news articles. Also, the cost would have increased with time

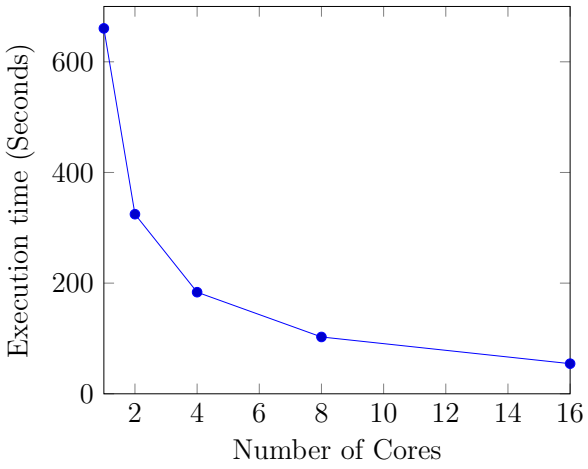



Figure 4.6: Relation between execution time and number of available processing cores.

Table 4.3: Comparison between English and Spanish Event coding on parallel corpus.

| | |
|---|---|
| Number of news articles | 132 |
| Number of events generated(English) | 107 |
| Number of events generated(Spanish) | 98 |
| Number of events matched exactly | 78 |
| Number of events matched in Event code | 93 |

as the corpus gets larger. This calculation is based on the estimate by Google who states $0.05 is required to translate a document with around 500 characters [3].

**Event coding coverage across languages**

In this part of the experiment, we run the event coders in parallel for a set of Spanish articles and the English translated version of those articles. We observe the similarities between the events from the original and the translated version. Statistics are shown in the Table-4.3. As we observed, a large number of events matched exactly with each other. Of the remaining ones, there were partial matches present (ex. USAGOV captured in English as source, GOV in Spanish for the corresponding event).

Table 4.4: Accuracy of different classifiers

| Classifier | Accuracy |
|---|---|
| Naive Bayes | 81.4 |
| Support Vector Machine (SBF) | 80.7 |
| **Random Forest** | **82.7** |
| BERT | 74.6 |

### 4.5.3 Article Filtration using ML Classifier

Using the annotated $\tilde{4}50$ documents, we create a multiclass classifier that tags each document as irrelevant, political and non-political. We observe an overall accuracy of 82.5% with Random Forest classifiers (Table 4.4) and average accuracy of 67% among the three classes. In our observation, we found that the classifier struggles to differentiate between political and non-political news articles. Most of the irrelevant articles (i.e. ads, homepages, etc.) were identified with reasonable accuracy with no false negatives (articles identified as irrelevant but are actually political/non-political). We also applied Deep Neural Network (DNN) model BERT [27] but found it less accurate ($\sim$ 75%) than the traditional machine learning model, due to less annotated data to train the DNN model.

### 4.5.4 Ontology Translation

Here we discuss about the verb pattern translation app (VTA), where first the coders select correct English synonym sets from WordNet w.r.t a particular CAMEO code and then identify appropriate translation using Spanish synsets. They provided 10,358 correct verdicts (35.8% of the total verdicts) for English synonym set. Of those correct sysnsets, 90.1% of the Spanish translations are regarded as correct which shows the effectiveness of WordNet[11] based translation. The verb translation app [70] also provides valuable insights for inter-annotator reliability. The system tracks individual annotator's verdicts and generates an overall sense of agreement among them. In general, there is a correlation factor of 0.96 between the average proportion of correct and incorrect verdicts across users. This indicates

a high degree of agreement between annotators in which they consistently identify English synsets as correct when they indeed correspond to the corresponding CAMEO concept, and consistently tag them as incorrect when their definition does not align with CAMEO's meaning[70].

# CHAPTER 5

# EXPLORING THE ROLES OF SOCIAL MEDIA DATA TO IDENTIFY THE LOCATIONS AND SEVERITY OF ROAD TRAFFIC ACCIDENTS

## 5.1 Introduction

Social media is currently shaping the way news spreads as it allows people to know about an event before it is published in broadcasted or printed media (e.g., tv, radio, or newspapers). Furthermore, a lot of people publish news via social media through their accounts, which is very often concise and easy to interpret. Keeping aside the false information or fake news, the amount of relevant real-world news is available on a larger scale through social media along with irrelevant information. This motivated us to focus on mining social media feed for accident-related reports and gather key information such as location and severity of an accident.

Twitter, a social media outlet, is proven to be a well-defined source of information during different historical events and disasters; for example, it was used to predict election outcomes around the world [32, 37]. Another time Twitter acted as a medium to spread information was during a devastating earthquake in Nepal. The data provided gave people information on the needs of the victims as well as different campaigns to raise funds [90]. For the recent worldwide pandemic of Covid-19, Twitter is acting as a reliable source of information that can be used to predict the second wave of Covid-19 and people's sentiment on the different preventative measures (i.e., Social Distancing [19]). For all these scenarios, Twitter acts as a giant hub of information driven by a massive community of 330 Million users [99].

Traffic accidents are currently in a steady growth cycle due to increased load on traffic, distracted driving, and various other causes. There is already a considerable amount of research conducted in different locality and at different scales [43, 103]. However, in the developing world, the problem is more prevalent due to less traffic maintenance as the World

Health Organization statistics show that the mortality rate is highest in Africa (26.6 per 100,000 population) and lowest in Europe (9.3 per 100,000 population). For instance, every year, 39,000 Nigerian people are killed in traffic accidents [13]. Therefore, immediate and adequate responses are needed to control such high levels of causality. Currently, only a limited number of studies have used historical tweets to obtain accident-related information, but they did not consider grouping tweets that convey the same information [43]. Additionally, severity detection based on the message content may be beneficial for care providing agencies, especially when medical aid resources are limited. In the developing world with poor road networks, it is more important because the aid might not be similarly spontaneous compared to the developed world. For example, analyzing traffic incident data from Lagos, Nigeria, we found the average delay in response by the medical aid providers to be between 45 to 90 minutes, whereas in Dallas the response time is roughly between 8 to 15 minutes. The longer delay is mostly because of the poor road network and lack of medical supply. Hence, any re-evaluation of available supply and requesting more when needed will not be feasible in most cases. So, the better and quicker we can assess the situation, the more of a chance we have to save lives.

While using social media data to find accidents, key challenges like location extraction and information dissemination steps, remain in data collection. During the data collection phase, we find Twitter to be most accessible with its easy-to-use API. The keyword-based search results contain tweets in different contexts and necessitate a context-driven filtration outside of the API using machine learning techniques to identify accident-related tweets. We learned that the location reported by tweet's metadata is almost non-existent so that motivated us to adopt content-based location identification (i.e., geolocation) and refine the results using semantic properties of tweets. Once the tweets are collected, the efficient distribution of the extracted information becomes the next point of focus, where we group similar tweets. We find key-insights (e.g., the severity of an accident) from tweets using Machine Learning(ML)

and Natural Language Processing (NLP) techniques. This extracted information will help medical-aid providers to assess the overall situation related to the accident and allocate their resources accordingly. Processing large number of tweets requires application of distributed processing and we use Apache Spark [102] based infrustructure to accomodate that. Apache Kafka [92] is used for synchronizing the streaming nature of incoming tweets along with guranteed delivery at different steps of the framework.

In this chapter, we propose a comprehensive framework that will provide supportive information related to traffic accidents reported on Twitter. More specifically, we will use current state-of-the-art machine learning (ML) and natural language processing (NLP) techniques to extract location information and severity.

The proposed framework works in real-time with active and passive modes of operation. In the active mode, it collects tweets continuously and identifies different accidents based on the information in the tweets. In the passive mode, it takes input about some information related to the accident (i.e., possible location, type, and so on) and collects geographically relevant tweets that express accident-related information and extend the knowledge about that accident. The framework has the following key steps - First, we collect and filter accident-related tweets using Twitter API. Second, we perform tweet filtration via machine learning classification where it identifies the context of a tweet representing an accident. Third, we arrange the tweets into groups or clusters when they are related to the same accident. Fourth, we provide a summarization of a group of tweets, highlighting key information about a particular accident. Fifth, we extract geolocation based on tweet text and associated metadata. Finally, we conduct severity analysis based on text mining approaches.

The chapter is organized in the following ways. Section 5.2 summarizes related works to provide the context of the chapter. Section 5.3 provides some background on underlying concepts. Section 5.4 describes the data collection and processing pipeline of the twitter-

based accident information. Section 5.5 shows how to interface with our system. Section 5.4.1 describes the dataset and provide system specification. Section 5.7 represents empirical analysis and experimental results for different modules.

## 5.2 Related Works

In this section, we will describe related works for twitter based event detection, location extraction from tweets and finally, summarization of tweets.

### 5.2.1 Event detection

Twitter-related event detection is an interesting problem and attracts the research community in computer and transportation science. Li et al. [56] identified social events from tweets along with the temporal-spatial information and the importance of the events. Other notable applications addressed the early detection of contagious disease outbreaks by monitoring influenza-related blogging trends during the emergence of the U.S. 2008 flu season; which demonstrated the role of analyzing the structure of social media networks in improving the prediction of the spread of these outbreaks [26]. Influenza epidemic outbreaks were also detected in the same application domain of bioinformatics [42]. Earle et al. attempted to assess how fast tweeters reacted to the small and localized earthquake of Morgan Hill, California, in March 2009 [34]; while Crooks et al. also identified spatial characteristics of this information dissemination avenue [34].

### 5.2.2 Traffic Incident Identification

Gu et al. [43] focused on the identification of accidents in tweets in the Pittsburgh and Pennsylvania areas, however, their work is different from our work in the following ways. First, they used geolocation services but did not consider the content of the tweets to extract

the location. In our case, we utilize both to extract the location. Second, we aggregated similar tweets that hadn't been considered in their work. Finally, their work is narrowly focused based on PA area whereas our work not only focuses on cities in the US but also cities from developing countries like Nigeria.

