GDELT GA Search Users Manual

User's Guide and Manual

November 22, 2021

Contents

1 Introduction **1**
- 1.1 Finding Relevance: A GA Approach . . . 2
- 1.2 Acknowledgements . . . 4

2 Getting Started: Running GDELT_GA_Search Searches **5**
- 2.1 Input Data . . . 5
  - 2.1.1 GDELT Data . . . 5
  - 2.1.2 Time Series Comparative Data . . . 6
- 2.2 Running Single Runs . . . 6
  - 2.2.1 The GDELT properties file . . . 6
  - 2.2.2 Using the Runner Class from the Command Line . . . 9
- 2.3 Running Parallelized Suites of Runs . . . 10
  - 2.3.1 The Run Group Specification File . . . 10
  - 2.3.2 Running The ParallelMultiRunner Class from the Command Line . . . 11

3 The Structure of GDELT Data **13**
- 3.1 The GDELT Events Table . . . 13
- 3.2 Metadata . . . 14
- 3.3 Date Information . . . 15
- 3.4 Characterization . . . 15
- 3.5 Location . . . 16
  - 3.5.1 Location Field Descriptions . . . 17
- 3.6 Actor Information . . . 18
- 3.7 Discourse . . . 19
- 3.8 The GDELT Mentions Table . . . 20
  - 3.8.1 Metadata . . . 20
  - 3.8.2 Source Information . . . 21
  - 3.8.3 In-Document Information . . . 21

4 Data Types and Domains in GDELT_GA_Search **23**

4.1 Theory: Fundamentals of Data Types . . . 23
4.1.1 Bounded Data . . . 24
4.1.2 Unbounded Data . . . 24
4.2 Functional Requirement of a Data Type in GDELT_GA_Search . . . 25
4.3 Data Types and Data Fundamental Domains in GDELT_GA_Search Code . . . 25
4.4 Aggregated Data Domains . . . 26
4.5 GDELT Fields and their Domains in GDELT_GA_Search . . . 27

5 Queries **29**
5.1 Query Matching . . . 29
5.2 Time Shifting . . . 30
5.3 Nested Queries . . . 30
5.4 Mutation . . . 31
5.5 Specifying query behaviors for specific runs . . . 31

6 Prediction Methods **33**
6.1 Definitions . . . 33
6.2 Prediction Methods . . . 35
6.2.1 Raw . . . 35
6.2.2 Base Values . . . 35
6.3 Replay . . . 35
6.3.1 Linear Regression . . . 36
6.4 Long Short-Term Memory . . . 36
6.5 Elastic Net . . . 36
6.6 Gradient Boost . . . 36
6.7 Lasso . . . 36
6.8 Adding Prediction Methods . . . 36
6.8.1 Java . . . 36
6.8.2 Python . . . 36

7 Scoring Methods **37**
7.1 Root Mean Square Error (RMSE) . . . 38
7.2 Normalized Root Mean Square Error (NRMSE) . . . 38
7.2.1 Mean . . . 38
7.2.2 MinMax . . . 38
7.2.3 Standard Deviation . . . 38
7.3 Correlation (Positive Only) . . . 38
7.4 Input Correlation . . . 38
7.4.1 Trimmed . . . 38
7.4.2 Positive Only . . . 38
7.5 Adding Scoring Methods . . . 38

7.5.1 Python 38
7.5.2 Java 38

8 Downloading GDELT Data **39**
8.1 Using the Events and Mentions Scrapers 39
8.2 Setting download parameters 40
8.3 Filtering Events 40

Chapter 1

Introduction

GDELT is the Global Database of Events, Language, and Tone, a record of real-world events harvested automatically from world news sources. Its records begin from 1979 and continue to the present, and the most recent versions of the GDELT data set are updated every 15 minutes. The records are compiled using natural language processing (NLP), and the record for each event includes not only information about the actors and actions, but also about aspects of how the action is described and discussed in the news: how widely, for how long, in what tone, etc. Consequently, GDELT provides an incomparable record of how events in the real world are being considered and expressed by the general public.

At a micro-level, a record in the GDELT data set is able to shed light on how a specific event was considered. For example, the GDELT data set shows whether an event was discussed widely or only briefly, whether it was discussed while it ocurred or only afterwards, whether it was debated or celebrated, or whether it was memorialized and recalled long after it first took place. At a macro-level, the complete record of GDELT events forms a richer data source, one that can inform about long-term dynamics and ongoing world historical processes, of which individual events are only single lines in a longer play.

A difficulty, however, is that the complete GDELT data set covers the whole world: it is more than just one play; it is a whole library. The challenge is often to find the correct level at which to understand the story that GDELT is telling. More formally, one might wish to isolate specific subsets of the GDELT data that apply to an area of interest and to leave out all of those that do not.

GDELT is integrated with a query search that can extract subsets based on the data contained in GDELT's records. For example, if one were interested in diplomacy between France and Russia in 2018, one could perform a search in which those two countries are

listed as 'actors' and the time range was limited to the period of interest of 2018. Other filters (for example, actions of specific kinds) could also be applied. However, for some purposes, it may be difficult to guess at what is the proper query because false positives and false negatives can dominate, washing out the signal.

1.1 Finding Relevance: A GA Approach

The GDELT_GA_Search toolkit assumes that there can be valuable relationships between subsets of GDELT data and other time-series data found in the real world. The canonical example is that bursts of activity in global news reporting concerning some ongoing process (e.g., a world conflict) will be echoed by similar activity on social media platforms. Because of this relationship between news and social media, time-series showing activity on these social media platforms should be able to be related to a time-series showing discussions in world news events as catalogued by GDELT.

An additional expectation is that these relationships will only be visible on specific subsets of the GDELT data. The global number of news articles collected by GDELT and the number of events catalogued will remain more or less constant, but the content will vary through time. The subset of content that is of interest will be unified by some common element such as a specific term, a set of terms, or even other criteria. The challenge is to find the criteria that, when issued as a query against the full GDELT data set, returns a set that has a signature related to the long-term issue being considered.

For example, the four graphs in figure 1.1 show time series data with time on the x-axis. The upper graph represents the activity on some social media platform: activity is low early in the period shown, then there is a burst followed by heightened activity, and there is a drawdown through the remainder of the period.

The second graph from the top shows the total GDELT events. Note that this graph would already have been subselected using keywords. (Although the GDELT_GA_Search toolkit can work with the entire GDELT data set, upstream preprocessing like this is common to winnow the GDELT data to only elements likely to be relevant.) The key aspect here is the alignment with the upper graph. Notice that there is a sharp burst and then a drawdown that shows a strong correlation between the graphs. Even so, the alignment is far from exact. (Note also that scales would be omitted from the axes. It is assumed that the x-axes are the same, but this is not true for the y-axis: the number of events in the upper graph may be in the millions while the number in the lower graph is in the thousands. What matters is their correlation.)