### 5.2.3 Location extraction from tweets

Here we will present current state-of-the-art approaches to extract locations from tweets. Many of them rely on the geotags associated with the tweets rather than using the analysis to extract location from the text itself. However, implicit location identification based on activity analysis on Twitter is also an emerging topic that can benefit the location identification for related events. Li et al. [55, 23] identified the location with temporal information for a particular Twitter user based on the historical tweets containing check-ins in different Point of Interest (POI) collected from Foursquare. The method will be effective in a way that whenever a user publishes a tweet related to an accident, the user's tweet history can be analyzed to identify her/his recently visited places. From that information along with some additional temporal analysis, a possible location of the accident can be identified. Studies [43] show that usually, around 70% of the tweets are from a well-known organization or media accounts when it is related to accidents. Those accounts, however, are not as dynamic as a regular, personal Twitter account when it comes to location changes Zhang et al. [103] suggested a hybrid mechanism based on latent Dirichlet allocation and document clustering for modeling incident-level semantic information, while spatial point pattern analysis would be applied to explore the spatial patterns and to assess the spatial dependence between incident-topic tweets and traffic incidents.

### 5.2.4 Summarization of Tweets

Inouye et al.[49] evaluated previous works in text summarization applied for tweet summarization, such as SumBasic [95], MEAD [94], Phrase Reinforcement (PR) [81], and Hybrid TF-IDF [82]. All of these algorithms are briefly described as follows: SumBasic is a notable text summarization algorithm that performs well in tweet summarization. The primary idea of SumBasic came from the tendency of a manual summary to include more frequent words in documents as part of the summary. MEAD and other clustering approach did not perform as good as SumBasic for tweet summarization. Phrase Reinforcement (PR) is a unique algorithm because it uses word graphs to summarize tweets. The key concept behind PR was to determine and combine partial summaries from tweets. Another summarization algorithm specifically for tweets is hybrid tf-idf. The "hybrid" keyword came from the adaptation of tf-idf weighting for short documents, particularly in its different definition of one document when calculating tf and idf value of terms. The performance of hybrid tf-idf came second to SumBasic in ROUGE scores, but slightly better in human evaluation. Other methods in tweet processing are TwitterStand [75] and TweetMotif [67]. TwitterStand did not explicitly mention summarization, but their intention to find current event and provide several representative tweets were very similar to tweet summarization task. In contrast, TweetMotif explicitly intended to summarize a topic but did not limit their topic to only trending topics and rather focused on topic modeling. We adopted these approaches and while also providing severity information extracted from the summary.

## 5.3 Background

In this section, we provide insights into some key concepts/tools used in the project. This will be helpful for the reader to understand the subsequent sections.

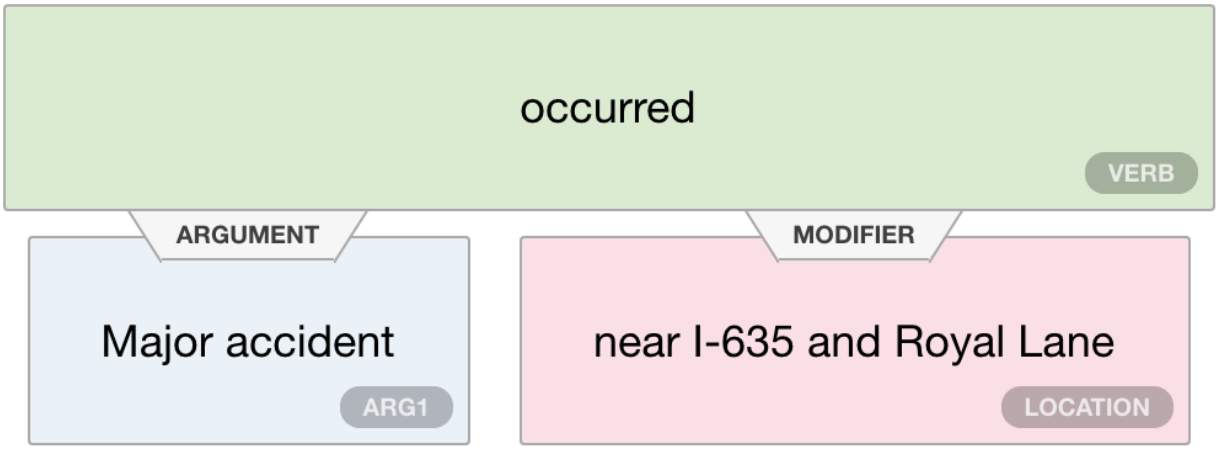



Figure 5.1: Example of Semantic Role Labeling

**Semantic Role Labeling**

Semantic Role Labeling (SRL) is the task of determining the latent predicate-argument structure of a sentence and providing representations that can answer basic questions about the sentence like, who did what to whom, etc. For example, given the following sentence,
*Major accident occurred near I-635 and Royal Lane.*
has the semantic structure with the key verb "occur" and associated argument and location phrases (as shown in Figure 5.1).

We use AllenNLP[36] software stack to compute the semantic role labels.

**Bidirectional Encoder Representations from Transformers - BERT**

BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modeling. This is in contrast to previous efforts that looked at a text sequence either from left to right or combined left-to-right and right-to-left training. The study results [28] show that a language model that is bidirectionally trained can have a deeper sense of language context and flow than single- direction language models. They detailed a novel technique named Masked LM (MLM) that allows bidirectional

training in models, which was previously impossible. One key application of BERT is to help in situations where the training dataset has fewer number of instances. It uses a pre-trained language model trained on Wikipedia and similar corpora. The task-specific instances will be used to fine-tune the pre-trained model to increase the domain-specific accuracy.

## 5.4 Framework

In this section, we describe the framework in detail and use Figure 5.2 as a reference. There are two modes of operation - *active* and *passive*.
In active mode, we collect tweets continuously from the Twitter API and extracted accident-related information if there is any. The pipeline described in this section represents this mode of operation.
In passive mode, we expect a trigger from an external system with location and time information to initiate a twitter search for accident related tweets. It also includes the pipeline with a request handler in between to handle the request and serve the data. Details will be provided in Section 5.5.

The pipeline works as follows. It downloads the data through Twitter API in a regular interval of 2 minutes (See Section 5.4.1). It filters out the non-accident-related tweets and gives them as input to the classification based filtering module (See Section 5.4.2). After the classification based filtering, we group similar tweets based on semantic, temporal, and spatial attributes (See Section 5.4.4). This group of tweets represent information related to a particular accident. Then, we apply summarization to get the most important information from the tweets (See Section 5.4.5). After summarization, we record this group of tweets as a particular accident and identify the severity using text mining techniques (See Section 5.4.6). We also extract location from text and metadata using a text mining and geolocation tool-based approach (See Section 5.4.3).

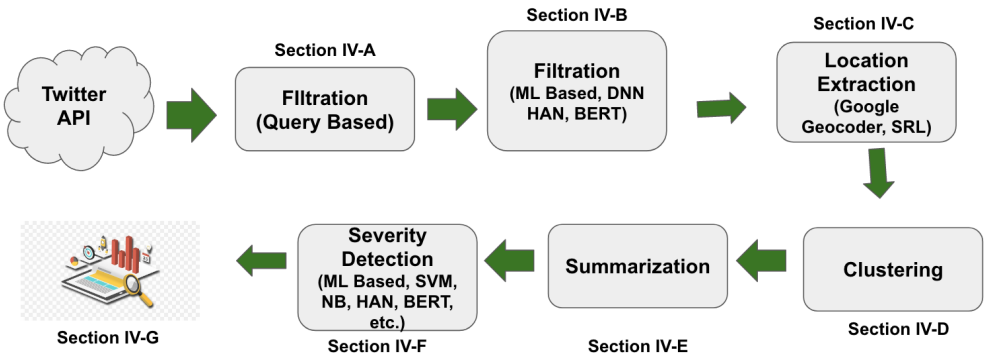



Figure 5.2: Framework for accident-related tweet processing

### 5.4.1 Filtration (Query based) and Data Collection

We collect the data from Twitter API via their search API. We use "accident" as the keyword and perform a keyword-based filtration. We send request every two minutes and collect around 100 new tweets that contain the "accident" keyword, either as a regular word or hashtag. The tweets are downloaded in the extended mode where we get the text content of length 280 characters.

We use only the keyword "accident" as it is less restrictive than using phrases (i.e. "car accident", "road accident", etc) Every tweet stored in the database includes tweet specific and user-specific information. User-specific information is stored under the "user" attribute. One additional attribute is "searched_city", which indicates the city where we collected the tweets. It includes the center of the city as a geographic coordinate (latitude and longitude) and we consider the radius to be 12 to 25 miles (depending on the size of the city itself) to represent the city limit as a circle.

### 5.4.2 Filtration (ML based) - Context driven classification of tweets

After downloading the tweets, we consider whether the tweets are accident-related or not since the keyword "accident" may be used in different contexts. For instance, both of the following two sentences use "accident" but not in the same context.

Table 5.1: Examples of Non-accident vs Accident related tweets

| **Non-accident-related** | **Accident-related** |
|---|---|
| The National Transportation Safety Board (NTSB) has released a Marine Accident Brief about the Carnival Horizon's A... | Vernon, NY – Five Teens Injured in NY-26 Car Accident |
| Are you sure James Charles termination was on accident?! | Accident, two lanes blocked in New York on The FDR Drive SB at The Manhattan Br/South St/x3, stop and go traffic ba... |
| New soft-muscled RoboBee is accident proof. Read more: | New post: One dead, nine injured in Lagos-Ibadan Eressway accident |

Sentence 1 : *I left the wallet in office by accident*

Sentence 2 : *Ghastly accident on i-45, 2 dead*

The first sentence signifies a mistake, while the second signifies a car accident event. For these types of tweets, we need to differentiate true accidents from other unfortunate events. Table 5.1 shows some examples of tweets falling into these two contexts.

Those tweets express information about certain road accidents and contain location information, casualty information, etc. This information is extracted further down in the pipeline.

For classification purposes, we use Google's BERT pretrained model [28] and fine-tune with annotated samples from a set of tweets. We also run other classifiers using shallow learning (i.e., Naive Bayes, SVM) and word-to-vec based features from tweets. We convert each word to a 300-dimension vector using word-to-vec [62] and perform an unweighted averaging of words to retrieve a 300-dimension representation of the tweet itself and later use that as input for the models. Related performance data and comparison are highlighted in the Section 5.7.

The regular preprocessor works for tokenization, stopwords removal, and lemmatization purposes. The BERT preprocessor has those steps built-in including the vectorizer.

The key motivation for using BERT for classification purposes is its effectiveness in being trained with small datasets compared to other deep neural networks (DNNs). Recall that BERT uses a pre-trained model and our data is used for domain-specific fine-tuning purpose. As we will see in Section 5.7 we have a small number of annotated examples that BERT model can perform well compared to others.

### 5.4.3 Location Extraction

To better identify the location mentioned for an accident, we use both methodical and tool-based approach and extract location from the text. We also consider the metadata associated with the tweets (i.e., geo-tags, user-location, and so on) to infer the location of the accident. For methodical approache, We use a semantic role labeling approach for identifying location based on smeantic tags in tweets. For tool based approach, We consider different geocoder tools[72] and the Google Geotagger results are presented in the chapter as they perform the best among all geotaggers. We capture location information at street level whenever possible. A detailed table and examples are shown in Section 5.7. Algorithm 3 briefly describes the steps involed.

Based on the methods/tools, the location extraction is done at 3 steps. First, we consider associated **metadata containing user location and geo-tag field**. The geo-tag field indicates from where the tweet was posted. As most of the people do not allow their tweets to be geo-tagged, we don't find it containing any value in the collected tweets. The profile location is not dynamic since it does not change when the user moves to a different location. We find around 35% of the tweets have a meaningful location in form of a city name or city-country name pair that can be derived from user profile (See line 3 to 6 of Algorithm 3). Second, we consider location extraction using **geocoding**(line 7). It is conversion between georeferenced information (i.e., latitude and longitude values) and address or location. "Forward geocoding" (or simply "geocoding") converts address to geocode, while "Reverse geocod-

**Algorithm 3** Location Extraction From Tweets

```
1: procedure LocationExtraction
2:     tweet ← accident related tweet
3:     location ← geo_tag(tweet)
4:     if location ≠ NONE then
5:         return location
6:     else
7:         user_location ← user_location(tweet)
8:         geo_coded_loc ← geocode(tweet)
9:         semantic_coded_loc ← semantic_role_based_loc_extraction(tweet)
10:        location ← get_most_detailed_address(
11: semantic_coded_loc, geo_coded_location)
12:        if location ≠ geo_coded_loc(tweet) then
13:            location ← geo_code(location.text)
14:        end if
15:        if location = NONE then
16:            location ← user_location
17:        end if
18:        return location
19:    end if
20: end procedure
```

ing" converts geocode to address. We use the geocoding function from Google Geocoder to identify the location for the tweets and their corresponding latitude and longitude values, if possible. For example, geocoding the following tweet with Google's Geocoder pinpoints the location "I-635 And Military Pky Mesquite TX" in Google Maps, as depicted in Figure 5.3.

```
CLEARED - accident:I-635  southbound

TX-352/Military Pky/Exit 4 Mesquite

various Lns blocked
```

Google's geocoding approach has been widely used by other open-source geocoders. It describes how different steps in fuzzy searching work for geocoding of input addresses. First, geocoding performs a lexical analysis on an input address to obtain its geocoding information. In one aspect, the lexical analysis may include at least one of a parsing operation, an abstraction operation, and a stretch operation. Second, it performs a fuzzy searching on

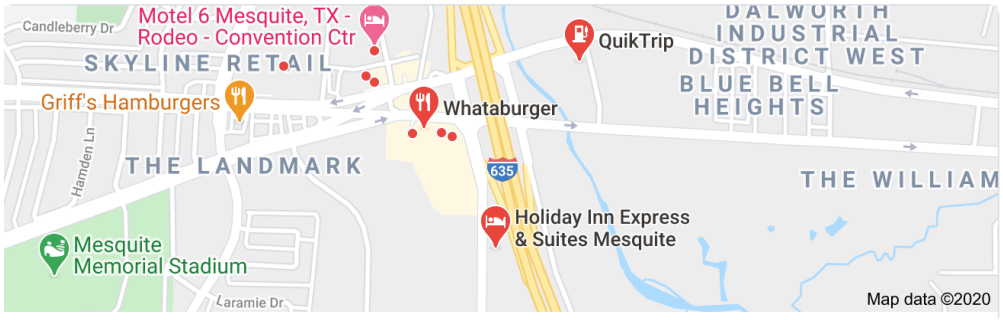



Figure 5.3: Google Maps showing the location identified by Geocoder

knot-sequence tree, using the portions of the input address, to identify a plurality of partial addresses which may match with the input address. Third, it computes a transposition and matching score for each of the identified plurality of partial addresses to determine the best matching candidate for the input address. Finally, it uses the geocoding database to find out the best matching candidate for obtaining the geocoding information of the input address.

Finally, we use **semantic role labeling** to disambiguate the location descriptions found in previous levels(line 8-9). Semantic role labeling (SRL) tags a phrase in the sentence as a verb or an argument associated with the verb. It identifies the location phrase as ARG-LOC, which can help identify any location mentioned in the tweet. Given the following tweet,

```
Ghastly accident occured on a
bridge near gt bank on okota road.
```

The *semantic_role_based_location_extraction* identifies the location as "bridge near gt bank on okota road", which provides more context than just "gt bank on okota road" identified by the *geocode*(). So if we consider geocoding, we will only identify "gt bank on okota road" and it's related metadata including approximate latitude and longitude, but semantic role labeling will provide more context to derive finer-grained location as it captures more detailed location expression (i.e., "bridge near gt bank on okota road"). *get_most_detailed_address*() (line 9) compares two addresses based on different components like street names, city, point of interests (i.e. "gt bank" in the current example), reference points (i.e, "bridge") and

gives more detailed one. If semantic role based approach gives the more detailed version of the address, we again use geocoder to derive the address components (latitude and longitude) in lines 11-12. This time we use only the location expression identified by the *semantic_coded_loc*. If geo-coder gives new geo-coordinate value we consider that value instead of *geo_coded_loc*. Otherwise, we consider both the *semantic_coded_loc* (provides more context) and the previously geocoded address, *geo_coded_loc* (provides latitude and longitude with proximity near to actual location of accident).

To get most detailed address (i.e. *get_most_detailed_address*), we compare the parts-of-speech tags between the arguments. We consider words with tags NN and NNP and form sets for *semantic_coded_loc*, $S$ and *geo_coded_location*, $G$. $S$ is derived from the ARG-LOC identified by semantic parser and $G$ is calculated based on the $formatted\_address$ attribute of geocoder's output. Then, We consider set difference i.e. $(S \setminus G)$. If it contains any element then *semantic_coded_loc* captures more detailed location, otherwise *geo_coded_location* is. For *geo_coded_location*, we consider $formatted\_address$ attribute, included in the output of geocoder.

For example consider the tweet mentioned above. Here, the $formatted\_address$ returned by Google's geocoder is "Okota Rd, Ilasamaja, Lagos, Nigeria", and *semantic_coded_loc* is "bridge near gt bank on okota road". In this case set S is {"*bridge*", "*gt*", "*bank*", "*road*"} and D is {"*Okota*", "*Rd*", "*Ilasamaja*", "*Lagos*", "*Nigeria*"}. $S \setminus G$ becomes {"*bridge*", "*gt*", "*bank*"} and is non-empty. So, we consider *semantic_coded_loc* will have more detailed information in this scenario.

### 5.4.4 Clustering - Forming groups of similar tweets

The motivation behind clustering is to identify related tweets that provide information about the same accident. In an ideal scenario, the number of clusters will be equal to the number of distinct accidents. It will also help us in the next step where we summarize the cluster to

make a single, unambiguous, tweet-like summary from the tweets in the same cluster. We use online clustering[24] as it does not require a target number of clusters like many other clustering algorithms (i.e., K-Means[57]). Since we consider spatial and temporal dimensions in the clustering process, incremental clustering will also be amenable for this purpose.

After the location extraction step, the relevant accident-related tweets are identified and grouped if they belong to the same accident. We keep a set of clusters divided into semantic, temporal, and spatial dimensions. For the semantic dimension, we consider tweet vector similarity based on word-to-vec [62]. A tweet vector is calculated based on an unweighted average of words in that tweet. For the temporal dimension, we set a fixed time-to-live for each cluster and use the cluster creation time to weigh new incoming tweets. For the spatial dimension, we consider the location information found either in text or using metadata (i.e., tweet geo-tag, user-location, etc.). Some examples of tweets falling into the same cluster are presented in Figures 5.4 and 5.5.

We use an incremental approach for clustering the tweets where for any new incoming tweet, the similarity with the existing clusters is calculated first. The similarity score lies between 0.0 and 1.0, where 1.0 means perfect similarity. The similarity calculation is based on the vector of the candidate tweet and the cluster centers. A cluster center is the normalized average of the tweet vectors multiplied by their similarity score. Cluster membership is granted once the score exceeds a certain threshold. Otherwise, a new cluster is created. We also use a Least Recently Used (LRU) based cluster eviction technique with temporal constraint. Temporal constraint is given as a time-window starting with the time cluster initially created, $t_{start}$ and has length of 60 minutes ($t_{end} = t_{start} + 60 minutes$). Within that time-window, the LRU approach works every 10 minutes to score the clusters based on new insertion. If a cluster is not selected for LRU eviction after 60 minutes, it is granted another 60 minutes to live. Once selected by the LRU eviction model, a probabilistic score is generated based on how many times it was flagged by the LRU scheme.

```
‘BREAKING NEWS: New York radio
legend Angie Martinez involved
in severe car accident. Currently
recovering from frac...’,

‘New York radio personality Angie
Martinez recovering after suffering
multiple injuries in ""severe car
accident""’,

‘Radio Host Angie Martinez Suffers
Shattered Vertebrae in Severe Car
Accident\n\nAngie Martinez is
in recovery after s...’

‘New York radio legend Angie Martinez
involved in severe car accident.
Currently recovering from fractured
lumbar and shattered vertebrae’
```

Figure 5.4: First example of Cluster of tweets

### 5.4.5 Summarization - summary for a cluster of tweets

As we collected a large number of tweets, a lot of tweets are semantically “duplicate”, meaning they convey the same information but are written in different linguistic forms. The goal of this step is to provide a concise overview of the duplicate tweets. More specifically, we want to isolate crucial information from the duplicate tweets and generate a single abstractive summary tweet that represents the informativeness of those tweets. In the clustering step, we identify the tweets that represent the same accidents. A single tweet that summarizes the information about that particular accident could provide key insights from those tweets.