The ALL GDELT W/SUBSETS ROOT CODE graph (c) shows the same graph as (b); however, where (b) showed only the total, (c) subdivides the GDELT data—in this case,

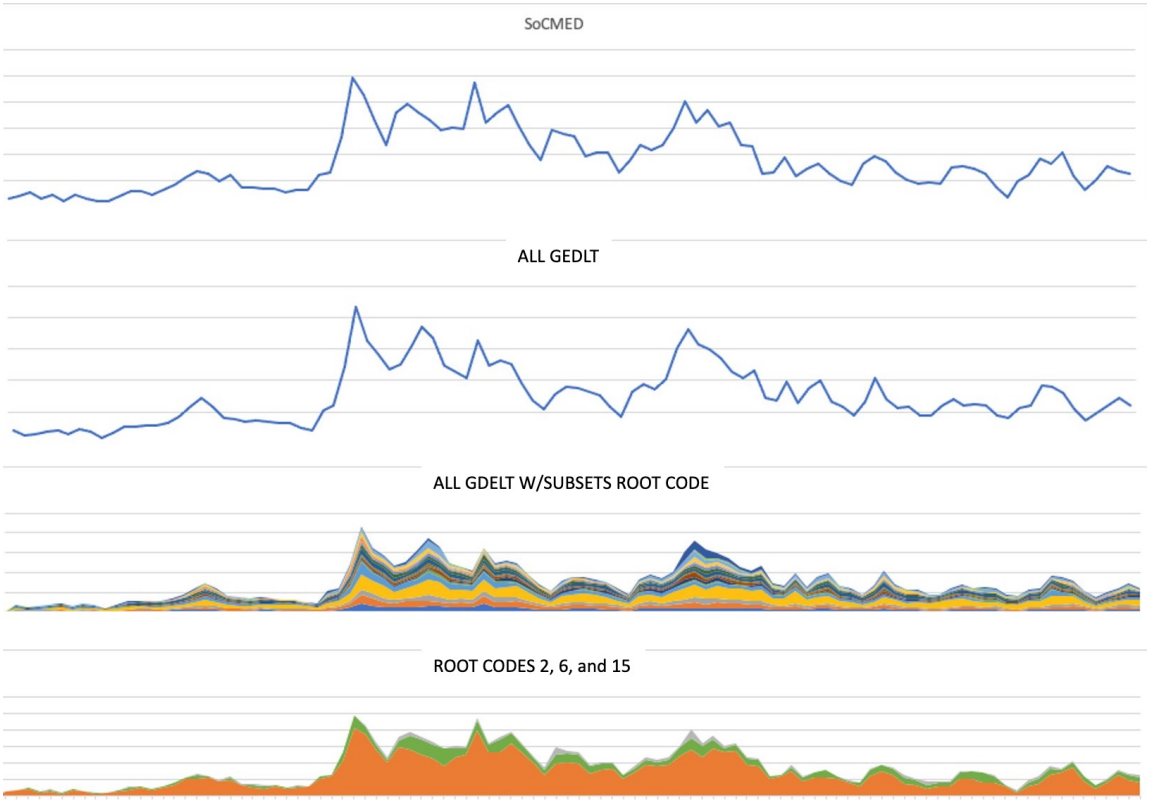

Figure 1.1: Conceptual view of the search process

by the value for *root code* (see below). There are 20 possible values for this field, and hence 20 bands in the graph. Each band represents the counts of GDELT events with the specified root code.

The lowest graph (d) shows the results when only three root codes (2, 6, and 15) are included in the GDELT totals. Whereas the graph of all the GDELT events matches the social media event graph at a general level, the graph of these three specific root codes matches much more closely.

This offers a nice example the GDELT_GA_Search toolkit because it is easily bounded: with 20 root codes, there are exactly $2^n - 1$ combinations possible. For 20 root codes this is 1,048,575 combinations. This would be manageable with a brute-force approach—it is not difficult to try 1 million combinations—but this number grows quickly as more possible criteria are added. It is easy to reach a space that is too large to explore fully, so a different approach is needed instead.

The GDELT_GA_Search tool searches through this space by building GDELT time series using *queries*. A query is a set of criteria that serve to classify the GDELT data and extract matching events. The crux of the GDELT_GA_Search algorithm is that these queries can be changed in an iterative way that moves through the search space. The algorithm is this:

- **Construct a query framework** that extracts subsets of GDELT data based on an array of possible fields
- **Score the results** based on the information they provide about the social media time series
- **Mutate the best-scoring variations** to explore whether expanding or reducing the criteria improve the scores

1.2 Acknowledgements

Chapter 2

Getting Started: Running GDELT_GA_Search Searches

This chapter provides a basic introduction to running the simulation

2.1 Input Data

The input data for GDELT_GA_Search searches consists of four kinds of files:

- GDELT Data, including:
  - Events Data
  - Mentions Data (can be omitted)
- Time Series Comparative Data
- The GDELT Properties File
- Run Group Specification File (for sets of parallelized runs)

The GDELT Properties File and the Run Group Specification File are used with single runs and with parallelized runs, which are both discussed in subsequent sections

2.1.1 GDELT Data

GDELT data consists of *events* and *mentions*. Both of these are given in CSV files downloadable from GDELT's archives. This toolkit provides a pair of tools that can automat-

ically download events and related mentions from within a given time frame. (Note that mentions may accumulate after the initial event, and thus some events' mentions may fall outside the original time limit for events). An alternative data structure in JSON, from the original project for which this tool was designed, is also accommodated but is not documented.

2.1.2 Time Series Comparative Data

The search will compare the GDELT data with a time series; the input data for the time series has the following structure (for more information, see **??**).

```
platform,frame,ConvertedDate,counts
twitter,capitol,2020-10-01,1034
twitter,capitol,2020-10-02,1081
twitter,capitol,2020-10-03,1096
twitter,capitol,2020-10-04,1116
twitter,capitol,2020-10-05,1198
twitter,capitol,2020-10-06,1234
twitter,capitol,2020-10-07,1271
twitter,capitol,2020-10-08,1368
etc.
```

2.2 Running Single Runs

2.2.1 The GDELT properties file

The GDELT properties file describes the location of the GDELT data, its chronological boundaries, the parameters of the GDELT queries (including their mutation properties), the location of the social media data, and some other properties of exploration. The structure of the file is simple: each line can contain one property specification, with a non-whitespaced property name on the left, an equals sign, and the property value on the right. Lines that start with number signs are ignored and can be used as comments.

Here is an example of a file

```
# GIT DIRECTORY
gitDir = /home/username/gdelt_ga_search

# GDELT DATA FILE LOCATION AND INFO
dataLocation      = ./data/examples/GDELT_Events_Data.csv
```

```
mentionsLocation = ./data/examples/GDELT_Mentions_Data.csv
beginHistory     = 2020-01-31
dayZero          = 2021-01-01

# GDELT QUERY SPECIFICATIONS
MaxShift                = 24
MinShift                = 24
MaxVizOffset            = 0
MinVizOffset            = 0
Actor1CountryCodes      = null,KEN,ZWE,UGA,AFR
Actor2CountryCodes      = null,KEN,ZWE,UGA,AFR
MatchGoldsteinScale     = FALSE
MatchAverageTone        = FALSE
MatchRootCode           = FALSE
MatchLatLon             = FALSE

# MUTATION WEIGHTS
Actor1CountryCodesMutateWeight = 10
Actor2CountryCodesMutateWeight = 10

# TIME SHIFT MUTATION PROBABILITY
MutatingTimeshiftProb          =  0

# TIME SERIES DATA FILE LOCATION AND INFO
timeSeriesDataLocation = ./examples/DummySocialMediaCounts.csv
timeSeriesDataStart    = 2020-10-01T00:00:00

# BAILOUT
bailoutInterval  = 3600
bailoutValue     = 100
```

- *gitDir* This specifies the directory of the GDELT code. In cases where the code may be modified, it can be useful to employ a mechanism that will stop a run if the code has not been committed.
- *dataLocation* The location of the GDELT events data file.
- *mentionsLocation* The location of the GDELT mentions data file. This can be omitted if no mentions are available
- *beginHistory* The earliest date in the GDELT data
- *dayZero* The date that will be considered the boundary between training and predic-

tion

- *MaxShift* Maximum number of hours the GDELT search will 'shift' times. This can be used so that predictions are not (or, occasionally, are) based on information from the future, e.g. a 24-hour shift means the predictions are based on the day prior.
- *MinShift* Minimum number of hours to shift. If this value is the same as the MaxShift, the shift will be constant
- *MaxVizOffset* Maximum visibility offset, or the number of hours offset the visibility of events will be
- *MinVizOffset* Minimum visibility offset
- *Actor1CountryCodes* List of country codes that are used for Actor 1 Country Code queries
- *Actor2CountryCodes* List of country codes that are used for Actor 2 Country Code queries
- *MatchGoldsteinScale* Whether to consider Goldstein Scale when matching queries
- *MatchAverageTone* Whether to consider Average Tone when matching queries
- *MatchRootCode* Whether to consider Root Code when matching queries
- *MatchLatLon* Whether to consider Latitude and Longitude when matching queries
- *Actor1CountryCodesMutateWeight* Weighted probability that the Actor 1 Country Code element of a query will mutate
- *Actor2CountryCodesMutateWeight* Weighted probability that the Actor 2 Country Code element of a query will mutate
- *MutatingTimeshiftProb* Probability that the shift value will mutate
- *timeSeriesDataLocation* File location for the social media counts
- *timeSeriesDataStart* Earliest date in the social media files.
- *bailoutInterval* Time (in seconds) after which the routine will stop a search if no improvement has been found
- *bailoutValue* Number of iterations after which the routine will stop a search is no improvement has been found

2.2.2 Using the Runner Class from the Command Line

It is possible to use the Runner class to run single runs of the search. This requires a collection of arguments. A typical command line will include arguments following this example:

- */usr/lib/jvm/java-8-openjdk-amd64/bin/java* is the path to the Java RunTime
- *-DGDELT_GAS_PSCRIPT_PATH="/home/gdelt_ga_search/pscripts/"* is the path to the Python Scripts. This is required even if Python Scripts are not used
- *-Dfile.encoding=UTF-8* A specification of file encoding
- *-classpath ./bin:./lib/gson-2.8.5.jar:./lib/jeromq-0.4.0.jar:./lib/commons-math3-3.6.1.jar gdelt.runners.Runner* The classpath
- *RUNNER_0* An identifier that is used for some interim files
- *./examples/SampleConfig.props* The properties file in which the characteristics of the input data and queries are specified
- *LINEAR_REGRESSION* The type of prediction to be used
- *NRMSE_MEAN* The type of scoring to be used
- *2021-01-01* The date to be used as day zero
- *28* The duration of the training period
- *21* The number of days forward to predict for the final prediction
- *21* The number of days to predict for the training periods
- *50* The population size
- *10* The number of survivors
- *15* The number of generations to run if no stopping condition is encountered
- *7* The number of days prior to day zero minus the test prediction period to take as the training period day zero
- *twitter_capitol* the data source ('platform') and topic ('frame') to be used; must match the input data given in the configuration properties file named above
- *DEMO_001_000000::NewQueries* An alphanumeric identifier and a description; the description can use underscores as spaces. The description will be written with every output file.

2.3 Running Parallelized Suites of Runs

2.3.1 The Run Group Specification File

The Run Group Specification File contains the information needed to structure a collection of parallelized runs. The format is a property on the left, a pair of colons, and a value on the right. The term on the right may include several different values. The configuration will create a set of runs based on all of the possible combinations of values.

An example listing of a Run Group Specification File (from ./examples/SampleConfig.props).

```
predMethods::LINEAR_REGRESSION
scoreMethods::NRMSE_MEAN
zeroTimes::2021-01-01
trainDurations::28
predDurations::21
predDurationTrains::21
popSizes::50
survivorValues::10
generations::15
predictionPoints::7,0
platforms::twitter,youtube
frames::capitol,insurrection
```

The meanings of these are:

- *predMethods* The method to be used for prediction; valid values are the names of the values in the PREDICTION_METHOD enum of the Predictor class.
- *scoreMethods* The method of scoring to be used; value values are the names of the values in the SCORE_METHOD enum of the Scorer class.
- *zeroTimes* Times that are to be used as the boundary between training and prediction
- *trainDurations* Number of days that are used for training periods
- *predDurations* Number of days forward (ahead of zero time) that is considered prediction
- *predDurationTrains* Number of days forward used by the training samples.
- *popSizes* Size of the GA population
- *survivorValues* Number of best-scoring survivors in the population

- *generations* Number of generations the search will run if it does not hit a stopping condition first
- *predictionPoints* Days prior to zero time *minus the predDurationTrains* value that will mark the train/test boundary for the predictions.
- *platforms* Social media platforms to be explored
- *frames* Frames (topics) to be explored

In the above example, eight combinations would be created based on the fact that multiple values for prediction points, platforms, and frames are combined.

2.3.2 Running The ParallelMultiRunner Class from the Command Line

When running the ParallelMultiRunner from the command line, the arguments should be

- The full path to your Java runtime, e.g. /usr/lib/jvm/java-8-openjdk-amd64/bin/java
- The relative path to your GDELT Properties file, e.g. ./examples/SampleConfig.props
- A group name for this set of runs, e.g. DEMO_001
- A description for this group of runs, surrounded by quotation marks, e.g. "Queries with Actor 1 and Actor 2 fewer test periods, longer training periods, larger population, fewer survivors"
- The number of repetitions to run, i.e. the number of duplicate runs for every combination in the Run Group Configuration File (e.g., 2)
- The number of processes available for parallelization (e.g. 2)
- The relative path to the Run Group Configuration file, e.g. examples/Example_Spec.rgspec

Chapter 3

The Structure of GDELT Data

The GDELT data set consists of two tables: an **events table** that is a list of real-world events that GDELT has extracted from news sources and a **mentions table** that lists the specific times that this event has been discussed in world news sources. For example, "Country X condemns the human rights abuses in Country Y" might appear as an entry in the events table, and several news articles mentioning it would appear in the mentions table (e.g., articles appearing on or in the BBC, CNN, NY Times services). The two tables are in a one-to-many relationship, meaning that every entry in the Events table can have multiple entries in the Mentions table. Because the events are drawn from mentions—that is, GDELT discovered every event in at least one data source—every event in the events table should have at least one entry in the mentions table.

3.1 The GDELT Events Table

This section reviews the fields in the GDELT events table. To simplify the discussion, some fields are considered as families: all *1Actor1* variables and *1Actor2* variables are specified identically, so both Actor 1 and Actor 2 are characterized the same way. For example, *Actor1CountryCode* and *Actor2CountryCode* are the same kind of variable and are represented in the same way. In this discussion here, *ActorX* (e.g., *ActorXCountryCode*) is used to refer to both Actor1 and Actor2.

Fields in the events can be grouped into several categories:

- **Metadata** are fields that are used primarily for recordkeeping (e.g., GDELT's distribution filenames) and do not generally contain information about the real-world event being recorded or the discourse about it.