We follow the approach in [74] to summarize the tweets. First, we build a word-graph, which is a graphical representation of a sentence where nodes represent words and edges represent a link between words. We generate bigrams from each tweet of a particular cluster

```
* A man driving drunk left the scene
of an accident following a crash that
injured a pedestrian in ,...

Man dies, injured on Lagos-Ibadan
Eressway accident - ...

One killed, nine others injured in Lagos
-Ibadan Eressway accident - The Punch

Man dies, nine injured on Lagos-Ibadan
Eressway accident

Overtaking: Man dies, injured in
Lagos-Ibadan E/Way accident
```

Figure 5.5: Second example of Cluster of tweets

and add the bigrams to the graph iteratively. Then, we generate all possible word paths, each of which denotes a sentence or in this case a tweet, by traversing the graph. As this process may generate illogical sentences, we need to select the sentences which convey more information and are more linguistically correct. To ensure informativeness, we compute the cosine similarity value between the tf-idf vector of each word-path and the average tf-idf vector of all word-paths. To ensure that the word paths represent more realistic sentences, we use a tri-gram language model, which computes the probability of possible sequences of word occurrences. We train the language model with text corpus from COCA (Corpus of Contemporary American English) using the KenLM [47] tool and calculate the score of each word-path. We then select the best scoring word-path from the objective function containing the two scores by following the method in [74]. This word-path represents the summarized sentence or tweet with the highest linguistic and informativeness score of all the tweets from that particular cluster.

Figure 5.6 illustrates an example summarization from tweets of the cluster presented in Figure 5.4. The detailed procedure is depicted in Figure 5.7.

```
break news new york radio personality
angie martinez recover fractured lumbar
shatter vertebrae severe car accident
```

Figure 5.6: Example of summarization of a cluster of tweets in Figure 5.4

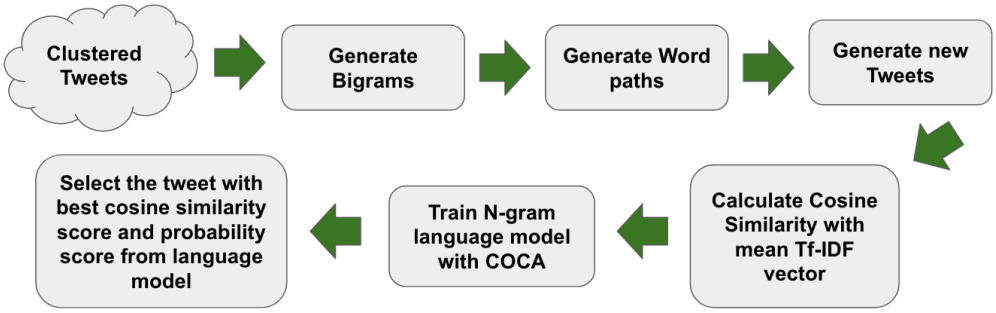



Figure 5.7: Framework for summarization

The main motivation of using this approach over other approaches [95, 94, 81, 82] is to have a summary that is more semantically close to a human-written like tweet and easier to interpret by the user.

### 5.4.6 Severity Detection

Detecting the severity is a crucial step to properly assess the situation and correctly estimate the damages, which will allow proper allocation based on given limited resources. For severity analysis, we attempt to extract information about casualty and injury from the twitter text, which allow the user to interpret how severe the accident is. We use a rule-based approach to extract information about the severity and trained machine learning models for classification of severity levels with 3 classes to consider.

We use the verb-arguments structure of semantic role labeling to identify the injury or casualty data. We start with a seed list of verbs and extend the verbs based on the cluster of tweets. Algorithm 4 describes how the process works. We start with the seed list of verbs (line 2), consider each tweet in a cluster (line 4), and then try to find verbs from the seed list and its associated argument. If successful, we search for arguments in the other tweets in the cluster where no verb was identified and add that verb to the list (lines 12-17). We match arguments based on semantic similarity using word-to-vec. If none of the tweets has a verb that indicates some level of casualties, we ignore the cluster.

**Algorithm 4** Identifying semantically labelled arguments related to accident severity

```
1:  procedure SeverityInformationDetection
2:      arguments ← {}
3:      seed_list ← {list of verbs related to accident}
4:      tweets ← tweets from cluster
5:      while tweets ≠ ∅ do
6:          tweet ← pop(tweets)
7:          (verbs, arguments) ← Semantic_Role_Labeling(tweet)
8:          while verbs ≠ do
9:              verb ← pop(verbs)
10:             if verb ∈ seed_list then
11:                 arguments ← arguments ∪ argument1(verb)
12:             else
13:                 similarity_score ← Semantic_Similarity(argument1(verb), arguments)
14:
15:                 if similarity_score ≥ THRESHOLD then
16:                     seed_list ← seed_list ∪ verb
17:                     arguments ← argument ∪ argument1(verb)
18:                 end if
19:             end if
20:         end while
21:     end while
22: end procedure
```

As shown in Algorithm 4, the code "Semantic Similarity (a, list)" gives us the highest similarity with "a" to any item in the list. *THRESHOLD* indicates how much similarity is required to be considered same argument and we set it to 0.8 (1.0 being the exact match).

For example, we use “injury” and “kill” as our seed list of verbs. Let’s assume the following tweets that form a cluster.

```
Man dies, injured on Lagos-Ibadan
Eressway accident - ...

Man killed, nine others injured in
Lagos-Ibadan Eressway accident - The Punch

Man dies, nine injured on
Lagos-Ibadan Eressway accident

Overtaking: Man dies, injured
in Lagos-Ibadan E/Way accident
```

Within this cluster, we find the second tweet includes some verbs from the seed list to indicate how severe the accident is. Using semantic role labeling, we get ARG1, ARG2 associated with each VERB in the sentence. In this example, ARG1 is “man” for the verb “kill”. This indicates that a single person was killed in the accident. We also observe other related tweets with “Man” as an argument for other verbs like “die”. We include the verb “die” in our seed-list of verbs and present this information to the viewer for interpretation.

To identify severity level based on content in the tweet, we considered two classification approaches. One is identifying severe and non severe accidents assuming ML based tweet filtering is active (Section 5.4.2). In another approach, we unify these two steps into one and and build model using BERT and aforementioned techniques to classifiy a tweet to be “non-accident related”, “non-severe” and “severe”. As for features, we use word-to-vec

based vectorization of words in the tweet followed by unweighted averaging to generate fixed length representations, except BERT. It has its own preprocessing technique to generate vector form and we use the default configuration in our experiments. Section 5.7 shows the related results.

### 5.4.7 Visualization and API

For the visualization part, we use Elasticsearch[29] and Kibana to showcase the processed tweets in a structured json-like format. For more general-purpose users, we use a website to show the accident-related tweets along with other processed data. For the collected tweets, we provide a public API where a user can get access all the tweets related to the accidents identified by the system (the active approach) and submit incident information (location and type). The system can then give that incident-specific tweets collected on the fly.

## 5.5 Passive Mode of Accident Report Collection

In previsous sections, we describe the active mode of operation which collects tweets and extracts accident-related information without any input or trigger from other applications. We also provide a passive mode where other systems/applications may trigger the data collection process. For this purpose, we built an API to interface with our system and other applications. The application will provide us with the time, T, and location information, L. We use the time to determine our window of 30 minutes (T, T+30 minutes) and collect tweets published around L within that given window. We run the collection every 2 minutes within that time window and update the results. The result can be accessed with a token. Figure 5.8 shows the information-sharing pipeline.

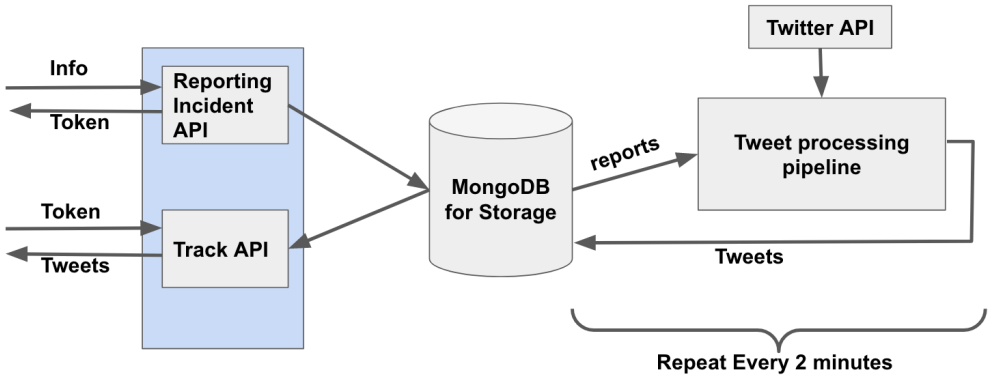



Figure 5.8: Passive mode of accident report identification with the pipeline depicted in Figure 5.2

## 5.6 System Specification and Dataset Description

The pipeline described in Figure 5.2 is written in Python Programming language. Different modules are run as a separate programs and Apache Kafka [93] is used to communicate in between them. We use MongoDB to store the collected data. The system runs in a DELL's RDX740Xd machine with 40 cores and 96GB of RAM.