- **Date Information** contains fields that characterize the real-world action that was performed in the event (e.g., level of conflict or specific kind of interaction that was undertaken).
- **Characterization** contains fields that describe the character, or attributes, of the action performed in the event.
- **Location Information** contains information about the geographic position of the event; note that it can include the location of the event and (as separate values) the locations of the actors when they performed the event.
- **Actor Information** contains information about the actors (at least one, possibly two) involved in event.
- **Discourse** contains fields that indicate elements of the way the event was discussed (e.g., the average 'tone' of the articles that mention this event).

These categories, and the fields within them, are shown in Table 3.1.

Table 3.1: Fields in GDELT Events Table

Metadata	Date Information	Characterization	Location		Actor Information	Discourse
globaleventid	day	QuadClass	ActionGeo_FullName	ActorXGeo_FullName	ActorXName	IsRootEvent
filename	MonthYear	EventRootCode	ActionGeo_Type	ActorXGeo_Type	ActorXCode	NumMentions
sourceurl	Year	EventBaseCode	ActionGeo_CountryCode	ActorXCountryCode	ActorXCountryCode	NumSources
dateadded	FractionDate	EventCode	ActionGeo_ADM1Code	ActorXGeo_ADM1Code	ActorXKnownGroupCode	NumArticles
		GoldsteinScale	ActionGeo_ADM2Code	ActorXGeo_ADM2Code	ActorXEthnicCode	AvgTone
			ActionGeo_Lat	ActorXGeo_Lat	ActorXReligionCode	
			ActionGeo_Long	ActorXGeo_Long	ActorXTypeNCode	
			ActionGeo_FeatureID	ActorXGeo_FeatureID		

3.2 Metadata

Metadata includes information that is used to track the source of the information and to indicate other aspects that have to do with the piece of information as a GDELT record (as opposed to information about the real-world event). Much of this information is not directly related to the real-world event itself.

- **globaleventid** - This is a unique identifier that is used to specify this event (and link it to the related records in the mentions table). These IDs are assigned sequentially (and hence could be used as range limiters or for comparison) but should be treated as arbitrary.
- **filename** - This refers to the file in which GDELT stores this record (generally named after the 15-minute interval in which the record was collected)

- **sourceurl** - This is the URL of the **first** source in which GDELT identified this real-world event. It is not necessarily the earliest source in which it appeared in the real world, nor is it the most important source. While it can be representative, it i also somewhat arbitrary.
- **dateadded** - This is the date that the record was added to GDELT, including a time stamp with 15-minute resolution. Note that this is **not** the date of the event, which could be much earlier (even years).

3.3 Date Information

Date information is related to the date that the real-world event took place. (This is in contrast to *dateadded*, which is when GDELT discovered the event in the news sources.)

- **day** - This gives the date the event took place in YYYYMMDD format.
- **MonthYear** - This excludes the day and presents the event date in truncated YYYYDD format.
- **Year** - This gives only the year in YYYY format.
- **FractionDate** - This gives the percentage of the year completed as YYYY.FFFF, essentially collapsing the month and day fields.

The date information includes redundancy: If *day* information is available, then all other date information codes can be derived.

3.4 Characterization

GDELT provides a set of fields that describe the character of the action performed. This is done by assigning the event to a category based on the which lists an extensive taxonomy of events (e.g. 'Reach Accord' or 'Issue Statement'). Fundamentally, this is *one* assignment: GDELT selects a single CAMEO code that it believes represents the kind of event performed. In the dataset, however, GDELT provides several versions of the CAMEO code assigned. The different versions of the CAMEO code reflect different levels of specificity, ranging from general to more specific and following the structure described in the CAMEO code book. These versions are:

- **QuadClass** - This field specifies the primary classification of an event type to allow for analysis at the highest level of aggregation. Events are organized into four classifications, each with a numeric code: 1 is Verbal Cooperation, 2 is Material Cooperation, 3 is Verbal Conflict, and 4 is Material Conflict.

- **EventRootCode** - Sometimes referred to as *RootCode*, this is the first part of the CAMEO code (the first two digits, possibly including a leading zero), representing a general category of the action such as 'Express Intent to Cooperate'.
- **EventBaseCode** - Sometimes referred to as the *BaseCode*, this is both the first and second parts of the CAMEO code, representing a more specific action, such as 'Express Intent to Engage in Material Cooperation'.
- **EventCode** - This is the full CAMEO code assigned, fully categorizing the action, e.g. 'Express Intent to Cooperate Militarily'.

In the example above, the *QuadClass* would be 1 (verbal cooperation), the *RootCode* would be '03,' the *BaseCode* would be '031,' and the full code, *EventCode*, would be '0312.' Note that some events are only coded to the *BaseCode* level of specificity. '031' is a valid code and represents 'Express Intent to Engage in Material Cooperation' without specifying the form of that cooperation. Similarly, '030' is a valid code that is effectively just the *RootCode*, the trailing zero indicating that the action could not be classified with more specificity.

GDELT provides an additional field:

- **GoldsteinScale** - This is an event's likely impact on the stability of a region on a scale from -10 (highly destabilizing) to +10 (stabilizing).

The *GoldsteinScale* score is assigned based on the event *type* and not for real-world characteristics of the event itself. For example, if they were both assigned the same CAMEO code, a riot with 10 participants would be assigned the same Goldstein Scale value as a riot with 10,000 participants.

NOTE: Although there are five distinct values here (*QuadClass*, *RootCode*, *BaseCode*, *EventCode*, and *GoldsteinScale*), in effect there is only one: the full CAMEO event code contains all the other codes. In other words, all characterization codes can be derived from the full CAMEO event code. Although they may be useful for aggregation, analysis, and data exploration (as we will see), the character codes are functionally redundant.

3.5 Location

There are three location types: *Event*, *Actor1*, and *Actor2*. Action geography and actor geography have the same structure, so their data types are the same. The geographic location is about where the actors were *when they did the event*. It is not the an intrinsic attribute of the actor. In other words, it is not about where an actor is from or with where an actor is affiliated. This section describes the various event and actor location fields. The concordance among the actor and location fields is in table 3.2

Table 3.2: Location Fields

Variable	Action	Actor1	Actor2
Geo_Fullname	ActionGeo_Fullname	Actor1Geo_Fullname	Actor2Geo_Fullname
Geo_Type	ActionGeo_Type	Actor1Geo_Type	Actor2Geo_Type
Geo_FeatureID	ActionGeo_FeatureID	Actor1Geo_FeatureID	Actor2Geo_FeatureID
Geo_CountryCode	ActionGeo_CountryCode	Actor1Geo_CountryCode	Actor2Geo_CountryCode
Geo_ADM1Code	ActionGeo_ADM1Code	Actor1Geo_ADM1Code	Actor2Geo_ADM1Code
Geo_ADM2Code	ActionGeo_ADM2Code	Actor1Geo_ADM2Code	Actor2Geo_ADM2Code
Geo_Lat	ActionGeo_Lat	Actor1Geo_Lat	Actor2Geo_Lat
Geo_Long	ActionGeo_Long	Actor1Geo_Long	Actor2Geo_Long

3.5.1 Location Field Descriptions

- **FullName** - This is the actor's name as it is used in the source text. As such, it can be a name for a country, state, city, or landmark. Names of countries are given as just 'Country Name'. For both US and World states, the names are given as 'State, Country Name.' For cities or landmarks, the names are given as 'City/Landmark, State, Country'.
- **Type** - This is the geographic scale. It can be broad or specific. Locations are organized into five classifications, each with a numeric code: 1 is Country, 2 is US State, 3 is US City, 4 is World City, and 5 is World State.
  NOTE: The GDELT handbook says that if the Type is 1, 2, or 5, then *FeatureID* will be blank. This is not the case, although it is possible to ignore *FeatureID* for those types.
- **CountryCode** - This is a 2-letter code for the country.
- **ADM1Code** - This is a 2-letter code for a state, province, or prefecture. The scale that is more specific than country but less specific than city or landmark.
- **ADM2Code** - This is a code that is more specific than *ADM1Code*: it is for for a county, city, or landmark (that is inside the state, province, or prefecture). If the location is international, then If the location is within the US, then the *ADM2Code* is the 2-letter state code with 2-digit county code.
- **FeatureID** - If the *Type* code is 3 (US City) or 4 (World City), then *FeatureID* is a unique numeric value. If the *Type* code is 1 (Country), 2 (US State), and 5 (WorldState), then the *FeatureID* is a text field. According to the GDELT handbook, the ID field should be unique. However, in the case of codes 1, 2, and 5, the *FeatureID* is not necessarily unique: some countries and states can only be differentiated by *Type*. For example, the *FeatureID* 'PA' could mean 'Paraguay' or 'Pennsylvania.' If

PA occurs with Type 1, it is the country Paraguay. If PA occurs with Type 2, it is the US state. Because of this, for a given GDELT data set, consider inspecting the data to see if *FeatureIDs* include these inconsistencies.

- **Lat** - Short for latitude, Lat is a Y-coordinate between -90 and +90 degrees. All locations are given a latitude to intersect with the longitude. Lat/long are coordinates used to specify precise locations via a grid system.
- **Long** - Short for longitude, Long is an X-coordinate between -180 and +180 degrees. All locations are given a longitude to intersect with the latitude. Lat/long are coordinates used to specify precise locations via a grid system.

According to the GDELT documentation, the *lat/long* are assigned based on the location. This implies that *CountryCode*, *ADM1Code*, and *ADM2Code* should fully determine *FeatureID*, and vice versa, and these should in turn fully determine *Type* and *Lat/Long* parts of event codes. In practice, some coding inconsistencies exist so that these relationships do not fully hold.

In contrast to these code fields, name fields (e.g. *FullName*) reflect the name of the location as it was found in the text and is not necessarily consistent (e.g., "Texas", "State of Texas", "TX").

3.6 Actor Information

An actor's characteristics and attributes are given in 3-letter codes and are combined to form the /textitActorX code. It is possible to query individual fields separately with these codes.

NOTE: The titles for attributes are taken from TABARI ACTORS dictionary rather than from the source documents. ta For example, a dictionary label for *Group* is 'Insurgents', but the source document may refer to the group as "radicalized terrorists".

- **ActorXName** - This is the name of the actor, a proper name.
- **ActorXCode** - This is the complete raw code comprised of codes for *Country*, *KnownGroup*, *Ethnic*, *Religion*, and *Type*.
  For example, AFGINSTALMED is a valid *ActorXCode*, and it contains: *CountryCode* AFG (Afghanistan); *Type1Code* INS (Insurgent); *KnownGroupCode* TAL (Taliban); and *Type2Code* MED (Media). The value in the *ActorXName* field is TALIBAN.
- **ActorXCountryCode** - This is 3-letter code of the actor's country affiliation. It is an attribute of the actor; it is not where an actor participated in an action. For example, if the source text says, "US President meets Supreme Leader in North Korea,"

then the *CountryCode* for *Actor1* (US President) is 'USA', while the *CountryCode* for *Actor2* (Supreme Leader) is 'PRK'.
Additionally, the *CountryCode* can be blank. For example, if the source text refers to an "unidentified gunman", then the country affiliation of the gunman is unknown.

- **ActorXKnownGroupCode** - This is applicable if the actor is associated with a known IGO, NGO, or rebel organisation. It can be blank.

- **ActorXEthnicCode** - This is the ethnic affiliation of the actor given that ethnicity is specified in the source document and has a code.
NOTE: Ethnic groups can also be coded as *Type* (e.g., ARAB), so some codes may not comprehensively capture ethnic affiliations. To better capture affiliations, consider using the Global Knowledge Graph's ethic, religious, and social group taxonomies.

- **ActorXReligionNCode** - This is the religious affiliation of the actor given that the religion is both specified in the source document and has a code. Some religious groups (e.g., JEW) can be given as geographic and type codes.
N is the number of codes associated with an actor. There can be multiple religious codes per actor. For example, Catholicism invokes Christianity for Code1 and Catholicism for Code2.

- **ActorXTypeNCode** - This is a 3-letter code for the *type* or 'role' of an actor. For example, types can be specific: Police Forces, Government, Military, Political Opposition, Rebels, etc. Types can also be broad: Education, Elites, Media, Refugees, or organizational classes like Non-Governmental Movement. Types can also be for the operational strategy of a group: Moderate, Radical, etc. It can be blank.
N is the number of codes associated with an actor. There can be multiple type codes associated with an actor. For example, an actor can have both 'Radical' and 'Media' types.

3.7 Discourse

These variables characterize the way that the news sources discuss the real-world event.

- **IsRootEvent** - This indicates whether the event was mentioned in the first paragraph of the document given by *sourceurl*.

- **NumMentions** - This is the number of mentions recorded for a real-world event.

- **NumSources** - This is the number of distinct news sources that mention a real-world event.

- **NumArticles** - This is the number of distinct articles that mention a real-world event.
- **AvgTone** - This is the average tone (calculated via NLP) of the articles that mention this real-world event.

For clarity, *NumArticles* refers to the number of different news articles that mention the event; *NumSources* refers to the number of different news sources in which those articles appear; and *NumMentions* refers to the number of different times that the event is mentioned.

For example, if the NY Times carried an article in which an event was mentioned twice and a second article that mentioned it once, while BBC carried one article that mentioned it twice, the number of sources would be two (NY Times and BBC), the number of articles would be three (two in the NYT and one in BBC), and the number of mentions would be five (2 + 1 + 2).

NOTE: There is an important limitation on these fields: They are recorded only during the first 15-minute interval during which GDELT initially recognizes the real-world event. Other articles that are discovered after this are not incorporated into these totals. In general, it may be preferable to use the records in the Mentions table rather than relying on these values.

3.8 The GDELT Mentions Table

The Mentions table lists each mention of a particular event. A single article can mention the same event more than once, resulting in multiple entries in this table.

NOTE: It is possible to use the Mentions table to create new variables for the Event table by aggregating mentions. For example, instead of a true/false *IsRoot* field, a field could be created that calculates the fraction of articles for which a mention of the event occurs in the first 5 sentences. Doing this for some fields would allow evading the time limitation of the Events table, which only includes information from the first 15 minutes after the event is discovered. For instance, fields (e.g., *NumMentions*, *NumSources*, *AvgTone*) could be recalculated in this way. Other new attributes (e.g., how long after the event occurred did it continue to be discussed) could be created.

3.8.1 Metadata

- **GlobalEventId** - This is a unique ID. It is the field that links to the Events table.