The data was collected every day from the four cities including Dallas, Austin, Houston, Pittsburgh, and Lagos in Nigeria. We collected tweets for August 2020 and annotated sampled tweets. The data size varies from 3,400 to 17,000 per day for accident-related tweets in United States cities. Dallas had more accident-related tweets than Austin, as there is more traffic on the roads of Dallas than Austin and its neighboring area. However, we collected fewer Tweets in Lagos, Nigeria with around 341 tweets per day on average. In total, we collected on average 42,000 tweets per day that are accident-related for all sample cities. Table 5.2 shows the distribution of the downloaded tweets for each city. To understand how much the collected data were attached to location information, we sampled data for Austin and Dallas cities in the United States. For Austin, we sampled 4,100 tweets, where 900 tweets were accident-related according to the classifier. Among those 900 tweets, 7.8% were

Table 5.2: Citywise Tweet collection statistics for August 2020

| City | Total | Average (Daily) | Max (Daily) | Min (Daily) |
|---|---|---|---|---|
| Dallas | 383,086 | 12,768 | 17,033 | 10,338 |
| Austin | 154,361 | 5,145 | 5,780 | 3,418 |
| Houston | 360,391 | 12,013 | 14,203 | 11,008 |
| Pittsburgh | 359,040 | 11,968 | 14,832 | 9,842 |
| Lagos | 10,230 | 341 | 784 | 127 |

geocoded at the city level and 6.2% were geocoded at street level with respect to the total number of tweets (i.e. 4,100).

For Dallas, we sampled 7,400 tweets where 2,850 were related to the accident. 26.9% of them were geocoded at the city level and 24.2% were correct at the street level.

## 5.7 Experiments and Results

We present different experiment setup and corresponding results in this section.

### 5.7.1 Geolocation

We collected tweets from Lagos, Nigeria, and different cities in the United States. We collected tweets for August 2020 and annotated sampled tweets. From the United States, we considered Dallas, Houston, Austin, Pittsburgh, and New York. For our first and baseline approach for geolocation, we relied on Twitter metadata associated with each of the Tweets namely geo-tags and user location (GT-UL). We also applied semantic role labeling to identify any mention of locations in the tweet (SRL-Tweet). Finally, we used off-the-shelf tools like Google's geocoding software to encode the location mentioned in the tweet (GC). Below, we presented tables focusing on one city at a time and show highlights of different methods in terms of location retrieval. For each table, we presented how many annotated samples we have and how many of them contain address information (city level and street level)[31].

Table 5.3: Location extraction from tweets published within Dallas

| Indentified Location | Step 2: GT-UL | Step 3: SRL-Tweet | Step 4: GC |
|---|---|---|---|
| City | 32.67% | 0.00% | 71.5% |
| Street | 0.00% | 2.75% | 66.9% |
| Geo-Coordinates | 0.00% | 0.00% | 63.4% |

Geo-tag based approach was not showing any result as all of the tweets that we collected and relevant to the accident do not contain any geo-tag information. For that, we removed that column from the following tables.

For Dallas, we annotated 1,137 tweets and the location was identified for 1,089 of them, including 1,076 tweets for which street names were mentioned. Table 5.3 shows what percentages of tweets were georeferenced at each spatial unit in each of the three steps.

As for Dallas and other cities, we derive city information from the user location whereever user mentioned the city in their profile. When we analyze the text, we extract street names in a few cases where the tweet is written close to correct English sentence structure. The semantic parser was able to identify positional tags and dependencies from the text and extract street names. For Dallas and neighboring cities, most of the tweets follow a structured format without having any proper English sentence structure. An example tweet is following -

```
"Motor Vehicle Accident | L B J Acrd N /
Plano Rd | EN29; EN48; RE73; T | 570029 |
20:29 | B | https://t.co/vdTb1ICfa3"
```

For Austin, we have 798 annotated tweets, 243 of them have city names mentioned and 25 of them have street names, as shown in Table 5.4.

In this case, we saw a one percentage increase on text-based location extraction using semantic role labeling. From the collected tweets, we observed more well-structured English

Table 5.4: Location extraction from tweets published within Austin

| Indentified Location | Step 2: GT-UL | Step 3: SRL-Tweet | Step 4: GC |
|---|---|---|---|
| City | 0.00% | 0.00% | 83.3% |
| Street | 0.00% | 16.7% | 66.6% |
| Geo-Coordinates | 0.00% | 0.00% | 33.4% |

Table 5.5: Location extraction from tweets published within Houston

| Indentified Location | Step 2: GT-UL | Step 3: SRL-Tweet | Step 4: GC |
|---|---|---|---|
| City | 13.8% | 0.00% | 97.0% |
| Street | 0.00% | 5.00% | 86.0% |
| Geo-Coordinates | 0.00% | 0.00% | 83.0% |

writings and the semantic parser were able to capture information in more tweets. For example, given the following tweet

```
"This accident cause by speeding between
Austin and Gatineau
https://t.co/Q9QlqzDwFX",
```

the semantic role labeling based approach identified "between Austin and Gatineau" as ARG-LOC for the verb "cause" and and provided more context than simlply identifying two cities as in the case for Google's geo-coding.

For Houston, Texas, we find similar data as Dallas. Of 1081 annotated tweets, 405 have location information and 380 of them have street-name information, as shown in Table V.For Pittsburgh, PA, we have $1,056$ annotated tweets and 764 of them have location information and 657 have street names mentioned. Table 5.5 shows statistics on the dataset.

For Pittsburgh, PA, we have 1056 annotated tweets and 764 of them have location information and 657 have street names mentioned. Table 5.6 shows statistics on the dataset.

Table 5.6: Location extraction from tweets published within Pittsburgh

| Indentified Location | Step 2: GT-UL | Step 3: SRL-Tweet | Step 4: GC |
|---|---|---|---|
| City | 7.00% | 0.00% | 87% |
| Street | 0.00% | 12.00% | 78.67% |
| Geo-Coordinates | 0.00% | 0.00% | 74.0% |

Table 5.7: Location extraction from tweets published within Lagos

| Indentified Location | Step 2: GT-UL | Step 3: SRL-Tweet | Step 4: GC |
|---|---|---|---|
| City | 70.0% | 33.33% | 87.5% |
| Street | 0.00% | 0.00% | 40% |
| Geo-Coordinates | 0.00% | 0.00% | 5.00% |

For Lagos, Nigeria, we observed substantially fewer number of tweets related to accident and annotated only 842 tweets, where 518 of them have location information and 108 have street names. Table 5.7 shows the data on the location extraction methods. For most of the tweets collected from Lagos, the locations were mentioned as the names of the highway. For example, Lagos-Ibadaan Expressway is mentioned in a lot of tweets as accident location. Although, we cannot pinpoint any specific geo-location with it as it is a long highway and nothing was mentioned about a crossing street or exit number, which reduces the number of geo-coordinates found significantly.

### 5.7.2 Severity Detection

There are two types of experiment involved for severity detection.

For identification of severity related information, we considered 568 tweets clustered into 201 groups. We use “injure”, “die” as our seedlist of verbs. Based on human annotation, the final set of verbs should be the following.

```
expire, hurt, cause, injure,
die, kill, take away
```

Table 5.8: Performance of different machine learning models for severe-vs-non-severe classification.

| Classification Model | Accuracy | Precision | Recall | Average Accuracy |
|---|---|---|---|---|
| Naive Bayes | 76.23 | 0.66 | 0.24 | 0.65 |
| SVM | 86.89 | 0.71 | 0.84 | 0.86 |
| HAN | 85.18 | 0.79 | 0.73 | 0.81 |

Table 5.9: Performance of different machine learning models for severe-vs-non-severe classification for Austin area tweets only

| Classification Model | Accuracy | Precision | Recall | Average Accuracy |
|---|---|---|---|---|
| Naive Bayes | 91.6 | 0.3 | 0.6 | 0.77 |
| SVM | 87.03 | 0.26 | 0.57 | 0.73 |
| HAN | 82.40 | 0.22 | 0.71 | 0.76 |

We apply Algorithm 4 which used semantic role labeling based verb-arguments structure to enhance the knowledge-base. We found fewer verbs to assess accident severity than the synonym set of those verbs. With our incremental approach of verb identification and argument extraction, we successfully retrieved all of the verbs and associated arguments. We also observed for all the verbs that the ARG1 has the information for severity.

For identifying the severity level of from the tweet, we conducted two experiments. In the first experiment, we considered only "Severe vs Non-Severe" accident tweet identification. Approximately 850 annotated tweets were collected with 259 of them as "severe". We applied the Hierarchical Attention Network (HAN), Naive Bayes, and SVM based models. Performance data is presented in Table 5.8. We annotated 698 tweets with labels "Severe", "Non- Severe" and "Non-Accident". The distribution of data points for different classes are shown in Table 5.12.

Table 5.9 shows the analysis based on 326 Tweets generated around Austin, with 108 of them used as test instance and rest for training purposes. Only 18 of the given tweets were an indicative of severe accident.

Table 5.10: Performance of different machine learning models for severe-vs-non-severe classification for Dallas area tweets only

| Classification Model | Accuracy | Precision | Recall | Average Accuracy |
|---|---|---|---|---|
| Naive Bayes | 90.32 | 0.42 | 0.67 | 0.79 |
| SVM | 91.61 | 0.4 | 0.6 | 0.77 |
| HAN | 85.8 | 0.20 | 0.63 | 0.75 |

Table 5.11: Performance of different machine learning models for severe-vs-non-severe classification for Lagos area tweets only

| Classification Model | Accuracy | Precision | Recall | Average accuracy |
|---|---|---|---|---|
| Naive Bayes | 76.73 | 0.68 | 0.607 | 0.73 |
| SVM | 82.44 | 0.75 | 0.74 | 0.80 |
| HAN | 84.89 | 0.75 | 0.83 | 0.84 |

Table 5.11 also shows the analysis based on 467 Tweets that were generated around Dallas, with 155 of them used as test instance and the rest for training purposes. Only 12 of the given tweets were an indicative of severe accident.

For Lagos, Nigeria, 741 tweets were generated including 245 as test instance and the rest for training purposes. There were 260 severe-accident related tweets. It is found that the traditional machine learning approach slightly outperforms HAN due to the less amount of training data, although we annotated more data to improve the performance of the HAN-based approach. For all the datasets, we found average accuracy lower than the overall accuracy, indicating that classifiers face difficulty in identifying minority class given the imbalance nature of the dataset itself.

**Unified classifier for filtering and severity detection**

We run the experiment with another classifier which performs the task of identifying whether a tweet is accident-related or not and assessing the severity of the accident. We present the detailed results using BERT based model in Table 5.13. The distribution of instances in different classes are listed in Table 5.12.

Table 5.12: Distribution of 3 severity levels of tweets.

| Severe | Non-Severe | Non-Accident |
|---|---|---|
| 112 | 294 | 292 |

Table 5.13: BERT Performance for Identifying 3-Severity levels

| Instance Class | Precision | Recall | f1-score | support |
|---|---|---|---|---|
| Non-Accident | 0.62 | 0.97 | 0.76 | 99 |
| Non-Severe | 1.00 | 0.70 | 0.82 | 113 |
| Severe | 0.58 | 0.21 | 0.31 | 33 |
| accuracy | | | 0.74 | 245 |
| macro avg | 0.74 | 0.63 | 0.63 | 245 |
| weighted avg | 0.79 | 0.74 | 0.73 | 245 |

We used 245 instances to validate the classifier performance among 698 total instances (35% of the entire dataset).

All classifiers give overall accuracy of 74% but still do not perform well in identifying severe accidents. The confusion matrix is shown in Table 5.14 highlighting different performance for "severe" type classification. Most of the "severe" instances were incorrectly classified as "non-accident-related", which motivates us to further investigate for identifying difference semantically. We provided a detailed result for BERT-based classifier as it outperformed other methods of the classification task. The performance data of Naive Bayes, SVM (Linear Kernel), and HAN based approaches are presented in Table 5.15, 5.16, and 5.17, respectively. We expect higher performance gain for HAN and BERT with a larger dataset compared to traditional ML approaches used here. We found all of the classifier's performance is compromised when identifying a severe accident, which further motivates us to concentrate on how to improve the performance with a more sophisticated representation of the tweet including semantic relationships.

Table 5.14: Confusion matrix in BERT based classification

| | Non-Accident | Non-Severe | Severe |
|---|---|---|---|
| Non-Accident | 96 | 0 | 3 |
| Non-Severe | 32 | 79 | 2 |
| Severe | 26 | 0 | 7 |

Table 5.15: Naive Bayes Performance for Identifying 3-Severity levels

| Instance Class | Precision | Recall | f1-score | support |
|---|---|---|---|---|
| Non-Accident | 0.94 | 0.60 | 0.73 | 99 |
| Non-Severe | 0.60 | 0.97 | 0.74 | 113 |
| Severe | 0.091 | 0.15 | 0.11 | 33 |
| accuracy | | | 0.67 | 245 |
| macro avg | 0.54 | 0.57 | 0.53 | 245 |
| weighted avg | 0.67 | 0.71 | 0.65 | 245 |

Table 5.16: SVM Performance for Identifying 3-Severity levels

| Instance Class | Precision | Recall | f1-score | support |
|---|---|---|---|---|
| Non-Accident | 0.93 | 0.66 | 0.76 | 99 |
| Non-Severe | 0.63 | 0.99 | 0.77 | 113 |
| Severe | 0.15 | 0.24 | 0.19 | 33 |
| accuracy | | | 0.70 | 245 |
| macro avg | 0.58 | 0.62 | 0.57 | 245 |
| weighted avg | 0.69 | 0.74 | 0.68 | 245 |

Table 5.17: HAN Performance for Identifying 3-Severity levels

| Instance Class | Precision | Recall | f1-score | support |
|---|---|---|---|---|
| Non-Accident | 0.91 | 0.64 | 0.75 | 99 |
| Non-Severe | 0.67 | 0.94 | 0.78 | 113 |
| Severe | 0.21 | 0.29 | 0.25 | 33 |
| accuracy | | | 0.71 | 245 |
| macro avg | 0.60 | 0.62 | 0.59 | 245 |
| weighted avg | 0.70 | 0.73 | 0.69 | 245 |

Table 5.18: Police Accident Reports and Tweets compatibility

| | Dallas | Austin | Houston |
|---|---|---|---|
| no. of accident reports | 1371 | 179 | 234 |
| no. of accident reports with tweets | 863 | 83 | 93 |
| percentage | 63% | 47% | 40% |

**Temporal Analysis with Police Accident Reports**

For model validation with the actual police accident reports, 730 police reports in Dallas, 568 reports from Houston and 432 reports from Austin were compared with tweets in a similar timeline that corresponds to any particular accident. The accident information was available for June 2020. We observed when tweets are published at the actual police report and for how many incidents we get tweets associated with it. Table 5.18 shows key values for the experiment results.

An example of police report in JSON is given in Figure 5.9.

```
{
   "date": "2020-06-25T20:29:00.000Z",
   "location": "N JOSEY LN , Lewisville,
               Denton, TX, US",
   "vehicles": [
     "FORD",
     "HONDA"
   ],
   "deaths": 0,
   "injuries": 0
 }
```

Figure 5.9: Information found in a Dallas Area Police Report about an Accident

Based on the date, we create a search window in time to collect the tweets. The tweets are collected based on filtering with the location information.

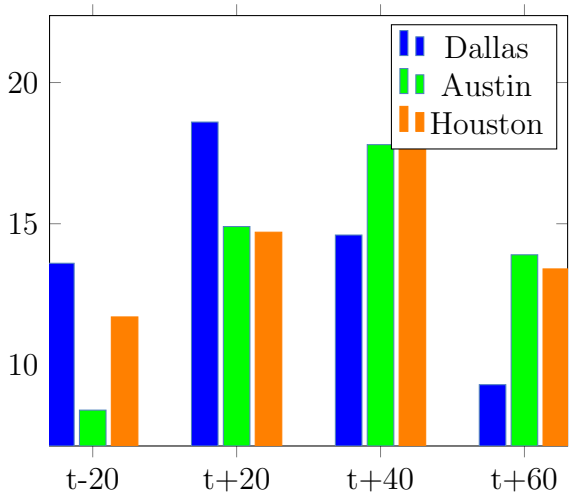



Figure 5.10: Percentage-wise distribution of published tweet w.r.t time windows around the police report time

Figure 5.10 shows a distribution of timeline for published tweets in comparison to the accident report time. Here "t" indicates the time of the accident and the numbers are expressed in minutes. For instance, "t-20" means a time window of 20 minutes prior to the reporting time. The numbers are presented in percentage form. Overall, for Dallas, 58% of the accidents fall in this range since the rest of the tweets have a higher delay of being published in social media. For other cities like Lagos, Nigeria, the average time to address an accident with aid falls between 45 minutes to 1 hour 10 minutes. According to our study for Dallas (assuming similar user behavior over time), we could gather informative tweets within the time-window when the aid is being served.

# CHAPTER 6

# CONCLUSION AND FUTURE WORKS

In this chapter we will discuss our key contributions along with future directions of work related to the previous chapters.

## 6.1 Distributed Framework for Political Event Coding in Real-Time

We describe our framework for real-time automated event coding in a distributed manner for better scalability. We also discuss the data sharing process through a rich query based API serving JSON data. We present a summary of the data to give readers an insight about the gathered dataset. We also discuss some future applications that can be built on top of the framework and the dataset.

In future, we will extend the system to capture other social and political phenomena. Currently we are working on a limited set of Spanish news sources. Once we increase that and add Arabic news sources, we need to study how the system performs in-terms of scalability. We will also work on methods that can be used to semi-automate the ontology extension part. We will also simplify the query mechanism to the API by incorporating more natural way of specifying parameters and query translation.

## 6.2 RePAIR: Recommend Political Actors In Real-time From News Websites

Here, we address the problem of detecting and recommending new political actors and their roles in real-time to code events from news reports. We propose a Spark-based framework called *RePAIR* to recommend new actors with unsupervised ranking techniques and new actor aliases grouping that works on news articles collected on a periodic basis. Moreover, we integrate external knowledge bases (e.g., Wikipedia) to capture the timeline of changes for existing actors and also suggest new political roles for them

Currently, we limit ourselves to finding new political actors but this approach can be extended to recommend new political actions in the CAMEO verb dictionary. In addition, we will extend this to build CAMEO dictionaries for other languages (e.g., Spanish, Arabic).

## 6.3 Automatic Event Coding Framework for Spanish Political News Articles

We describe our infrastructure for real-time automated multilingual event coding in a distributed manner for better scalability. We present some statistics and experimental results to show effectiveness of the individual modules and system as a whole.

In the future, we will extend the system to capture other social and political phenomena. Currently we are working on a limited set of Spanish news sources. Once we increase that and add Arabic news sources, we need to study how the system performs in-terms of scalability. Moreover, we will assess the performance of the UD-PETRARCH event coder in other languages including Arabic, French and Portuguese. We will run our annotation process again for new articles to increase the accuracy of the DNN models used in the chapter.

## 6.4 Exploring the roles of social media data to identify the locations and severity of road traffic accidents

We present a processing pipeline that can capture traffic-accident related information (such as location, severity, etc.) from tweets in real-time. We also discuss how other systems can garner this information. The empirical data analysis and experimental results show promising outcomes and encourage us to further extend and investigate related methods/tools. We find that google geo-coding works reasonably well in identifying location mentioned directly, but Semantic Role Labeling (SRL) extracts locations with context which in some cases provides more accurate information. We also observe good performance on severity detection using SRL and ML based approaches.

In the future, we want to extend the study by including more cities from different countries as the current analysis shows different user behavior and different level of information available city-by-city basis. We will extend the data collection process by dynamically including relevant keywords to be used with api query with twitter. We will annotate more tweets for building a robust dataset for classifiers used in the paper and further improve their performance by tuning the parameters and retraining. We will also train the model with spatial restriction (training only on tweets published within "Dallas") and see how the performance differs across different methods. We will model spatial/temporal distribution of accidents and identify hotspots along those dimensions. We will conduct inter-operability tests with other systems requiring accident-related social media data.