- **EventTimeDate** - This is equal to *dateadded* in the related record in the Events table (and is therefore redundant). It can be used to determine the date and timestamp for when the record was added to GDELT without consulting the Events table.
- **MentionTimeDate** - This is the date of when a mention was recorded by GDELT. (This is not necessarily the same as the the publication date for the article.)

3.8.2 Source Information

- **4MentionSourceName** - This is the source in which a mention occurs, e.g., BBC, NY Times.
- **MentionType** - This is an integer indicating the type of a source and how to interpret the *MentionIdentifier* field. For example, 1 is from a URL for web page, 2 is from a citation only, 5 is from JSTOR, 6 is from a non-textual source, etc.
- **MentionIdentifier** - This can depend on the *MentionType* but generally will be a URL If it is a journal article or paper, then it is a DOI. For other types, other unique identifiers are used.
- **MentionDocLen** - This is the length of the document (in English characters).
- **MentionDocTone** - This is the tone of the document as determined by NLP.
- **MentionDocTranslationInfo** - This is any translation information. It is given as a string with a series of flags indicating original language, translation method, etc.

3.8.3 In-Document Information

- **SentenceID** - This is the sentence where the event is mentioned in the source text. It is 1-based, meaning that the first sentence is ID'd by 1, the second sentence is ID'd by 2, etc.
- **ActorXCharOffset** - This is the location of an actor during the event as stated in the source text.
- **ActionCharOffset** - This is the location where the action of the event occurred as stated in the source text.
- **InRawText** - This field is either a 1 or a 0 and is a boolean flag recording. 1 is an event discovered in the document's raw text. 0 is an event discovered when rewriting with the TABARI system.

- **Confidence** - The confidence level is a score of certainty, given as a percentage (0-100), for the extraction of an event from an article. Low percentages indicate uncertainty whereas high percentages indicate certainty. Raw text extractions (*InRawText* = 1) rate higher than texts rewritten by the TABARI system (*InRawText* = 0).

Chapter 4

Data Types and Domains in GDELT_GA_Search

To search through GDELT data in a structured way requires an understanding of the way that these data 'work': what are the characteristics of the different kinds of data, and how can they be manipulated? To examine this, we begin by looking at what kinds of data are contained in the GDELT data set, and what operations are possible on them.

4.1 Theory: Fundamentals of Data Types

Data are not merely values, but are values that exist in structured collections. A value that represents something like a person's height, for example, can have nearly continuous values, but only positive values, and within this the difference between any two values is meaningful using straightforward subtraction; a value that represents the tone of a document may have one of a set of values ranging from some negative to some positive number, and the difference between two values may not strictly be the linear difference. A collection of names forms a different kind of data set, one with no natural (or semantically meaningful) order. It is useful to begin by characterizing the kinds of data that the GDELT_GA_Search tool will use. These are:

- **Nominal Data** - For this data type, the different possible values represent different categories. It is also known as **categorical data**.
- **Ordinal Data** - This data type is is like nominal data in that ordinal data are identified as categories. However, with ordinal data, there is an order or a sequence so that 'close' categories are more similar than 'distant' ones in some way.

- **Interval** - In this data type, distance is measurable in regular intervals (so that it is possible to say that one thing is 2x as far away as a second from some known point). This data types lacks a fixed reference point by which the entire data set can be related to an absolute.
- **Ratio** - This type is like interval data but with a known fixed point.

4.1.1 Bounded Data

An important consideration when developing the search strategy for a particular set of data is understanding that data set's boundaries. There are two types of boundaries: *terminal* and *wrap-around* data.

Terminal Data

Terminal data refers to a field that has a known upper and lower bound. An example is the *Goldstein Scale*, which by definition ranges from -10 to 10. These boundaries represent extreme ends of the scale. In other words, they are two pieces of data that lie at -10 and +10 are as far apart as possible (20 units) on this scale.

Wrap-Around Data

In contrast to terminal data, *wrap-around data* refer to a field that has a single axis but is cyclic. An example is longitude: someone traveling west across the middle of the Pacific will cross longitude values -178 (or 178°West), -179, -180 (equivalent to +180, and also 180°E), +179, +178, +177). The distance between any two values must be calculated in a way that recognizes this wrap-around, so that the angular distance between -179°and +179°is 2°[1] .

4.1.2 Unbounded Data

In theory, there could exist a kind of data for which the range is not known and is possibly indefinite, indicating a case of *unbounded data*. However, the GDELT_GA_Search toolkit acts on a fixed data set, so the ranges of all data in the data set can be known before the exploration of that data begins. Because of this, there is no reason to consider any data *unbounded* because it is always possible to determine what the boundaries are. For a fuller discussion, see Chapter **??**.

[1] ***NOTE***: for geographic distances across the (roughly) spherical earth, a true distance via 'great circle' route could be calculated, but this would involve more complicated math than what is presented here.

4.2 Functional Requirement of a Data Type in GDELT_GA_Search

To be employed as a data type in the GDELT_GA_Search toolkit, a data type must be able to perform the operations needed to move through the 'search space' of which a variable of that type forms an axis. In general, this means defining a range of values, and then expanding, reducing, or shifting that range along the axis.

In practice, this set of operations is carried out differently for different data types:

- For *nominal data*, a range is defined as some subset of the valid values. For example, if the range includes five possible country codes, then the current value may be three of these codes. Reducing this range means removing a code; expanding it means adding a code. However, expansion and reduction are done randomly. For expansion, one of the two remaining codes is chosen randomly and added to the set, while for reduction, one of the three current codes is selected randomly and removed. Because the data are purely categorical, there is no presumed order that would make one code a candidate for removal or addition over any other available option. The notion of an 'axis' is somewhat misleading in this case, because the values are unordered.
- For *ordinal data*, however, the data are considered to be in an order; hence, expanding or reducing a range means moving along the ordered axis. Suppose, for example, that our categories were letters of the alphabet, A, B, C, D, ... H. The current range might be a subset of this: D, E, F, G. Expansion could be done by adding 'C' or 'H', in effect moving upward or downward along the list. Reduction could be done by trimming the list, removing 'D' or 'G'. The range created by either of these operations would remain a set of contiguous elements. Discontinuous sets (e.g. "B, C, F", omitting D and E) are *not* possible (but can be accommodated by a nested query structure).
- For an ordinal data set that is *wrap-around*, expansion could go around the wrapping boundary. If the A, B, C, D, ... H set is wrap-around and the current values are 'F, G, H', expansion could be performed by adding 'E' or 'A', as 'E' is contiguous with F and 'A' is contiguous, across the wrapping boundary, with H.

4.3 Data Types and Data Fundamental Domains in GDELT_GA_Search Code

The GDELT_GA_Search toolkit defines several classes that represent data types; these are found in the *gdelt.query.domains.components* package, and include:

- *Double* For continuous values
- *Integer* For integer values

- *IntegerWrapped* For integer values that exist on a 'wrapped' scale
- *OrderedSet* For sets that contain a natural order
- *UnorderedSet* For sets that contain no natural order

4.4 Aggregated Data Domains

With some domains, it is natural to consider two or more to form a single data type. The most immediate example of this is *latitude* and *longitude*, which are commonly considered together. Searching on only latitude or only longitude, or independently on these two, is less intuitive and arguably less effective than considering them to form a single region on the Earth's surface.