## REFERENCES


[1] *Global Terrorism Dataset*. https://www.start.umd.edu/gtd/.

[2] *Google Knowledge Graph*. https://developers.google.com/knowledge-graph/.

[3] *Google Translation Pricing*. https://cloud.google.com/translate/pricing.

[4] *Newspaper Python Library*. https://github.com/codelucas/newspaper.

[5] *PETRARCH Event Coder*. http://petrarch.readthedocs.org/en/latest/.

[6] *practnlptools 1.0*. https://pypi.python.org/pypi/practnlptools/1.0.

[7] *Real-time Event Data Server*. https://github.com/Sayeedsalam/spec-event-data-server.

[8] *RSS Whitelist*. https://github.com/openeventdata/scraper/blob/master/whitelist_urls.csv.

[9] *Two Ravens*. http://eventdata.2ravens.org/#!/home.

[10] *Web Scraper*. https://github.com/openeventdata/scraper.

[11] Wordnet. https://wordnet.princeton.edu/.

[12] Abrol, S. and L. Khan (2010). Twinner: understanding news queries with geo-content using twitter. In *Proceedings of the 6th Workshop on Geographic information Retrieval*, pp. 1–8.

[13] Adeloye, D., J. Y. Thompson, M. A. Akanbi, D. Azuh, V. Samuel, N. Omoregbe, and C. K. Ayo (2016). The burden of road traffic crashes, injuries and deaths in africa: a systematic review and meta-analysis. *Bulletin of the World Health Organization 94*(7), 510.

[14] Althaus, Scott, J. B. J. F. C. B. P. and D. A. Shalmon (2017). Cline Center Historical Phoenix Event Data. v.1.0.0.

[15] Babko-Malaya, O. (2005). Propbank annotation guidelines.

[16] Baker, C. F., C. J. Fillmore, and J. B. Lowe (1998). The berkeley framenet project. In *17th International Conference on ACL, Volume 1*, pp. 86–90.

[17] Beieler, J. (2016). Generating politically-relevant event data. *arXiv preprint arXiv:1609.06239*.

[18] Beieler, J., P. T. Brandt, A. Halterman, P. A. Schrodt, and E. M. Simpson (2016). Generating political event data in near real time: Opportunities and challenges. *Computational Social Science: Discovery and Prediction. ed. by R. Michael Alvarez, Cambridge, Cambridge University Press*, 98–120.

[19] Bhat, M., M. Qadri, M. K. Noor-ul Asrar Beg, N. Ahanger, and B. Agarwal (2020). Sentiment analysis of social media response on the covid19 outbreak. *Brain, Behavior, and Immunity*.

[20] Boschee, E., J. Lautenschlager, S. O'Brien, S. Shellman, J. Starz, and M. Ward (2018). ICEWS Coded Event Data.

[21] Boschee, E., P. Natarajan, and R. Weischedel (2013). Automatic extraction of events from open source text for predictive forecasting. In *Handbook of Computational Approaches to Counterterrorism*, pp. 51–67. Springer.

[22] Broder, A. Z. (1997). On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pp. 21–29. IEEE.

[23] Chandra, S., L. Khan, and F. B. Muhaya (2011). Estimating twitter user location using social interactions–a content based approach. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pp. 838–843.

[24] Charikar, M., C. Chekuri, T. Feder, and R. Motwani (2004). Incremental clustering and dynamic information retrieval. *SIAM Journal on Computing 33*(6), 1417–1440.

[25] Chen, J. Y., J. Johnson, and G. Yennie. Rnns for stance detection between news articles.

[26] Christakis, N. A. and J. H. Fowler (2010). Social network sensors for early detection of contagious outbreaks. *PloS one 5*(9), e12948.

[27] Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2018a). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[28] Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2018b). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[29] Divya, M. S. and S. K. Goyal (2013). Elasticsearch: An advanced and quick search technique to handle voluminous data. *Compusoft 2*(6), 171.

[30] Doddington, G. R., A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. Strassel, and R. M. Weischedel. The automatic content extraction (ace) program-tasks, data, and evaluation.

[31] Dong, B., J. Guo, Z. Wang, R. Wu, Y. Gao, and L. Khan (2019). Regression prediction for geolocation aware through relative density ratio estimation. In *2019 IEEE International Conference on Big Data (Big Data)*, pp. 1644–1649.

[32] Dwi Prasetyo, N. and C. Hauff (2015). Twitter-based election prediction in the developing world. In *Proceedings of the 26th ACM Conference on Hypertext & Social Media*, pp. 149–158.

[33] D'Ignazio, C., R. Bhargava, E. Zuckerman, and L. Beck (2014). Cliff-clavin: Determining geographic focus for news. *NewsKDD: Data Science for News Publishing, at KDD 2014*.

[34] Earle, P. S., D. C. Bowden, and M. Guy (2012). Twitter earthquake detection: earthquake monitoring in a social world. *Annals of Geophysics 54*(6).

[35] Eck, K. (2012). In data we trust? a comparison of ucdp ged and acled conflict events datasets. *Cooperation and Conflict 47*(1), 124–141.

[36] Gardner, M., J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. F. Liu, M. Peters, M. Schmitz, and L. S. Zettlemoyer (2017). Allennlp: A deep semantic natural language processing platform.

[37] Gaurav, M., A. Srivastava, A. Kumar, and S. Miller (2013). Leveraging candidate popularity on twitter to predict election outcome. In *Proceedings of the 7th workshop on social network mining and analysis*, pp. 1–8.

[38] Georgescu, M., N. Kanhabua, D. Krause, W. Nejdl, and S. Siersdorfer (2013). Extracting event-related information from article updates in wikipedia. In *European Conference on Information Retrieval*, pp. 254–266. Springer.

[39] Gerner, D. J., P. A. Schrodt, and Ö. Yilmaz (2009). Conflict and mediation event observations (CAMEO): An event data framework for a post Cold War world. In J. Bercovitch and S. Gartner (Eds.), *International Conflict Mediation: New Approaches and Findings*, Chapter 13, pp. 287–304. New York: Routledge.

[40] Gerner, D. J., P. A. Schrodt, O. Yilmaz, and R. Abu-Jabr (2002). Conflict and mediation event observations (cameo): A new event data framework for the analysis of foreign policy interactions. *International Studies Association, New Orleans*.

[41] Gildea, D. and D. Jurafsky (2002). Automatic labeling of semantic roles. *Computational linguistics 28*(3), 245–288.

[42] Ginsberg, J., M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, and L. Brilliant (2009). Detecting influenza epidemics using search engine query data. *Nature 457*(7232), 1012–1014.

[43] Gu, Y., Z. S. Qian, and F. Chen (2016). From twitter to detector: Real-time traffic incident detection using social media data. *Transportation research part C: emerging technologies 67*, 321–342.

[44] Halterman, A., J. Irvine, M. Landis, P. Jalla, Y. Liang, C. Grant, and M. Solaimani (2017, Dec). Adaptive scalable pipelines for political event data generation. In *2017 IEEE International Conference on Big Data (Big Data)*, pp. 2879–2883.

[45] Haque, A., L. Khan, M. Baron, B. Thuraisingham, and C. Aggarwal (2016). Efficient handling of concept drift and concept evolution over stream data. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pp. 481–492. IEEE.

[46] Hassan, N., B. Adair, J. T. Hamilton, C. Li, M. Tremayne, J. Yang, and C. Yu (2015). The quest to automate fact-checking. *world*.

[47] Heafield, K. (2011). Kenlm: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pp. 187–197.

[48] Helou, M. A., M. Palmonari, and M. Jarrar (2016). Effectiveness of automatic translations for cross-lingual ontology mapping. *Journal of Artificial Intelligence Research 55*, 165–208.

[49] Inouye, D. and J. K. Kalita (2011). Comparing twitter summarization algorithms for multiple post summaries. In *2011 IEEE Third international conference on privacy, security, risk and trust and 2011 IEEE third international conference on social computing*, pp. 298–306. IEEE.

[50] Khan, L. and F. Luo (2002). Ontology construction for information selection. In *14th IEEE International Conference on Tools with Artificial Intelligence, 2002. (ICTAI 2002). Proceedings.*, pp. 122–127.

[51] Khan, L. and D. McLeod (2000). Effective retrieval of audio information from annotated text using ontologies. In *Proceedings of the International Workshop on Multimedia Data Mining, MDM/KDD'2000, August 20th, 2000, Boston, MA, USA*, pp. 37–45.

[52] Kim, H., V. D'Orazio, P. Brandt, J. Looper, S. Salam, L. Khan, and M. Shoemate (2019). Utdeventdata: An r package to access political event data. *Journal of Open Source Software 4*(36), 1322.

[53] Kreps, J., N. Narkhede, J. Rao, et al. (2011). Kafka: A distributed messaging system for log processing. NetDB (pp. 1-7).

[54] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, Volume 10, pp. 707.

[55] Li, C. and A. Sun (2014). Fine-grained location extraction from tweets with temporal awareness. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pp. 43–52.

[56] Li, R., K. H. Lei, R. Khadiwala, and K. C.-C. Chang (2012). Tedas: A twitter-based event detection and analysis system. In *2012 IEEE 28th International Conference on Data Engineering*, pp. 1273–1276. IEEE.

[57] Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory 28*(2), 129–137.

[58] Lou, H., S. Li, and Y. Zhao (2013). Detecting community structure using label propagation with weighted coherent neighborhood propinquity. *Physica A: Statistical Mechanics and its Applications 392*(14), 3095–3105.

[59] Lu, J. Universal dependency based petrarch, language-agnostic political event coding using universal dependencies.

[60] Manning, C., M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pp. 55–60.

[61] McDonald, R., J. Nivre, Y. Quirmbach-Brundage, Y. Goldberg, D. Das, K. Ganchev, K. Hall, S. Petrov, H. Zhang, O. Täckström, et al. (2013). Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Volume 2, pp. 92–97.

[62] Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119.