Latitude and longitude form an especially nice example because of their differences. Latitude is bounded, ranging from -90 degrees (or 90°S) to +90 degrees (90°N). Longitude, conversely, is wrap-around, ranging from -180 degrees (or 180°W) to +180 degrees (180°E), such that 180°W = 180 °E. These two can be combined to form a region in multiple ways. For example, a single lat/long pair could be used to define a point on the earth's surface, and a radius used as a distance from this point, such that any GDELT record with lat/long lying within the circle defined by this point + radius would be a match. Doing this, however, would raise a number of small issues. One is that the math to compute the distance from the center point is nontrivial, and while not extremely difficult would nevertheless introduce a performance concern. A second is that the notion of 'expanding' or 'reducing' the data set is intuitive with respect the radius (which can be increased or decreased, bounded by a minimum of zero and a maximum of the half the circumference of the earth[2]), but with respect to the center point is more challenging: the center point could be moved, but this would leave out some of the values that were original in the circle (e.g., if the center moves east, some points on the western boundary are omitted).

Instead we implement latitude and longitude as a linked pair, where latitude is a continuous bounded domain and longitude is a wrap-around domain. Together these two values define a box on the earth's surface; the box can be expanded E-W and/or N-S by simply invoking the expand and reduce methods on the underlying domains independently.

[2]The radius could be replaced with an angular offset, but this only changes and does not resolve the issues.

4.5 GDELT Fields and their Domains in GDELT_GA_Search

The GDELT data types are mapped into the GDELT_GA_Search tool in specific ways. Most of these are commonsensical: Average Tone, for example, is a Double value. But note some specific exceptions. Event Root Code, for example, is mapped as an ordered set; although it is represented by numerical values, there is a more natural ordering to them (ranging from cooperation to conflict), and this is not the same ordering as the integers' values.

Table 4.1 shows the GDELT fields and the methods used to implement them in the GDELT_GA_Search toolkit. Note that some fields (e.g. 'filename') are not used and are omitted here.

Table 4.1: Fields in GDELT Events Table and their GDELT_GA_Search Types

GDELT Field	**Data Type**	**GDELT_GA_Search Designation**
day		
MonthYear		
FractionDate		
Year		
dateadded		
EventCode		
QuadClass		
EventRootCode	Integer	Ordered Set
EventBaseCode		
IsRootEvent		
GoldsteinScale	Continuous Bounded	Double
Geo_FullName		
Geo_Type		
Geo_CountryCode		
Geo_ADM1Code		
Geo_ADM2Code		
Geo_Lat	Continuous Numerical	Combined with Longitude
Geo_Long	Continuous Numerical	Combined with Latitude
Geo_FeatureID		
ActorXName		
ActorXCode		
ActorXCountryCode	Categorical	Unordered Set
ActorXKnownGroupCode		
ActorXEthnicCode		
NumArticles		
NumMentions		
sourceurl		
NumSources		
AvgTone	Continuous Bounded	Double

Chapter 5

Queries

A query consists of two parts:

- A set of criteria that provide a boolean answer (yes/no) for the question, "Does a specific GDELT event meet these criteria?"
- A method for applying these criteria across a set of GDELT events and creating a time series from them

The query criteria are the more fundamental of the two components: intuitively, criteria can include any question based on the GDELT fields, such as whether the events GoldsteinScale score falls within a particular range, or whether Actor1 is one of a specific set of actors. Queries can contain multiple combinations of these criteria, allowing very specific queries (e.g., match only if GoldsteinScale score is between -4 and -6, and only if Actor1 is 'USA') to be created.

The second component is subtler: the query can expect to evaluate a collection of events, and it should return a time series based on those events. However, the query has the ability to consider an event whose internal data says that it occurred at one time to be counted as having occurred at a different time. Moreover, some events can be counted more heavily than others (via *weighting*). These are discussed below.

5.1 Query Matching

Query matching is straightforward: a collection of criteria are applied, and an event is considered to match if it meets **all** of the criteria. This is a logical 'AND' operation: if the event falls outside the boundaries of any criterion, it fails to match.

Note that any individual criterion can be 'reversed', such that it becomes a negative of itself—that is, in the positive version a values matches if it falls inside the boundaries set by the query, whereas in the negative version it is a match only if it falls outside those boundaries. However, this is incidental to the overall process of matching all criteria in a given query: some criteria may be negative, but however each criterion defines a match, an event must still match all the criteria in a query to be consider an overall match.

5.2 Time Shifting

A function of the query is to create a time series. One implication of this is that date ranges are rarely used as criteria. A second implication is that the query translates some date-related field of the event into a specific point in time, and then assembles all of the points for all of the matching events into a time series. This can include a time shift: an event that has a date field of time T is shifted +/- some amount, so that it appears in the time series at a different point.

This shift allows for events in the GDELT data to be associated with events in the target time-series data (e.g., the social media data) with a lag from one to the other. For example, an event that occurs in real-time with heavy social media activity (like a protest) may show up in news sources a day later. In this case, shifting the GDELT events backward a day allows the alignment between these series to be more clearly seen.

5.3 Nested Queries

Queries have an additional capacity: they can contain other queries. Queries can therefore be nested into multiple levels. This must form an acyclic graph because no query can contain itself. The nesting of queries can have the following attributes:

- The nesting can be limited by limiting the process that creates it, such that only N 'levels' of nesting are permitted (see below)
- Nested queries can have different weightings, and thus can contribute differentially to the overall time series
- Nested queries can have different time shift values

With respect to the last two, one can imagine a parent query that matches all events with a specific value for Actor1 and adds them to the time series with a weight of '1'; a nested query that matches all events with the same Actor1 but specific actor for Actor2 and adds them to the time series with a weight of '4'; and another nested query that captures events with the same actors but Goldstein Scale values below -5, and adds these with a weight

of '8' and a time shift forward of 1 day. These different shifts and weightings become superimposed in the final time series that the query produces.

5.4 Mutation

Mutation occurs when a query has succeeded in yielding some successful results, and there is a desire to see if expanding or reducing the range of events it includes improves these results. Mutation can include the following elements:

- Mutating one or more of the individual criteria by expanding or reducing the range considered
- Adding or removing one or more nested queries
- Modifying the time shift being applied to the query
- Modifying the weighting being applied to the query

5.5 Specifying query behaviors for specific runs

The GDELT Properties File (described more fully in section 2.2.1) specifies:

- The minimum and maximum values for time shifts and visibility offsets
- Which elements of the query will be used to match a valid GDELT event
- For elements that are sets, which values are contained in the set
- How mutation will take place

The last of these is done via weighting; for example, in:

```
# MUTATION WEIGHTS
Actor1CountryCodesMutateWeight = 10
Actor2CountryCodesMutateWeight = 30
```

The two Actor country codes can both be selected for mutation, but they will be weighted differently, with Actor 2 being three times more likely to be selected than Actor 1.

Chapter 6

Prediction Methods

6.1 Definitions

Predictions are created from known periods of time and are used on unknown periods of time. As such, making predictions requires splitting time in two places.

1. The first split occurs *before* or *after* the point of transition from known to unknown. The *before* period is called **training** and the *after* period is called **testing**. The *point of transition* from before to after is called **time zero**.