[63] Miller, F. P., A. F. Vandome, and J. McBrewster (2009). *Levenshtein Distance: Information Theory, Computer Science, String (Computer Science), String Metric, Damerau?Levenshtein Distance, Spell Checker, Hamming Distance*. Alpha Press.

[64] Müllner, D. (2011). Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*.

[65] Navigli, R. and S. P. Ponzetto (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence 193*, 217–250.

[66] Norris, C., P. Schrodt, and J. Beieler (2017, jan). PETRARCH2: Another event coding program. *The Journal of Open Source Software 2*(9).

[67] O'Connor, B., M. Krieger, and D. Ahn (2010). Tweetmotif: exploratory search and topic summarization for twitter. In *ICWSM*, pp. 384–385.

[68] OEDA. *List of news sources in Spanish*. `https://docs.google.com/spreadsheets/d/13DmJ140wW8pCp6nyRSAk911S7AoF-6zJOJ-F77qoMuM/`.

[69] Open Event Data Alliance. *Web Scraper*. Open Event Data Alliance. `http://oeda-scraper.readthedocs.io/en/latest`.

[70] Osorio, J., V. Pavon, S. Salam, J. Holmes, P. T. Brandt, and L. Khan (2019). Translating cameo verbs for automated coding of event data. *International Interactions 45*(6), 1049–1064.

[71] Palmer, M., D. Gildea, and P. Kingsbury (2005). The proposition bank: An annotated corpus of semantic roles. *Computational linguistics 31*(1), 71–106.

[72] Patty Frontiera (2018). Locating a geocoding tool that works for you and your data. [Online; accessed 30-Octobar-2020].

[73] Ritter, A., O. Etzioni, S. Clark, et al. (2012). Open domain event extraction from twitter. In *18th ACM SIGKDD*, pp. 1104–1112.

[74] Rudra, K., S. Banerjee, N. Ganguly, P. Goyal, M. Imran, and P. Mitra (2016). Summarizing situational tweets in crisis scenario. In *Proceedings of the 27th ACM Conference on Hypertext and Social Media*, pp. 137–147.

[75] Sankaranarayanan, J., H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling (2009). Twitterstand: news in tweets. In *Proceedings of the 17th acm sigspatial international conference on advances in geographic information systems*, pp. 42–51.

[76] Saraf, P. and N. Ramakrishnan (2016). EMBERS autogsr: Automated coding of civil unrest events. In *22nd ACM SIGKDD*, pp. 599–608.

[77] Schrodt, P. A. (2012a). Cameo: Conflict and mediation event observations event and actor codebook. *Pennsylvania State University*.

[78] Schrodt, P. A. (2012b). Precedents, Progress, and Prospects in Political Event Data Article in International Interactions.

[79] Schrodt, P. A. and D. J. Gerner (2004). An event data analysis of third-party mediation in the middle east and balkans. *Journal of Conflict Resolution 48*(3), 310–330.

[80] Schrodt, P. A. and D. Van Brackle (2013). Automated coding of political event data. In *Handbook of Computational Approaches to Counterterrorism*, pp. 23–49. Springer.

[81] Sharifi, B., M.-A. Hutton, and J. Kalita (2010a). Summarizing microblogs automatically. In *Human language technologies: The 2010 annual conference of the north american chapter of the association for computational linguistics*, pp. 685–688.

[82] Sharifi, B., M.-A. Hutton, and J. K. Kalita (2010b). Experiments in microblog summarization. In *2010 IEEE Second International Conference on Social Computing*, pp. 49–56. IEEE.

[83] Solaimani, M., R. Gopalan, L. Khan, P. T. Brandt, and B. Thuraisingham (2016a). Spark-based political event coding. In *2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService)*, pp. 14–23.

[84] Solaimani, M., R. Gopalan, L. Khan, P. T. Brandt, and B. Thuraisingham (2016b). Spark-based political event coding. In *BigDataService*, pp. 14–23. IEEE.

[85] Solaimani, M., S. Salam, L. Khan, P. T. Brandt, and V. D'Orazio (2017a). *APART: Automatic Political Actor Recommendation in Real-time*, pp. 342–348. Cham: Springer International Publishing.

[86] Solaimani, M., S. Salam, L. Khan, P. T. Brandt, and V. D'Orazio (2017b, Dec). Repair: Recommend political actors in real-time from news websites. In *2017 IEEE International Conference on Big Data (Big Data)*, pp. 1333–1340.

[87] Stewart, C. A., T. M. Cockerill, I. T. Foster, D. Y. Hancock, N. Merchant, E. Skidmore, D. Stanzione, J. Taylor, S. Tuecke, G. W. Turner, et al. (2015). Jetstream: a self-provisioned, scalable science and engineering cloud environment. In *XSEDE*, pp. 29–1.

[88] Straka, M. and J. Straková (2017, August). Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. pp. 88–99.

[89] Studer, M. and G. Ritschard (2016). What matters in differences between life trajectories: A comparative review of sequence dissimilarity measures. *Journal of the Royal Statistical Society: Series A (Statistics in Society) 179*(2), 481–511.

[90] Thapa, L. (2016). Spatial-temporal analysis of social media data related to nepal earthquake 2015. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 41*, 567.

[91] The University of Pennsylvania. *The Penn Treebank Project*. The University of Pennsylvania. `https://www.cis.upenn.edu/~treebank/`.

[92] Thein, K. M. M. (2014a). Apache kafka: Next generation distributed messaging system. *International Journal of Scientific Engineering and Technology Research 3*(47), 9478–9483.

[93] Thein, K. M. M. (2014b). Apache kafka: Next generation distributed messaging system. *International Journal of Scientific Engineering and Technology Research 3*(47), 9478–9483.

[94] Timothy, D., T. Allison, S. Blair-goldensohn, J. Blitzer, A. Elebi, S. Dimitrov, E. Drabek, A. Hakim, W. Lam, D. Liu, et al. (2004). Mead a platform for multidocument multilingual text summarization. In *International Conference on Language Resources and Evaluation*, pp. 699–702.

[95] Vanderwende, L., H. Suzuki, C. Brockett, and A. Nenkova (2007). Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing & Management 43*(6), 1606–1618.

[96] Wang, L., L. Liu, and L. Khan (2004). Automatic image annotation and retrieval using subspace clustering algorithm. In *Proceedings of the 2nd ACM international workshop on Multimedia databases*, pp. 100–108.

[97] Weninger, T., R. Palacios, V. Crescenzi, T. Gottron, and P. Merialdo (2016). Web content extraction: a metaanalysis of its past and thoughts on its future. *ACM SIGKDD Explorations Newsletter 17*(2), 17–23.

[98] Wikipedia contributors (2019). Elbow method (clustering) — Wikipedia, the free encyclopedia. [Online; accessed 20-August-2019].

[99] Wikipedia Contributors (2019). Twitter — Wikipedia,, the free encyclopedia. [Online; accessed 27-Octobar-2020].

[100] Yen, I.-L., J. Goluguri, F. Bastani, L. Khan, and J. Linn (2002). A component-based approach for embedded software development. In *Proceedings Fifth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing. ISIRC 2002*, pp. 402–410. IEEE.

[101] Zaharia, M., M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. *9th USENIX*, 2–2.

[102] Zaharia, M., R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, et al. (2016). Apache spark: a unified engine for big data processing. *Communications of the ACM 59*(11), 56–65.

[103] Zhang, S., J. Tang, H. Wang, and Y. Wang (2015). Enhancing traffic incident detection by using spatial point pattern analysis on social media. *Transportation Research Record 2528*(1), 69–77.

## BIOGRAPHICAL SKETCH

Sayeed Salam obtained his BSc in Computer Science and Engineering from Bangladesh University of Engineering and Technology. He is a PhD student in The Department of Computer Science at The University of Texas at Dallas. He has been working in the Big Data Analytics and Management Lab as a research assistant starting Spring 2016. His research interest includes large scale distributed system design, text processing and analytics with application towards Political Science and Healthcare. He worked on the Multilingual Framework development for identifying political events in different languages other than English. He also worked on the ontology extension for Conflicts and Event Observation and Mediation (CAMEO) ontology used for coding political events. Additionally, he works on geolocating and severity analysis of road-traffic incidents based on tweets. He has been working in projects funded by NSF and NIH and in collaboration with Lagos Area EMS service from Nigeria.

**CURRICULUM VITAE**

# Sayeed Salam

November 11, 2021

## Contact Information:

Department of Computer Science
The University of Texas at Dallas
800 W. Campbell Rd.
Richardson, TX 75080-3021, U.S.A.

Email: sxs149331@utdallas.edu

## Educational History:

BSc, Computer Science and Engineering, Bangladesh University of Engineering and Technology, 2013
PhD, Computer Science, The University of Texas at Dallas, 2021

*A Big Data Framework for Unstructured Text Processing with Applications towards Political Science and Healthcare*
PhD Dissertation
Computer Science Department, The University of Texas at Dallas
Advisors: Dr. Latifur Khan

## Employment History:

Research Assistant, The University of Texas at Dallas, January 2016 – present
Teaching Assistant, The University of Texas at Dallas, August 2014 – December 2015
Lecturer, BRAC University, Dhaka, Bangladesh, January 2014 – July 2014
Software Engineer, ReveSystems Ltd., Dhaka, Bangladesh, March 2013 – October 2013

## Publications:

Exploring the roles of social media data to identify the locations and severity of road traffic accidents, IEEE AIKE, 2021
Automatic Event Coding Framework for Spanish Political News Articles., IEEE IDS 2020
UTDEventData: An R package to access political event data, JOSS 2019
Translating CAMEO Verbs for Automated Coding of Event Data, International Interactions (II): Empirical and Theoretical Research in International Relations, 2019
Automated Verbal-Pattern Extraction from Political News Articles using CAMEO Event Coding Ontology., IEEE IDS 2019
Distributed Framework for Political Event Coding in Real-Time., IEEE EECS 2018
RePAIR: Recommend political actors in real-time from news websites., IEEE BigData, 2017

Near real-Time atrocity event coding. IEEE ISI 2016

## Professional Memberships:

Institute of Electrical and Electronics Engineers (IEEE), 2017–present