2. The second split occurs between the the GDELT data (i.e., independent variables) and the target time-series (social media) event counts (i.e., dependent variables). The GDELT data belong to **X** and social media event counts belong to **Y**.[1]

Making these two splits results in a total of four periods:

- **X-Train** - GDELT training
- **Y-Train** - Social media event counts training
- **X-Test** - GDELT testing
- **Y-Test** - Social media event counts testing

In the canonical case, a prediction is made using the following: X-Train is compared to Y-Train to establish the relationship between them (e.g., a linear correlation). Then, X-Test is used to create a time series for the testing period; this is termed the Y-Predict. It is

[1] This vocabulary is based on graphs in which the values of the independent variable are placed on the (horizontal) x-axis and the values of the dependent variable are placed on the (vertical) y-axis.

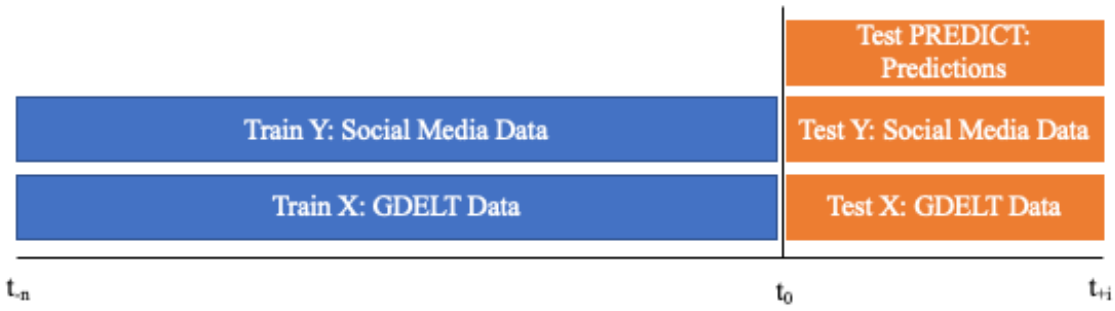

Figure 6.1: Conceptual View of the Prediction Process

then possible to score the performance of the prediction by comparing Y-Predict with the Y-Test values, which contain the 'correct' answer.

This can represent this as a function:

$$Y_{predict} = f(Y_{Train}, X_{Train}, X_{Test}) \tag{6.1}$$

or

$$y_{t0}, y_{t1}, y_{t2}, ...y_{ti} = f([y_{t-n}, ...y_{t-2}, y_{t-1}], [x_{t-n}, ...x_{t-2}, x_{t-1}], [x_{t0}, x_{t1}, x_{t2}, ...x_{ti}]) \tag{6.2}$$

While this is a canonical case, it is not the only case. For example, some predictions may need to be made without using X-Test. A true forecast into an unknown future, such as attempting to predict social media events days or weeks in advance, would be done without the values in X-Test, because the GDELT data in X-Test would also be unknown. In equation form:

$$Y_{predict} = f(Y_{Train}, X_{Train}) \tag{6.3}$$

Typically, the training period is longer than the prediction period. This is especially so in the second form, as predicting into the unknown without the X-Test values available is difficult.

The units for the time series values are usually 'days', so that the series values are 'events per day'. However, this can be changed and the series can be created for any unit of time (e.g., 4 hours). The limit is the resolution of the data. In GDELT, the event data

are identified by the 'day' field, which is a calendar date with no meaningful time stamp; however, the field *dateadded* field can be used, and this has a 15-minute resolution. The dateadded field can also be more appropriate if what is being examined is the relationship between the dependent variable (e.g., social media data) and the appearance of an event in the news rather than the event itself—that is, when the event was being discussed in the news and not when the event actually happened.

6.2 Prediction Methods

6.2.1 Raw

6.2.2 Base Values

6.3 Replay

A category of predictions known as **replay** has a simple idea: take values from the training period and repeat them in the testing period.

Variations on this theme are possible. For example, given the need to predict 8 days into the future, one could simply replay the 8 days immediately prior to t_0. Alternatively, one could align the days being replayed by day of the week, to catch weekly cycles of high and low activity.

6.3.1 Linear Regression

Linear Regression Weighted

6.4 Long Short-Term Memory

6.5 Elastic Net

6.6 Gradient Boost

6.7 Lasso

6.8 Adding Prediction Methods

6.8.1 Java

Creating new prediction methods in Java is straightforward: All prediction classes must implement

6.8.2 Python

Chapter 7

Scoring Methods

7.1 Root Mean Square Error (RMSE)

7.2 Normalized Root Mean Square Error (NRMSE)

7.2.1 Mean

7.2.2 MinMax

7.2.3 Standard Deviation

7.3 Correlation (Positive Only)

7.4 Input Correlation

7.4.1 Trimmed

7.4.2 Positive Only

7.5 Adding Scoring Methods

7.5.1 Python

7.5.2 Java

Chapter 8

Downloading GDELT Data

8.1 Using the Events and Mentions Scrapers

GDELT_GA_Search includes a pair of tools to allow you to acquire GDELT data. The *GDELT_Events_Scraper* tool downloads GDELT Event data; the *GDELT_Mentions_Scraper* downloads GDELT Mention data. Downloading Mention data is a second step that relies on an existing corpus of event data; when downloading mentions, only mentions associated with events in the events corpus are stored.

Running these tools is best done in an IDE; they are not currently configurable from the command line, and the code must be changed to download specific subsets of data. The result of these tools is a file (user-specifiable, but customarily either GDELT_Events_Data.csv or GDELT_Mentions_Data.csv) that contains the GDELT data. The data are in the standard GDELT format; they are not transformed. For historical reasons, the Mentions file has a header line, while the Events file does not.

GDELT data is stored online in a repository of ZIP files that are identified by a time stamp code. The tools here are given a start and end time range and move from beginning to end, downloading the files, unzipping them, and assembling the results into a file that can be used as GDELT_GA_Search input. The ZIP files can optionally be retained or deleted; retaining them allows the routine to be run again without hitting GDELT's servers, but the files are quite large to store.

8.2 Setting download parameters

Each of the tools allows you to specify the date/time range of files to be downloaded, and a few other options. In total the options are:

- startTime The earliest file to download (inclusive)
- endTime The latest time to download (not inclusive; the routine stops before downloading this file.)
- dataPath A path to directory in which the data output will be stored.
- ZIP_PATH_ROOT the path to the directory in which the ZIP files will be stored when downloaded (and optionally retained)
- LoopLimit A debug fail-safe to stop from downloading too many files; this is in place because the files are very large and you may want to stop downloading at some specific point (but this could also be limited by the time ranges)
- keepZip True if the ZIP files should be kept; false if they should be automatically deleted once expanded
- fileName The name of the file that will be created
- appendToFile True if the output file should be appended to; false if it should be deleted before writing begins.
- sleepInterval An interval in milliseconds that the routine should wait between downloading files. This is to ensure that the server does not believe it is being attacked. The default is 500. Reducing it can lower the total time the routine needs, but the response of GDELT's servers to faster downloads is not known.

Additionally, the Events scraper has parameter ('PROB') that can be set to a number lower than one; this will probabilistically exclude lines from the original files. This is a crude form of random sampling. If set to zero and the keepZip parameter is set to true, the routine effectively downloads GDELT ZIP files and does no other action.

The Mentions scraper has an additional required parameter that gives the path and name of the Events table to used to create the GDELT corpus; only mentions related to events in the corpus are loaded.

8.3 Filtering Events

The code can be modified to download only include specific events. Line 114 in the Events scraper includes an example. Filtering would have to be done in a customized way and is

not covered in detail here.